NOVA IMS Information Management School

MAA

Mestrado em Métodos Analíticos Avançados
Master Program in Advanced Analytics

G-SOMO: An Oversampling Approach based on Self-Organized Map Oversampling and Geometric SMOTE

Rene Rauch

Dissertation presented as partial requirement for obtaining the Master's degree in Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

G-SOMO: AN OVERSAMPLING APPROACH BASED ON SELF-ORGANIZED MAP OVERSAMPLING AND GEOMETRIC SMOTE

by

Rene Rauch

Dissertation presented as partial requirement for obtaining the Master's degree in Advanced Analytics

Advisor: *Fernando Bacao*

ABSTRACT

Traditional supervised machine learning classifiers are challenged to learn highly skewed data distributions as they are designed to expect classes to equally contribute to the minimization of the classifiers cost function. Moreover, the classifiers design expects equal misclassification costs, causing a bias for underrepresented classes. Thus, different strategies to handle the issue are proposed by researchers. The modification of the data set managed to establish since the procedure is generalizable to all classifiers.

Various algorithms to rebalance the data distribution through the creation of synthetic instances were proposed in the past. In this paper, we propose a new oversampling algorithm named G-SOMO, a method that is inspired by our previous research. The algorithm identifies optimal areas to create artificial data instances in an informed manner and utilizes a geometric region during the data generation to increase variability and to avoid correlation.

Our experimental setup compares the performance of G-SOMO with a benchmark of effective oversampling methods. The oversampling methods are repeatedly validated with multiple classifiers on 69 datasets. Different metrics are used to compare the retrieved insights. To aggregate the different performances over all datasets, a mean ranking is introduced.

G-SOMO manages to consistently outperform competing oversampling methods. The statistical significance of our results is proven.

KEYWORDS

Oversampling; Imbalanced Learning; Clustering; Synthetic Data Generation

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AUC ROC Area Under The Curve Receiver Operating Characteristics

DT Decision Tree

GBC Gradient Boosting Classifier

G-Score Geometric Mean Score

G-SMOTE Geometric Synthetic Minority Oversampling Technique

G-SOMO Geometric Self-Organizing Map Oversampling

IR Imbalance Ratio

KNN K-Nearest Neighbors

LR Logistic Regression

ROS Random Oversampling

SMOTE Synthetic Minority Oversampling Technique

SOM Self-Organizing Map

SOMO Self-Organizing Map Oversampling

1 INTRODUCTION

Learning from imbalanced datasets remains a challenge for supervised machine learning classifiers and therefore needs to be addressed by the research community. Class imbalance refers to classification problems where the target class is unequally distributed and therefore algorithms trained with imbalanced data tend to be heavily biased towards the majority class [Hoens and Chawla, 2013]. The paper on hand will present an effective way to deal with imbalanced datasets, the focus hereby is set on highly skewed binary datasets. Commonly known domains dealing with these challenges are fraud detection, medical diagnosis, risk management, airborne imagery, face recognition and of forecasting of ozone levels [Akbani et al., 2004].

Datasets with an unequally distributed target class are identified by their Imbalance Ratio (IR). The class represented by most of the instances is named the majority class, while the other class is the minority class [Chawla et al., 2003]. The Imbalance Ratio is defined as the ratio between the majority class and each of the minority classes. It is important to notice that the data imbalance is an integral part of the problem itself, implying that the cost of a false negative prediction is usually much larger than the cost of a false positive. Considering a machine learning based system to make medical decisions, if the model predicts a patient to carry a disease, even though the patient is healthy, the wrong decision will be revealed in further medical examinations. On the other hand, if the model classifies the patient to be healthy, even though the individual carries a disease, tremendous harm is caused [Wan et al., 2014]. Therefore, we can see that the correct classification of the minority class is usually more important than the correct prediction of the majority class.

There are two main explanations for the poor ability to learn from imbalanced data sets. The first reason to explain this behavior is, that during the training process of the model the instances of the minority classes contribute less to the minimization of the algorithms cost function. As a result, the model has a higher incentive to classify new instances as majority instances, since there is a higher chance of a correct classification. Second, due to the limited number of samples, the model faces difficulties to differentiate between outliers and important instances. This implies that the ability to create a decent model is not just dependent on the machine learning algorithm and its parameters, but also from the data itself and its imbalance ratio

Unlearned practitioners who trained models on imbalanced data might be lead to false conclusions while validating their models based on their accuracy. Accuracy is a measure that is heavily biased towards the majority class, the metric might imply a decent score, even though none of the minority classes are correctly classified. Assuming a dataset with 99% majority instances and only 1% minority instances, the model accuracy would still be 99% without classifying a single minority instance correctly. To address this concern, other validation metrics have to be selected that consider the importance of the minority class. A more detailed outline of the chosen metrics follows in section 5.1 'Metrics'.

In general, there are three different options to handle an imbalanced data set [Fernández et al., 2013]. The first option is the modification of the dataset itself, hereby one can create new synthetic data instances to the minority classes by over-sampling or remove existing data instances from the majority instances through under-sampling. Both approaches can also be combined to a hybrid approach, the general objective of the approaches is to minimize the skewed distribution of the classes within the dataset and to shift the classes towards an equal distribution. The second option is to adjust the cost function of the supervised learning algorithm. During the training process, the model will be heavily penalized for falsely classified minority class instances. The latter implies to modify the training process and create explicit algorithms to handle the class imbalance. The third approach focuses on the modification of algorithms that reinforce the learning towards the minority class. We decided to tackle the challenge of imbalanced datasets by oversampling. The reasons to modify the dataset and not the algorithm is that after the dataset modification any supervised machine learning algorithm can be used, without any further modification. With our proposed algorithm we focus on oversampling since it provides the advantage that none of the majority instances need to be removed and no valuable information is vanished, as it might happen through under-sampling.

The method presented in this paper extends the Self-Organizing Map Oversampling algorithm by making use of the Geometric SMOTE algorithm. SOMO is a cluster based algorithm that leverages the Self-Organizing map algorithm and managed to outperform related algorithms over various datasets [Douzas and Bacao, 2017b]. The algorithm preserves the topology while reducing the dimensionality to a two-dimensional representation. The emerging clusters are used to create minority instances within, but also between neighboring minority clusters by leveraging the SMOTE algorithm. The algorithm presented in this paper utilizes the procedure of

the SOMO algorithm and replaces the SMOTE algorithm with the Geometric SMOTE algorithm. G-SMOTE generates synthetic samples in a geometric region of the input space, around each selected minority instance. G-SMOTE also proved significant improvements in the generated data quality [Douzas and Bacao, 2017a]. Due to the combination of both algorithms, the naming choice of the algorithm proposed by us is G-SOMO, Geometric Self-Organizing Map Oversampling.

Following this introduction, we will outline related methods that proved to be efficient in the second chapter. Subsequently, we will expound the shortcomings of classifiers and explain our motivation for the oversampling method G-SOMO in chapter three. In the fourth chapter, we will discuss the G-SOMO algorithm in detail and present each step individually. The following chapter Research Methodology covers our experimental pipeline to validate the performance of our proposed method. We will present our significant results in chapter six. The last chapter will summarize our findings and provides a prospect for further research

2 RELATED WORK

The section outlines the current state of the art over-sampling methods. Over-sampling methods generate synthetic examples for the minority class and add them to the training set, therefore additional information is created. In contrast to over-sampling, under-sampling methods reduce samples of the majority class to establish a class balance This implies that information is excluded, which might affect the learning process negatively when the data set is small. Both methods have shown to be effective, depending on the problem addressed [Chawla et al., 2002]. More information on under-sampling methods can be found in [Ganganwar, 2012] and [Yen and Lee, 2006]. Synthetic data instances can be created uninformed, by randomly duplicating minority instances or informed, by identifying areas where oversampling is deemed to be most effective [Last et al., 2017].

The modest form of oversampling is Random Oversampling (ROS). ROS is an uninformed approach, which randomly selects minority samples and duplicates them exactly without any selection criteria. The method stands out through its simplicity, but proved to increase the risk of overfitting enormously, since the same information is used multiple times during the training process, no instances that clarify the decision boundaries are created.

The most popular approach among practitioners in the domain of oversampling is SMOTE, introduced in 2002. The algorithm chooses a random minority instance and identifies its k nearest neighboring minority instances. The parameter k is chosen beforehand. A synthetic instance is created on a random point along a line segment joining the selected minority instance and one of its neighbors [Chawla et al., 2002]. Depending upon the amount of over-sampling required, neighbors from the k nearest neighbors are randomly chosen. Figure 1 constitutes the process of SMOTE.

