SOM-XG: Self-Organizing Map Based Resampling with Sample Extraction and Generation.

Tharinda Dilshan Piyadasa, Kasun Gunawardana

Abstract— **The "Data Imbalance Problem" is a well-defined and challenging problem in the Machine Learning domain addressed throughout the past decades. With the emergence of Big Data, addressing the data imbalance has reemerged as a trending topic because traditional solutions for this problem are inadequate with the increasing volume and dimensionality of data. There exist a wide range of solutions, from data-level to algorithmic-level, proposed to address the data imbalance problem. Among these approaches, data-level approaches are popular among the scientific community because of their inherent classifier independence, making them generalizable over many different domains. Oversampling is one such data-level technique frequently explored by researchers, especially in extreme imbalance scenarios. This study introduces SOM-XG, an oversampling technique capable of addressing even the extreme imbalance scenarios. The proposed technique utilizes two Self-Organizing Maps and exploits their properties to address the within and between class imbalances and the decision boundary preservation, generating new synthetic samples that are topologically similar to the original samples in the dataset. The empirical results obtained for datasets with imbalance ratios ranging from 1.38 to 130, the number of features ranging from 3 to 300, and the number of samples ranging from 150 to 145,751, oversampled using SOM-XG, demonstrate enhanced classification results while consistently outperforming other state-of-the-art techniques.**

Keywords— **Data Imbalance Problem, Classification Analysis, Oversampling, Self-Organizing Map, Resampling**

I. Introduction

In the age of big data, data mining and knowledge discovery have become essential for making informed judgments. Classification analysis is a widely utilized data mining method applicable to numerous market and technical challenges like sentiment analysis, medical diagnosis, customer churn prediction, and recommendation systems. This involves instructing classifiers to distinguish between distinct problem-representing classes [1]. However, common state-of-the-art classifiers, when utilized for these tasks, only perform well with uniformly distributed data. Their focus is on accuracy enhancement without considering data dispersion [2]. However, in practice, data collected for classification analysis is frequently imbalanced in terms of class distribution.

The approaches used to overcome the data imbalance problem can be categorized into three groups.

Correspondence: Tharinda Dilshan Piyadasa (E-mail: tharindad7@gmail.com)
Received: 28.12.2022 Revised:13.09.2023 Accepted: 25-11-2023

Tharinda Dilshan Piyadasa and Kasun Gunawardana are from the University of Colombo School of Computing, Sri Lanka. (tharindad7@gmail.com, kgg@ucsc.