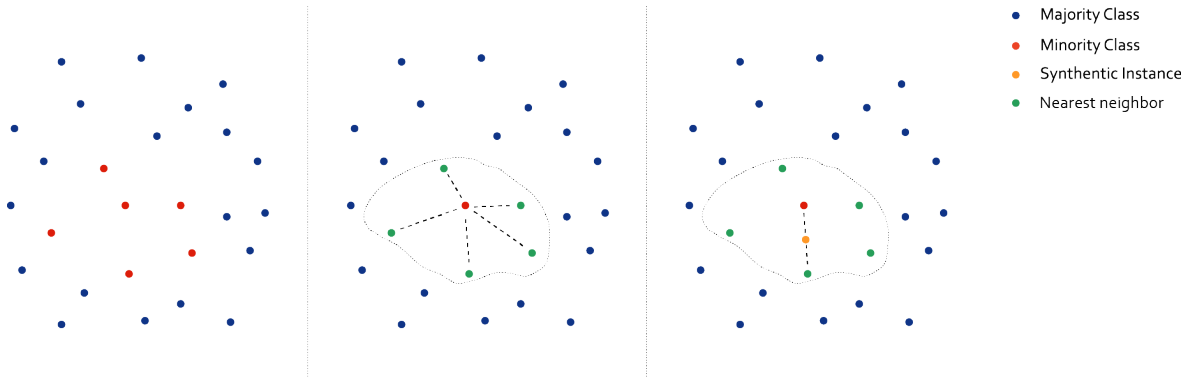

Figure 1: SMOTE Algorithm, inspired by [Schubach et al., 2017]

Compared to ROS the synthetic samples are more generalizable and therefore the high risk of overfitting is reduced. However, SMOTE has several drawbacks. First, the algorithm randomly selects a minority instance for oversampling with uniform probability. Hereby it manages to tackle between-class imbalance, while within-class imbalance is ignored. Areas that were highly populated by minority samples will expand, while smaller areas of minority samples will remain sparse [Prati et al., 2004b]. Second, the algorithm promotes the generation of noisy samples. When selecting a noisy sample and the linear interpolation to the nearest neighbor is created, a new noisy sample might be created, due to the great distance between them. It can not distinguish overlapping class regions from safe areas [Bunkhumpornpat et al., 2009].

To tackle the problems of SMOTE several modifications were created. SMOTE + Edited Nearest Neighbor removes any misclassified instances after the creation of synthetic samples, by applying the edited nearest neighbor rule [Maria, 2004]. Safe-Level SMOTE applies a weight degree to differentiate between noisy and safe instances [Bunkhumpornpat et al., 2009]. Furthermore, G-SMOTE manages to extend the linear interpolation of SMOTE by introducing a geometric region where the data generation process occurs. Thereby, the algorithm increases the data variability and prevents additional correlation between the created samples [Douzas and Bacao, 2017a]. Further algorithms like 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] are trying to avoid noisy samples and focus on hard to learn instances. Hereby, the borderline samples of the classes are identified to use the informative minority class instances.

The prior algorithms focus exclusively on between-class imbalance [Nekooeimehr and K. Lai-Yuen, 2015]. The second challenge is an informed approach, the within-class-imbalance. Within-class-imbalance refers to identifying sparse or dense clusters of minority or majority instances. To tackle this challenge, different clustering based oversampling methods have been proposed to identify areas, where oversampling is most effective. These methods are segmenting the instances and then apply traditional oversampling methods specific to each segment.

Cluster SMOTE utilizes the k-means cluster algorithm, to identify clusters with a specific threshold of minority instances, before applying SMOTE within these clusters [Cieslak et al., 2006]. In addition, K-means and SMOTE applies a similar approach, but also considers the density within the clusters by assigning samples weights. A high sampling weight corresponds to a low density of minority samples and yields more generated samples [Last et al., 2017]. Furthermore, multiple suggestions for the Hyperparameter selection are provided. DBSMOTE provides another approach to cluster based over-sampling, by applying the density-based DBSCAN algorithm to discover arbitrarily shaped clusters to create artificial data instances along the shortest path from each minority class instance to a pseudo-centroid of the cluster [Bunkhumpornpat et al., 2011]. A-SUWO uses a hierarchical clustering approach and adaptively determines the size to oversample each sub-cluster using its classification complexity and cross-validation [Nekooeimehr and K. Lai-Yuen, 2015].

The Self Organizing Map Oversampling (SOMO) algorithm applies a self-organizing map to create a two-dimensional representation of the multidimensional input space and creates inter- and intra-cluster synthetic samples based on the underlying manifold structure [Douzas and Bacao, 2017b]. The algorithm transforms the input in a two-dimensional space of clusters and applies the SMOTE over-sampling algorithm to generate synthetic instances in the minority clusters and between neighboring minority clusters. The densities in and between the clusters are also taken into account by utilizing the average Euclidean distance. The algorithm stands out through its ability to not only oversample the identified clusters, but also between neighboring minority clusters since those areas are also safe and effective for data generation.

Besides clustering based algorithms, further over-sampling algorithms evolved, that are based on ensemble methods like SMOTEBoost [Chawla et al., 2003] and DataBoost-IM [Guo and Viktor, 2004]. The boosting process is used to identify hard to learn instances, which are used to separately generate synthetic examples for the majority and minority classes [Guo and Viktor, 2004].

3 MOTIVATION

The section briefly outlines the challenges and shortcomings of competitive oversampling methods and clarifies the motivation for our proposed algorithm G-SOMO

A classifiers performance is significantly influenced by the positioning of the instances in the dataset. As outlined in figure 2, the instances of a class can be spread over the boundaries of the majority class, which complicates the classifiers task [Tang and Gao, 2007]. This is a common challenge in real-world problems. The different types of instances can be seen as safe and unsafe instances, while the latter also differentiate between borderline and outlier instances [Sáez et al., 2016].

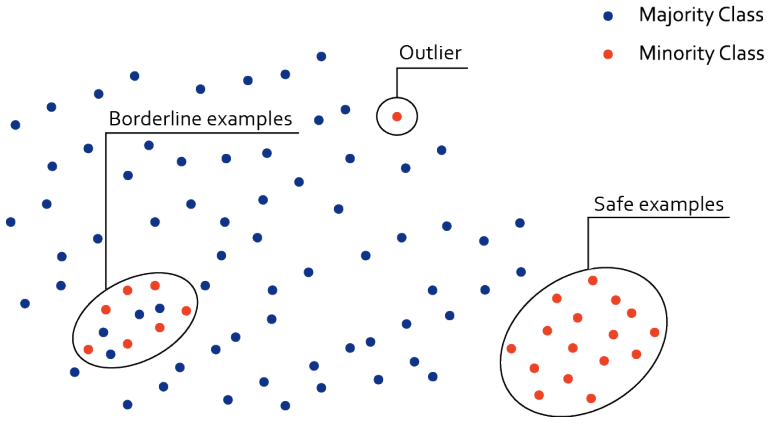

Figure 2: Challenging data formations, inspired by [Sáez et al., 2016].

Safe instances are located in the homogeneous regions and populated by the examples from one class only [Rodriguez et al., 2012]. They are clearly separated from examples of other classes. Most classifiers are able to correctly identify those instances [Prati et al., 2004a]. Instances that are not clearly separated are considered unsafe ones like borderline instances and outliers [Kubat and Matwin, 2000]. Borderline instances are placed in the boundary region between classes, where instances of multiple classes overlap [Sáez et al., 2016]. Outliers are isolated distinct instances, also termed as noise.

Learning algorithms are challenged when they are confronted with overlapping areas between classes, especially unsafe instances. Therefore, oversampling algorithms should be able to carefully consider in which areas to create artificial instances and which areas to ignore. Inefficiencies that were observed with SMOTE and similar algorithms are the generation of noisy instances penetrating the area of majority classes, as well as the generation of nearly duplicated examples. Noisy examples can be created when applying k-nearest neighbor approaches, since they can either choose a noisy instance as a starting point or select a noisy instance as nearest neighbor, as seen in figure 3a and 3b. Similar or nearly duplicated instances are created while generating new artificial instances within the borders of a minority cluster since they do not help to strengthen the decision boundary and might lead to overfitting, as it can be seen in figure 3c. Figure 3d outlines the case, in which the parameter k is too high and individuals from another cluster are selected.

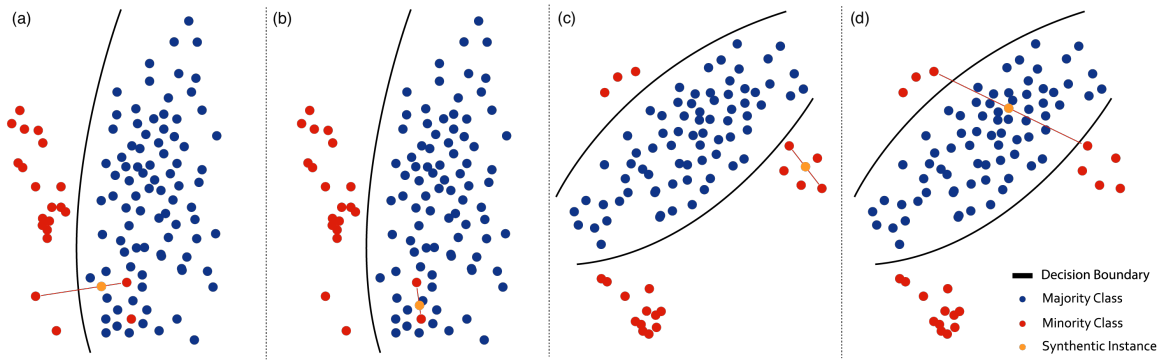

Figure 3: Shortcomings of SMOTE, inspired by [Douzas and Bacao, 2017b]

The Self-Organizing Map Oversampling (SOMO) algorithm improved the selection criteria of the minority class samples, which are used to generate synthetic examples. Through the synthetic data generation in and between neighboring minority clusters, as well as the consideration of the density, the algorithm also manages to generate the synthetic instances in more productive areas of the data space [Douzas and Bacao, 2017b]. Therefore, the generation of noisy examples is avoided. SOMO manages to identify efficient oversampling areas, the generation of synthetic instances is then handled by the SMOTE algorithm, which still provides several drawbacks. Challenges, as outlined in figure 3d, are prevented through the previous generation of clusters, nevertheless, the SMOTE algorithm introduces a high correlation between the samples, since it only creates synthetic instances that are on the line segment between two instances.

Geometric SMOTE (G-SMOTE) extends the linear interpolation mechanism by introducing a geometric region, in which the data generation process occurs [Douzas and Bacao, 2017a]. Therefore, the algorithm extends the linear interpolation mechanism and provides a geometric region in which the data generation occurs. Figure 4 illustrates the idea behind the data generation based on a geometric region, instead of on the line segment.

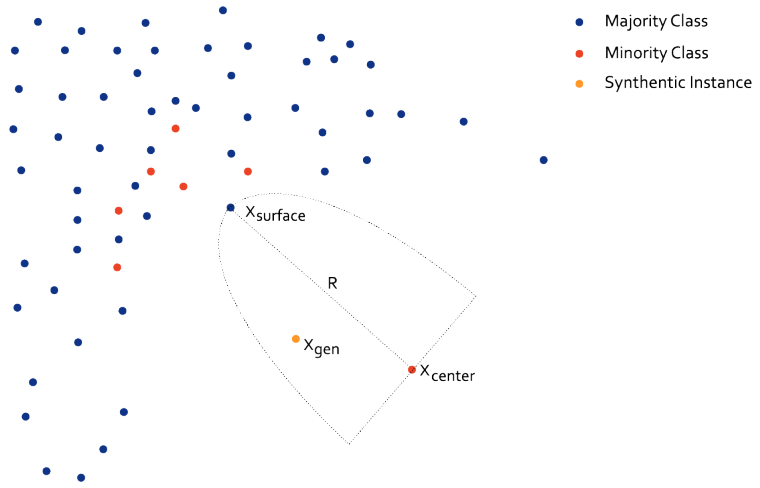

Figure 4: Synthetic data generation based on a geometric region, inspired by [Douzas and Bacao, 2017a]

Besides utilizing a geometric region for the data generation, G-SMOTE also introduced the majority selection strategy, which successfully prevents the scenario shown in figure 3a, 3b and 3d. The majority selection strategy prevents the creation of synthetic instances between two minority instances if the distance to a majority instance is shorter. Instead, the synthetic minority instance is created between the initially selected minority instance and the majority instance that had a shorter distance, as it can be seen in figure 4. G-SMOTE proved to significantly increase the performance of SMOTE.

SOMO provides an efficient process to identify areas that should be populated with minority instances, but lacks the improvements provided by G-SMOTE. On the other hand, G-SMOTE exploits an intelligent and efficient approach to generate synthetic instances, but lacks the ability to identify attractive regions for synthetic instances. In this paper we propose a combination of both techniques, leveraging the benefits of both algorithms.

4 PROPOSED METHOD

The previous section outlined the inefficiencies of the SMOTE algorithm, as well as of SOMO and G-SMOTE. The proposed method G-SMOTE manages to tackle them, by utilizing the best characteristics of both algorithms.

- G-SOMO manages to improve the selection criteria by leveraging Self-Organizing Maps.
- The method considers the density, based on the average Euclidean distances, in and between clusters when creating artificial data instances.
- G-SOMO creates synthetic instances within a geometric region, avoiding correlation between the instances.
- Through the combined selection strategy, the creation of noisy examples is avoided. This should already be prevented through the preselection of minority clusters, but still provides an advantage, in case the parameters to cluster were not selected carefully.

The proposed method consists of three stages. Initially, a Self-Organizing Map (SOM) is applied to the normalized input data set. The SOM algorithm creates mappings of high dimensional data space into low-dimensional space in such a way that the topological relations of the input patterns are preserved [Köküer et al., 2007]. To train a SOM, the Euclidean distance between the input vector and all neural weights has to be calculated. The Neuron that has the shortest distance to the input vector (the winner) is chosen and its weights are slightly modified to the direction represented by the input vector. Afterward, the neighboring neurons are taken and their weights are modified in the same direction [Brocki and Korzinek, 2007]. Due to SOMs ability to preserve topological relations from high dimensional input spaces, insights in the underlying data structure can be retrieved by analyzing the (usually) two-dimensional output map. In the second stage of the algorithm, the filtered clusters are defined, clusters where the minority class is dominating over the majority class. A weighted approach is used to create synthetic instances in the filtered clusters and also between neighboring filtered clusters. The last stage of the algorithm is marked by the creation of synthetic instances within the identified formations. The synthetic data generation is based on a geometric region, that is formed between the current minority instance and its selected neighbor. The shape of the geometric region varies, depending on its hyperparameter. Synthetic instances are created within the geometric region, avoiding the creation on a line segment as suggested by SMOTE.

4.1 G-SOMO ALGORITHM

The G-SOMO algorithm is the efficient combination of the SOMO and G-SMOTE algorithm, introduced in [Douzas and Bacao, 2017b] and [Douzas and Bacao, 2017a]. The complete G-SOMO algorithm is listed below.

Algorithm 1: Pseudo code for G-SOMO implementation

G-SOMO algorithm($args$):

Input : S_{maj} (Set of majority class samples)
S_{min} (Set of minority class samples)
N_{intra} (Total intracluster number to be generated)
N_{inter} (Total intercluster number to be generated)
$filtered\_cluster_{ratio}$ (Treshold to identify a cluster as filtered)
$inter\_intra\_cluster_{ratio}$ (Ratio of inter- and intracluster generated samples)
$SOM_{parameters}$ (Parameters of SOM algorithm)
α_{trunc} and α_{def} (Parameters to form the geometric region)

Algorithm

1. Normalize the data and train a SOM on the input data set, $S = S_{min} \cup S_{maj}$.
2. Identify each node in the map as a cluster, where cl describes all clusters $\{1,2,...,N^2_{Grid}\}$. N_{Grid} is the dimension of the grid, one of the SOM parameters.
3. Define the number of minority instances as n_{+i} and the number of majority instances as n_{-i} for each cluster $i \in cl$.
4. Calculate the imbalance ratio IR_i for each cluster $i \in cl$ as $IR_i = \dfrac{n_{+i}+1}{n_{-i}+1}$.
5. Identify the filtered set cl_f of cluster labels as $cl_f = \{i \in cl : IR_i > filtered\_cluster_{ratio}$.
6. In each filtered cluster $i \in cl$ calculate the average Euclidean distance $dist_i$ across all pairs of positive class instances belonging to the cluster.
7. Calculate the density factor for each filtered cluster as $den_i = \dfrac{n_{+i}}{dist_i^2}$.
8. Identify the density factor of each filtered neighboring units i and j as $den_{ij} = den_i + den_j$ for each combination of $i, j \in cl_f$.
9. Define the sampling weights
   9.1 Calculate the intracluster weights as $w_{intra} = \dfrac{1/den_i}{\sum_{i \in cl_f} 1/den_i}$
   9.2 Calculate the intercluster weights as $w_{inter} = \dfrac{1/den_{i,j}}{\sum_{i,j \in cl_f} 1/den_{i,j}}$
10. Define the number of artificial instances to be created
    10.1 For each filtered cluster generate $w_{intra,i} \cdot N_{intra}$ artificial minority samples
    10.2 For each neighborhood combination of filtered clusters i,j generate $w_{intra,ij} \cdot N_{inter}$ artificial minority samples
11. **for** *each filtered minority cluster formation and neighborhood formation* **do**
    11.1 Shuffle S_{min_f}, where S_{min_f} are the minority instances of the current formation
    repeat
        11.2.1 Select x_{center} from S_{min_f}
        11.2.2 Apply the combined selection strategy
        11.2.3 Generate a random point inside the unit-hypersphere centered at the origin of the input space
        11.2.4 Truncate the current unit hyper sphere
        11.2.5 Deform the unit hyper sphere to a hyper spheroid
        11.2.6 Rescale the created sample x_{gen}
        11.2.7 Add the sample x_{gen} to the set of generated instances in the current formation
    until $N_{instances}$ *of synthetic minority instances are created in the current formation*;

end

Output: X' (oversampled matrix of observations)

4.2 EXPLANATION OF G-SOMO

The G-SOMO algorithm relies on filtered clusters, which are areas where a specific minority class dominates over the majority class. These regions can be considered as safe areas for the generation of minority samples. Outliers or noisy examples should belong to non-filtered clusters and are therefore ignored. The density and weights of each filtered cluster are calculated to create new instances in each filtered cluster (intracluster) and between filtered clusters that are neighbors on the topological output map of the SOM algorithm (intercluster). The geometric region used for the data generation increases the variety of the generated instances.

Parameters: The G-SOMO algorithm requires the following parameters. S_{maj} representing the instances of the majority class, while S_{min} represents the instances of the minority class. N_{intra} is the absolute number of artificial instances that will be created within clusters, N_{inter} represents the absolute number of artificial instances that are created between neighboring filtered clusters. N_{intra} and N_{inter} are obtained by the total number of artificial samples split by the $inter\_intra\_cluster_{ratio}$, which describes the distribution of the total amount of artificial samples to be generated between the inter and intracluster process. The $filtered\_cluster_{ratio}$ has to be defined between 0 and 1, describing the threshold to accept a cluster as filtered. One notable SOM parameter is the size of the grid, which describes the number of nodes and therefore clusters. Typical parameters to form the geometric region such as α_{trunc} and α_{def} are required, their function will be explained in the corresponding steps.

Step 1: The first step of the algorithm is to normalize the input data and apply the SOM algorithm with the provided parameter. The data is normalized that each feature has a mean of 0 and a variance of 1. This step is necessary to assure that each feature is aligned on the same scale when assigning the best matching unit by utilizing the Euclidean distance during the training phase of the SOM algorithm. After normalizing the data, the correct size of the grid has to be chosen, which is a crucial parameter of the G-SOMO algorithm. The high dimensional input space will be transformed into a two-dimensional grid that consists of N_{Grid}^2 clusters, which are used to identify safe areas for the data generation process. The challenge hereby is to select a value allowing to discriminate between sparse and dense minority class areas. A very small value will not be able to identify subclusters, the identified clusters will have a very large size of instances. Assuming a uniform distribution, one can set the threshold to $\sqrt{|S_{min}|}$ to ensure that each cluster contains on average one minority class, but the assumption of the distribution is unreliable for real-world problems. High values of N_{Grid} will result in more smaller sized clusters, which might lead to filtered clusters in areas that we would usually like to ignore, because they are considered outliers. $\sqrt{|S_{maj}|}$ provides a reasonable upper bound for N_{Grid}. The optimal value for N_{Grid} is dependent on the characteristics of the data set and can only be approximated in an experimental approach.

Step 2 - 5: The filtered clusters are identified by their Imbalance Ratio $IR_i = \frac{n_{+i}+1}{n_{-i}+1}$, where n_{+i} is the number of instances belonging to the minority class and n_{-i} are the instances of the majority class. Clusters having a higher ratio than the given $filtered\_cluster_{ratio}$ threshold are considered to be filtered clusters of the current class. The parameter $filtered\_cluster_{ratio}$ is usually set to 0.5. The G-SOMO extension will not create any artificial instances if no filtered clusters are identified. In this case, the minority instances are sparse and the distinction between noisy and informative instances is not possible. One possibility hereby is to increase the parameter N_{Grid} to create more clusters.

Step 6 - 8: For each identified filtered cluster, the average Euclidean distance is calculated between the minority instances. Using the average Euclidean distance, we assign a density factor to each filtered cluster $den_i = \frac{n_{+i}}{dist_i^2}$. This measure provides information about the distribution of the instances within each cluster and will be helpful in further stages to assign the correct amount of artificial samples to each cluster. For each topological neighboring combination of filtered clusters, the density is defined as the sum of both individual density factors. Filtered clusters that are not topological neighbors, or have no topological neighbors are excluded from the process. Figure 5a illustrates the SOM Grid with all identified filtered clusters, figure 5b outlines the topological neighbor structures. In the next step, we calculate the relative weight of each filtered cluster and each filtered cluster neighborhood relationship by using its density. The weight determines the number of synthetic samples that will be created in each filtered cluster and between filtered clusters.

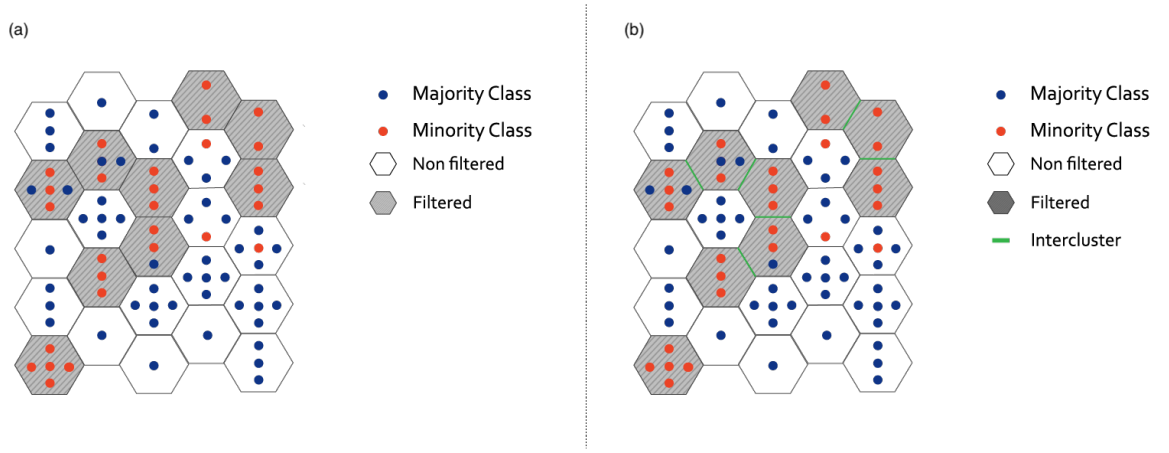

Figure 5: Overview of identified clusters and neighboring relations, inspired by [Douzas and Bacao, 2017b]

Step 9 - 10: The following steps determine an efficient number of synthetic instances for each filtered cluster and each neighboring relation of filtered clusters. N_{intra} and N_{inter} provide the guideline of the total synthetic instances to be created in the inter and intracluster process. Utilizing the density information, a relative weight is calculated for each filtered cluster as $w_{intra} = \frac{1/den_i}{\sum_{i \in cl_f} 1/den_i}$, where 1 divided by the density of the current cluster is divided by 1 through the sum of all densities of filtered clusters. Hereby, we obtain a relative weight of each filtered cluster based on the cluster size and the density of each cluster. N_{intra} times the weight for each specific filtered cluster results in the amount of artificial data that is generated in each filtered cluster. Afterwards the weights of the neighboring clusters are calculated as $w_{inter} = \frac{1/den_{i,j}}{\sum_{i,j \in cl_f} 1/den_{i,j}}$ in a similar manner as the intra weight calculation. Once each neighborhood relation has a relative weight assigned, N_{inter} times the relative weight results in the number of synthetic instances that are created in each filtered cluster neighboring relation. In case there are no neighborhood relations for a class all the artificial samples are created through the intracluster process.

Step 11 - 11.2.2: In previous steps, we managed to identify the corresponding number of required synthetic instances for each filtered cluster formation and each neighboring relation formation. The following process is applied to each formation individually. The set of minority instances of the current formation is shuffled, minority instances are repetitively selected, each minority instance can be selected multiple times if the number of required synthetic instances is higher than the number of minority instances of the formation. The current selected minority sample of $N_{instances}$ is named x_{center}. Based on x_{center} the combined majority selection strategy is applied. The combined selection strategy identifies $x_{surface}$, the final neighbor of x_{center} that is used for over-sampling by applying the minority and majority selection strategy, both based on a k-nearest neighbor approach. The minority selection strategy selects a neighbor instance x_{min} within the k-nearest minority instances of x_{center}, similar to SMOTE. The majority selection strategy selects x_{maj}, a neighboring majority instance that has been identified with the identical approach applied to neighboring majority instances. $x_{surface}$ is defined as the instance x_{min} or x_{maj} that is closer to x_{center}, based on its distance. The distance between x_{center} and $x_{surface}$ is R. Figure 6a outlines the result of the majority and minority selection strategy, each strategy proposes one nearest neighbor of its class, the combined selection strategy in figure 6b selects the nearest sample of both selection strategies as $x_{surface}$.

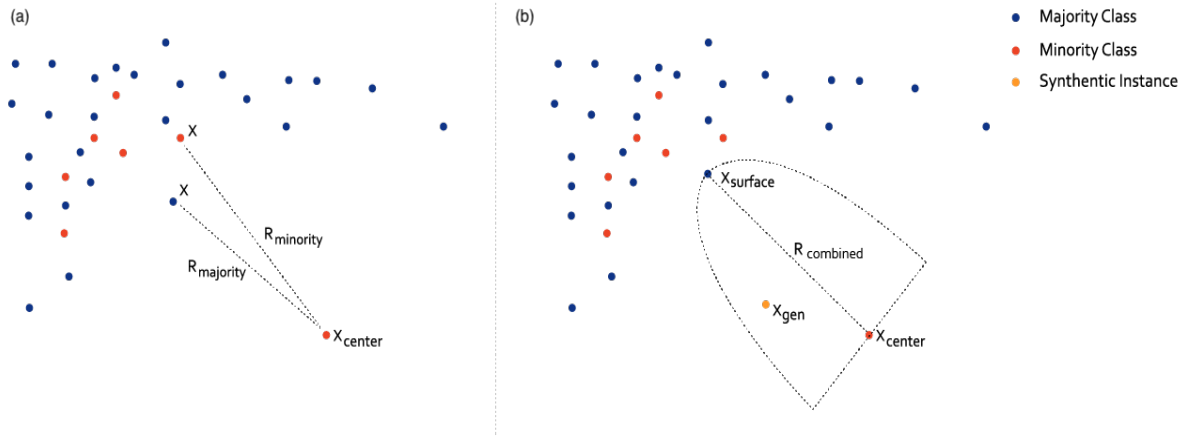

Figure 6: Procedure of the combined selection strategy, inspired by [Douzas and Bacao, 2017a]

Step 11.2.3 - End: To identify the most suitable geometric shape for the data generation process between x_{center} and $x_{surface}$, a unit hypersphere is established around x_{center}, as it can be seen in figure 7a . This hypersphere is going to be modified within the next steps. Initially, a random point e_{sphere} is created on the edge of the unit hypersphere, centered around x_{center}. Subsequent, e_{sphere} is transformed to x_{gen} a random point along the line segment, uniformly distributed. Step 11.2.4 applies a truncation, a partition of the hypersphere, as illustrated in 7b. The parameter α_{trunc} determines the degree of the truncation, as seen below. The truncated area is orthogonal to the unit vector $e_{//}$, where $e_{//}$ defines the direction between x_{center} and $x_{surface}$. If α_{trunc} is bigger than zero, the area that does not include x_{gen} is truncated. In case that x_{gen} is within the truncated area, the point is reflected on the opposite side of the hypersphere.

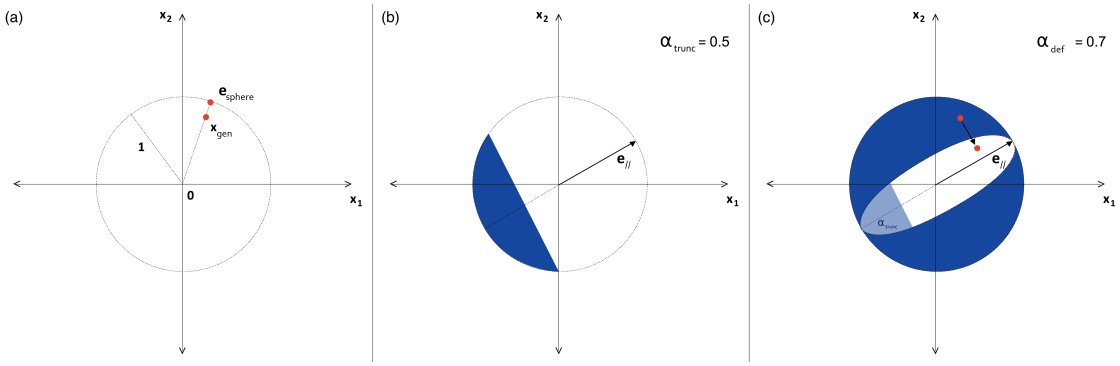

Figure 7: Constructing the geometric region, based on the unit hyper sphere, truncation and deformation, inspired by [Douzas and Bacao, 2017b]

Step 11.2.5 deforms the truncated hypersphere to a hyper spheroid, as it can be seen in 7c The parameter α_{def} controls the degree of the deformation. The point x_{gen} is moved to the same extent in the orthogonal direction to the unit vector $e_{//}$. The previous steps of truncation and deformation modify the initially uniform distribution of the new point x_{gen}. Due to the truncation and deformation, we take better account of the characteristics of x_{center} and $x_{surface}$, while still having a higher variability than methods based on line segments, such as SMOTE. In a final step we rescale the point x_{gen} based on x_{center} and the distance to $x_{surface}$, R. The rescaled outcome of the forming process can be seen below in figure 8. X_{gen} is created within a hypersphere, that was truncated and deformed to a hyper spheroid.

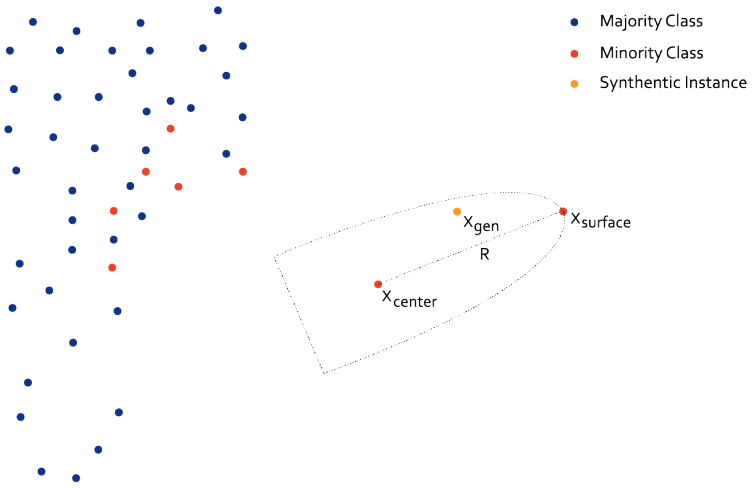

Figure 8: Rescaled result of geometric construction, inspired by [Douzas and Bacao, 2017b]

The process is repeated in each formation until $N_{instances}$ are added to each formation. This results in the total number of synthetic instances, the sum of N_{intra} and N_{inter}. The final output of the G-SOMO algorithm is the oversampled matrix X', consisting of the input data set and the added synthetic minority instances.

5 RESEARCH METHODOLOGY

The objective of a sampling method is the improvement of the classification results. This implies that validation methods have to be chosen to provide reliable and comprehensive results. The correct metrics have to be selected to not create misleading improvements as outlined in the next section. To receive comparable results, all oversampling methods need to be trained with the same classifier algorithms and are compared to the performance of the classifier algorithm on the same data set without oversampling.

Selecting a reliable validation technique is a common challenge to assess the generalizability of a classifier. Oversampling methods can tend to encourage the process of $overfitting$. Overfitting implies that the classifier learned the data structure too well and lost its ability to generalize on unseen data. A popular approach to validate the performance of a classifier is to split the data set into two subsets, one used for the training phase of the algorithm and the other one remains unseen during the training phase and is only used for validation, the test set. As a matter of fact, this validation method can even increase the process of overfitting, if the dataset is not split randomly. Certain characteristics of a feature might only occur in the test set and might not appear in the train set and vice versa. A more sophisticated method is k-fold Cross-Validation. The data set is split into k different subsets (also called folds). K-1 subsets are used to train the classifier and the last fold remains unseen to validate. The process is applied in an iterative manner, such that each fold is used for validation once. The validation results of each model are averaged afterwards. The process can be repeated multiple times to avoid bias due to random grouping [Japkowicz, 2013]. An extension to k-fold provides the stratified k-fold method. In stratified cross-validation, the random folds are chosen such that the class distribution in each fold is maximally similar to the class distribution in the whole dataset [Vanwinckelen and Blockeel, 2015]. The method successfully avoids the problem of an unequally split dataset, but faces problems if a minority class is highly underrepresented and does not have enough samples to split them equally in k folds.

5.1 METRICS

The model evaluation is based on different evaluation metrics. Traditional metrics, such as accuracy, show a strong bias towards the majority class and are recommended to avoid on imbalanced datasets. The overall accuracy would seem to be precise, even though the model might not perform well on the minority class. To retrieve more accurate insights on a models performance, one should reflect upon the confusion matrix, outlined below:

	Predicted as Positive	**Predicted as Negative**
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negatives (TN)

Based upon the confusion matrix additional metrics, such as Precision and Recall [Dalianis, 2018] were introduced:

$$Recall = \frac{TP}{TP+FN} \qquad Precision = \frac{TP}{TP+FP}$$

The evaluation of our experiments is conducted with the following metrics, that are based on the previously introduced fundamentals. These metrics proved to be reliable for imbalanced learning problems since they focus on both classes equally independent from their ratio.

- F1 Measure (F1)

  The F-measure is described as the harmonic mean between precision and recall, assuming that both metrics are of equal importance [Guo et al., 2018]. The F1 Score is defined as follows:

$$F1\ Measure = 2 * \frac{Precision*Recall}{Precision+Recall}$$

- Geometric Mean Score (G-Mean)

  The geometric mean score, as the name implies, is defined as the geometric mean between True Positive Rate (TPR) and True Negative Rate (TNR). The g-measure ranges between 0 and 1 and considers the TPR and TNR with equal importance. The metric is defined as follows:

$$G - Mean = \sqrt{\frac{TP}{TP+FN} * \frac{TN}{TN+FP}}$$

- Area Under The Curve Receiver Operating Characteristics (AUC - ROC)

  The ROC Curve is created by plotting the TPR against the FPR [Hand, 2009]. The AUC score is the area under that curve. The better the model distinguishes the majority and minority class the better the final score.

5.2 OVERSAMPLING METHODS

The following section provides an overview of the oversampling methods used as a benchmark to validate the effectiveness of G-SOMO. The baseline oversamplers are introduced in the second section of the paper and will therefore not be presented in detail. The oversamplers used for our benchmark results are popular techniques for binary classification tasks.

The optimal imbalance ratio to oversample is not known and might vary between datasets [Provost, 2018]. Usually, oversamplers are utilized to generate as much minority instances as required to equal the class distribution. Within our experimental framework, G-SOMO is the only oversampling method that uses an informative approach to identify effective areas to oversample. In case that no area is identified, the algorithm will not create any synthetic instances. The oversamplers used in our experimental framework are:

- No Oversampling
- Random Oversampling
- SMOTE
  - knn = {3,5}
- G-SOMO
  - knn = {3,5},
  - truncation_factor = {-1.0, 0.0, 0.25, 1.0},
  - deformation_factor = {0.0, 0.5, 1.0},
  - Grid_size = {0.2, 0.5}

While the parameters were already introduced during the explanation of the G-SOMO algorithm, it is worth to outline the Grid_size parameter. Hereby, we determine the number of clusters proportional by the number of input samples multiplied with the relative Grid_size parameter.

5.3 CLASSIFIERS

Different classifier algorithms are chosen to evaluate the performance of the oversampling methods. It is crucial to ensure that the obtained results are generalizable to different classifiers and not only to specific ones, due to the different algorithms characteristics. To reduce biased results only algorithms with a low number of hyperparameter are chosen. The chosen classifiers are Logistic Regression (LR), K-Nearest Neighbors (KNN), Decision Tree (DT) and Gradient Boosting Classifier (GBC).

The Logistic Regression is used to model the outcomes of a categorical dependent variable and predicts the probabilities for the different possible outcomes based on several independent variables. Fitting a linear model is an optimization problem which can be solved using simple optimizers which require no hyperparameters to be set [McCullagh, 1984]

KNN is a popular algorithm, in which a new instance is classified into the class with the most members present among the k nearest neighbors [Suguna and , 2010]. The hyperparameter k describes the number of neighbors considered for the classification.

Decision Trees create their classification decisions based on a tree structure that was obtained during the learning process. The final result is a tree with decision nodes and leaf nodes, where a decision node has two or more branches and leaf nodes represent the decision.

Gradient boosting is a technique that creates an ensemble of underlying weak learners to perform better than random guessing. By combining these weak learners based on a weighted majority vote, a committee classifier dramatically reduces the training and testing error rates [Huang et al., 2007]. The number of trees to create is a hyperparameter of the algorithm.

All classifiers are trained with different values of hyperparameters, besides the LR, which does not require any hyperparameter to be set. The following list provides an overview of the classifiers used with its hyperparameters:

- LR
- KNN (k = {3,5})
- DT (max_depth = {3,6})
- GBC (max_depth = {3,6})

5.4 DATASETS

To ensure meaningful and significant insights we evaluated our models on a total of 69 datasets. These datasets consist of commonly used datasets mainly from UCI Machine Learning repository. In order to reach this high number of datasets, we randomly undersampled the minority class of some datasets to increase the Imbalance Ratio. Using this approach, we could create additional and more challenging datasets. Table 1 provides an overview of our datasets, the number of features, instances and their Imbalance Ratio.

5.5 EXPERIMENTAL FRAMEWORK

Our experimental framework calculates the results of each oversampling method in combination with each classifier on each dataset based on each hyperparameter of the methods. The combination of all dependencies is named Grid Search, a search for the best results among all possible combinations. The result of each combination is obtained by the 5-fold cross-validation, repeated for 5 times, resulting in 25 models for each combination. Subsequently, the process is repeated for each metric.

A ranking score is applied to compare the performance of the oversampling methods, aggregated over all datasets. The ranking score for the best performing method is 1, the worst performing methods is 4. The Friedman test is applied to the ranking results. The Friedman test is used to test differences between groups, assuming our target variable is categoric. The null hypothesis used in our tests states whether the classifiers have a similar performance across the oversampling methods and evaluation metrics when they are compared to their mean ranking.

The implementation of the classifiers and oversamplers are based on the python libraries Scikit-Learn [Pedregosa et al., 2012] and Imbalanced-Learn [Lemaitre et al., 2016]. The G-SOMO extension is implemented in Python, the code is available upon request.

Data Sets					
Datasets	# Features	# Instances	# Minority Instances	# Majority Instances	Imbalance Ratio
BREAST TISSUE	9	106	36	70	1.94
BREAST TISSUE (2)	9	88	18	70	3.89
DERMATOLOGY	34	358	20	338	16.9
ECOLI	7	336	52	284	5.46
ECOLI (2)	7	310	26	284	10.92
ECOLI (3)	7	301	17	284	16.71
EUCALYPTUS	8	642	98	544	5.55
EUCALYPTUS (2)	8	593	49	544	11.1
EUCALYPTUS (3)	8	576	32	544	17.0
GLASS	9	214	70	144	2.06
GLASS (2)	9	179	35	144	4.11
GLASS (3)	9	167	23	144	6.26
HABERMAN	3	306	81	225	2.78
HABERMAN (2)	3	265	40	225	5.62
HABERMAN (3)	3	252	27	225	8.33
HEART	13	270	120	150	1.25
HEART (2)	13	210	60	150	2.5
HEART (3)	13	190	40	150	3.75
IRIS	4	150	50	100	2.0
IRIS (2)	4	125	25	100	4.0
IRIS (3)	4	116	16	100	6.25
LED	7	443	37	406	10.97
LED (2)	7	424	18	406	22.56
LIBRAS	90	360	24	336	14.0
LIVER	6	345	145	200	1.38
LIVER (2)	6	272	72	200	2.78
LIVER (3)	6	248	48	200	4.17
MANDELON 1	20	4000	142	3858	27.17
MANDELON 1 (2)	20	3929	71	3858	54.34
MANDELON 1 (3)	20	3905	47	3858	82.09
MANDELON 2	200	3000	105	2895	27.57
MANDELON 2 (2)	200	2947	52	2895	55.67
MANDELON 2 (3)	200	2930	35	2895	82.71
NEW THYROID 1	5	215	35	180	5.14
NEW THYROID 1 (2)	5	197	17	180	10.59
NEW THYROID 2	5	215	35	180	5.14
NEW THYROID 2 (2)	5	197	17	180	10.59
PAGE BLOCKS 0	10	5472	559	4913	8.79
PAGE BLOCKS 0 (2)	10	5192	279	4913	17.61
PAGE BLOCKS 0 (3)	10	5099	186	4913	26.41
PAGE BLOCKS 1 3	10	472	28	444	15.86
PIMA	8	769	268	501	1.87
PIMA (2)	8	635	134	501	3.74
PIMA (3)	8	590	89	501	5.63
SEGMENTATION	16	2310	330	1980	6.0
SEGMENTATION (2)	16	2145	165	1980	12.0
SEGMENTATION (3)	16	2090	110	1980	18.0
VEHICLE	18	846	199	647	3.25
VEHICLE (2)	18	746	99	647	6.54
VEHICLE (3)	18	713	66	647	9.8
VOWEL	13	988	90	898	9.98
VOWEL (2)	13	943	45	898	19.96
VOWEL (3)	13	928	30	898	29.93
WINE	13	178	71	107	1.51
WINE (2)	13	142	35	107	3.06
WINE (3)	13	130	23	107	4.65
YEAST 1	8	1484	429	1055	2.46
YEAST 1 (2)	8	1269	214	1055	4.93
YEAST 1 (3)	8	1198	143	1055	7.38
YEAST 3	8	1484	163	1321	8.1
YEAST 3 (2)	8	1402	81	1321	16.31
YEAST 3 (3)	8	1375	54	1321	24.46
YEAST 4	8	1484	51	1433	28.1
YEAST 4 (2)	8	1458	25	1433	57.32
YEAST 4 (3)	8	1450	17	1433	84.29
YEAST 5	8	1484	44	1440	32.73
YEAST 5 (2)	8	1462	22	1440	65.45
YEAST 6	8	1484	35	1449	41.4
YEAST 6 (2)	8	1466	17	1449	85.24

Table 1: Overview of all 69 datasets.

6 EXPERIMENTAL RESULTS

For the purpose of a clear distinguishing between the oversampling methods, we introduced a ranking score, ranging from one to four. More precisely, one represents the best performance, four the worst. In order to accumulate the retrieved insights, we averaged repetitions on each hyperparameter and on each dataset, as well as for each of the five repetitions during the cross-validation. The table below illustrates the mean rank of each oversampler on each metric and each classifier.

Experimental Results					
Classifier	Metrics	No Oversampling	Random Over-sampling	SMOTE	G-SOMO
LR	ROC AUC	3.014	2.731	2.536	1.717
LR	F1	3.347	2.942	2.355	1.355
LR	G-SCORE	3.695	2.275	2.021	2.007
KNN	ROC AUC	3.079	3.420	2.144	1.355
KNN	F1	3.398	2.753	2.5	1.347
KNN	G-SCORE	3.782	2.420	1.789	2.007
DT	ROC AUC	3.318	2.898	2.311	1.471
DT	F1	3.094	2.971	2.594	1.340
DT	G-SCORE	3.768	2.521	1.949	1.760
GBC	ROC AUC	3.007	3.275	2.565	1.152
GBC	F1	3.297	2.920	2.442	1.340
GBC	G-SCORE	3.586	2.731	2.050	1.630

Table 2: Overview of results, Mean Ranking for each Classifier and Metric.

Taking a closer look at the averaged mean scores in table 2, one can determine certain characteristics. As expected, not using any oversampling method provides the worst results among all classifiers and metrics. Random oversampling performs better on the mean ranking, while SMOTE is superior to both of them. G-SOMO consistently outperformed other oversampling methods. We may observe that G-SOMO provides exceptional good mean ranking scores in combination with the metrics 'ROC AUC' and 'F1'. In one case SMOTE achieved a better mean score while validation with a Decision Tree on the Geometric-Mean Score. In total we can see that G-SOMO consistently outperformed other oversampling methods, the results in combination with Gradient Boosting Classifier can be emphasized.
Based on the insights from table 2 we can conclude, that G-SOMO successfully outperforms other oversampling methods, proven in combination with various Metrics and Classifiers, averaged over all datasets. Figure 9 underlines the dominance of G-SOMO, the graph provides the aggregated results or each classifier, hereby we can see that throughout all experiments G-SOMO demonstrates the best performance. Moreover, it highlights the previously mentioned ranking between all oversampling methods.

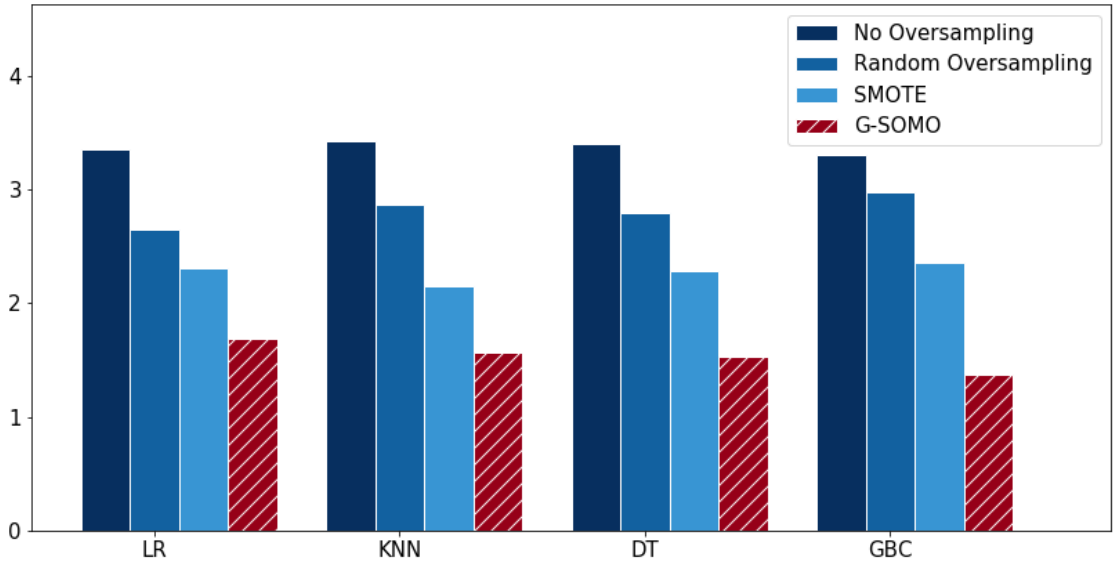

Figure 9: Graphical overview of Results, Mean Ranking for each Classifier

Besides the introduced mean ranking, we also analyze the relative differences in the metric scores in depth since the mean ranking might conceal high variation in specific repetitions. To avoid this scenario we observe the metrics scores overall results and aggregate them across all datasets and repetitions. To provide a direct

comparison between the performance of the oversampling methods, the results of G-SOMO are subtracted of each oversampling methods. The positive or negative difference implies whether the algorithm performed better or worse than the baseline G-SOMO. The differences are summarized in figure 10. The differences of the oversampling methods are partitioned across each classifier and each metric. The colored gradation illustrates the relative difference between the specific oversampling method and the performance of G-SOMO. The analysis of figure 10 reaffirms the previous insights, the majority of results are negative and therefore imply worse results than G-SOMO. The poor performance of no oversampling can be noted, the performance difference is up to 33%, random oversampling performs slightly better, SMOTE is the most challenging method. As previously recognized, G-SOMO performs exceptionally with the 'ROC AUC' and 'F1' measure. SMOTE manages to perform slightly better in the combination of the Geometric-Mean Score and KNN and LR classifier. Random Oversampling also performs better than G-SOMO in the combination of the Geometric Score and LR. Besides these exceptional cases, G-SOMO constantly outperforms all oversampling methods. The insights retrieved by the performance comparison is consistent with the insights obtained from the mean rankings.

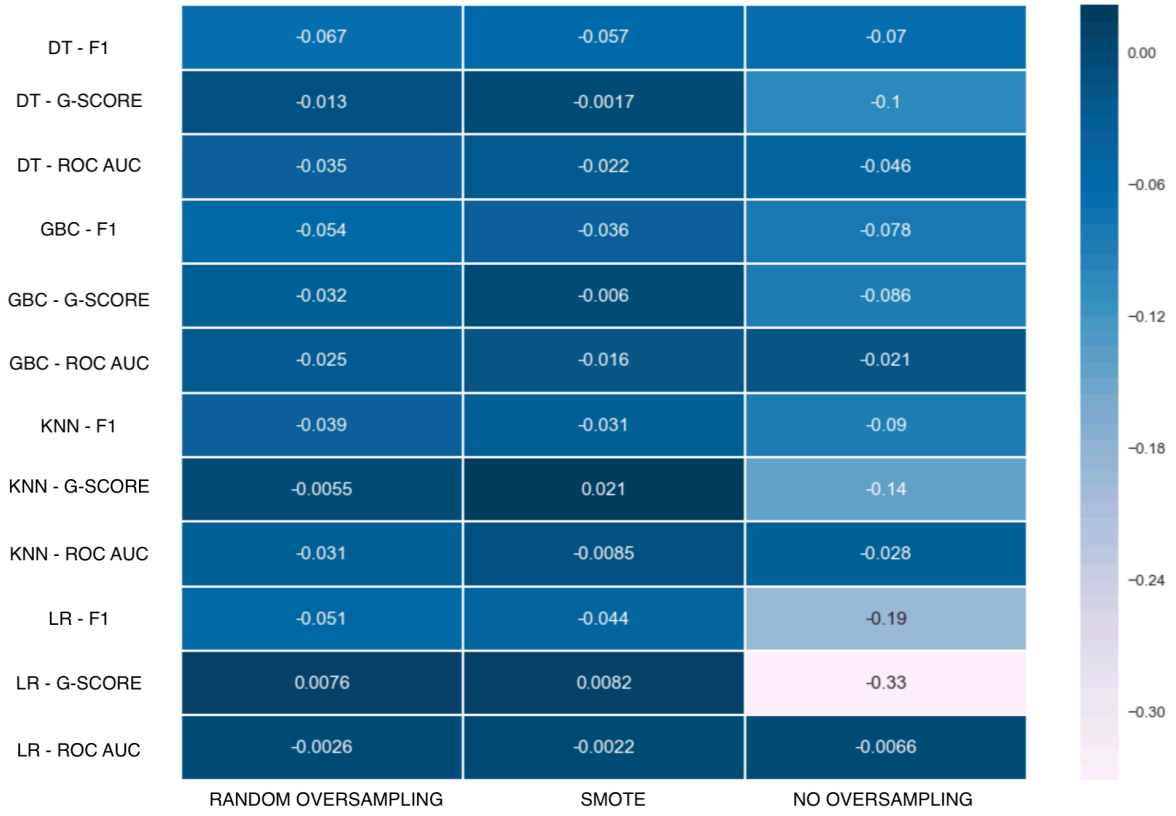

Figure 10: Average Score of Oversampling methods compared to G-SOMO performance

To prove the significance of our experiments, a Friedman test is applied to our results. The exact results are shown in table 3. Hereby, we can obtain that the null hypothesis is rejected by far for a significance level of a=0.05 for all classifiers and metrics. Therefore, we can assume that our obtained results are statistically significant.

Friedman Test			
Classifier	Metric	p-value	Significance
LR	ROC AUC	2.978464111147552e-09	True
LR	F1	1.6016121978014913e-21	True
LR	G-SCORE	1.1962255259807236e-18	True
KNN	ROC AUC	1.5777728619737928e-24	True
KNN	F1	1.2633506344258716e-20	True
KNN	G-SCORE	1.8594416543208717e-22	True
DT	ROC AUC	4.4812704879505726e-18	True
DT	F1	3.587895642878212e-18	True
DT	G-SCORE	4.37730518573295e-23	True
GBC	ROC AUC	3.6730748169610812e-25	True
GBC	F1	1.2458033635808544e-20	True
GBC	G-SCORE	5.957134339289162e-21	True

Table 3: Results of Friedman Test.

In general, we can observe that the informed geometric oversampling approach of G-SOMO has proven to be a successful new approach to handle imbalanced datasets. Our experiments prove the need for oversampling methods, using no oversampling methods consistently provided the worst results. We have found that Random Oversampling performs better than no oversampling, and SMOTE ranks second among our comparison in total. Based on our obtained insights, we determine that G-SOMO clearly outperformed other methods, often with great differences towards the other oversamplers. However, G-SOMO also requires a more intensive hyperparameter search, which in turn requires more computational resources.

7 CONCLUSION

In this paper, we proposed a new oversampling algorithm G-SOMO. The algorithm observes the characteristics of the multidimensional data input while grouping the input data to identify filtered clusters, where minority instances dominate. Within these filtered clusters, as well as between neighboring filtered clusters we create synthetic instances. During the creation of synthetic instances, we apply the combined selection strategy, that also takes near majority instances into account. The synthetic instances are created in a safe hyper-spheroid.

G-SOMO was evaluated on 69 different datasets and compared to popular oversampling methods. We validated the performance with different metrics, such as the F1 Score, the G-Score and the AUC-ROC. In order to avoid overfitting to a specific classifier, we chose multiple ones that differ in their characteristics. Each experiment is repeated 5 times with a 5-fold cross-validation. We proved the statistical significance of our results by a Friedman test.

In particular, our empirical results highlight the need for oversampling algorithms. Utilizing no oversampling method produced the worst results in our mean ranking, simple methods like random oversampling already increased the performance. The popular method SMOTE performed better than the previous ones, whereas G-SOMO dominated the ranking and outperformed all other oversamplers.

We are confident that G-SOMO is a new appealing approach for researcher and practitioners working with imbalanced datasets.

Future work will focus on a more efficient estimation of the algorithms hyperparameter. During our experiments we noticed the issue to be challenging, through additional research one might identify generalizable characteristics for the parameters based on the datasets characteristics. Additional improvements can be expected through further research. Alternatively, the behavior of the algorithm on imbalanced datasets with multiple target classes is worth to explore, little research about oversampling on multiclass classification problems is made.

8 REFERENCES

[Akbani et al., 2004] Akbani, R., Kwek, S., and Japkowicz, N. (2004). Applying support vector machines to imbalanced datasets. pages 39–50.

[Barua et al., 2014] Barua, S., Islam, M., Yao, X., and Murase, K. (2014). Mwmote–majority weighted minority oversampling technique for imbalanced data set learning. 26:405–425.

[Brocki and Korzinek, 2007] Brocki, L. and Korzinek, D. (2007). Kohonen self-organizing map for the traveling salesperson problem. pages 116–119.

[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. pages 475–482.

[Bunkhumpornpat et al., 2011] Bunkhumpornpat, C., Sinapiromsaran, K., and Lursinsap, C. (2011). Db-smote: Density-based synthetic minority over-sampling technique. 36.

[Chawla et al., 2003] Chawla, N., Lazarevic, A., O. Hall, L., and Bowyer, K. (2003). Smoteboost: Improving prediction of the minority class in boosting. 2838:107–119.

[Chawla et al., 2002] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357.

[Cieslak et al., 2006] Cieslak, D. A., Chawla, N. V., and Striegel, A. (2006). Combating imbalance in network intrusion datasets. pages 732–737.

[Dalianis, 2018] Dalianis, H. (2018). Evaluation metrics and evaluation. pages 45–53.

[Douzas and Bacao, 2017a] Douzas, G. and Bacao, F. (2017a). Geometric smote: Effective oversampling for imbalanced learning through a geometric extension of smote.

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

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

[Ganganwar, 2012] Ganganwar, V. (2012). An overview of classification algorithms for imbalanced datasets. 2:42–47.

[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. *SIGKDD Explor. Newsl.*, 6(1):30–39.

[Guo et al., 2018] Guo, H., Zhou, J., and Wu, C.-A. (2018). Imbalanced learning based on data-partition and smote. *Information (Switzerland)*, 9:238.

[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. 3644:878–887.

[Hand, 2009] Hand, D. (2009). Measuring classifier performance: A coherent alternative to the area under the roc curve. *Machine Learning*, 77:103–123.

[Hoens and Chawla, 2013] Hoens, T. R. and Chawla, N. V. (2013). Handling imbalanced datasets : A review. *Imbalanced Learning: Foundations, Algorithms, and Applications*, 1:43.

[Huang et al., 2007] Huang, J., Ertekin, S., Song, Y., Zha, H., and Giles, C. L. (2007). Efficient multiclass boosting classification with active learning.

[Japkowicz, 2013] Japkowicz, N. (2013). Assessment metrics for imbalanced learning. pages 187–206.

[Kubat and Matwin, 2000] Kubat, M. and Matwin, S. (2000). Addressing the curse of imbalanced training sets: One-sided selection.

[Köküer et al., 2007] Köküer, M., Naguib, R. N., Jancovic, P., Younghusband, H. B., and Green, R. (2007). Chapter 12 - towards automatic risk analysis for hereditary non-polyposis colorectal cancer based on pedigree data. pages 319 – 337.

[Last et al., 2017] Last, F., Douzas, G., and Bacao, F. (2017). Oversampling for imbalanced learning based on k-means and SMOTE. *CoRR*, abs/1711.00837.

[Lemaitre et al., 2016] Lemaitre, G., Nogueira, F., and Aridas, C. (2016). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. 18.

[Maria, 2004] Maria, G. E. A. P. A. B. R. C. P. (2004). A study of the behavior of several methods for balancing machine learning training data.

[McCullagh, 1984] McCullagh, P. (1984). Generalized linear models. 16:285–292.

[Nekooeimehr and K. Lai-Yuen, 2015] Nekooeimehr, I. and K. Lai-Yuen, S. (2015). Adaptive semi-unsupervised weighted oversampling (a-suwo) for imbalanced datasets. 46.

[Pedregosa et al., 2012] 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., Duchesnay, E., and Louppe, G. (2012). Scikit-learn: Machine learning in python. 12.

[Prati et al., 2004a] Prati, R., Batista, G., and Monard, M.-C. (2004a). Class imbalances versus class overlapping: An analysis of a learning system behavior. pages 312–321.

[Prati et al., 2004b] Prati, R. C., Batista, G. E. A. P. A., and Monard, M. C. (2004b). Learning with class skews and small disjuncts. pages 296–306.

[Provost, 2018] Provost, F. (2018). Machine learning from imbalanced data sets 101 (extended abstract).

[Rodriguez et al., 2012] Rodriguez, E., Snasel, V., Abraham, A., Wozniak, M., Grana, M., and Cho, S. (2012). *Hybrid Artificial Intelligent Systems: 7th International Conference, HAIS 2012, Salamanca, Spain, March 28-30th, 2012, Proceedings*. Number Teil 2 in Lecture Notes in Computer Science. Springer Berlin Heidelberg.

[Schubach et al., 2017] Schubach, M., Re, M., Robinson, P., and Valentini, G. (2017). Imbalance-aware machine learning for predicting rare and common disease-associated non-coding variants. *Scientific Reports*, 7.

[Suguna and , 2010] Suguna and , K. (2010). An improved k-nearest neighbor classification using genetic algorithm. 7.

[Sáez et al., 2016] Sáez, J. A., Krawczyk, B., and Wozniak, M. (2016). Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets. 57:164178.

[Tang and He, 2015] Tang, B. and He, H. (2015). Kerneladasyn: Kernel based adaptive synthetic data generation for imbalanced learning.

[Tang and Gao, 2007] Tang, Y. and Gao, J. (2007). Improved classification for problem involving overlapping patterns. 90-D:1787–1795.

[Vanwinckelen and Blockeel, 2015] Vanwinckelen, G. and Blockeel, H. (2015). Look before you leap: Some insights into learner evaluation with cross-validation. 47:3–20.

[Wan et al., 2014] Wan, X., Liu, J., Cheung, W. K.-W., and Tong, T. (2014). Learning to improve medical decision making from imbalanced data without a priori cost.

[Yen and Lee, 2006] Yen, S.-J. and Lee, Y.-S. (2006). Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset. 344:731–740.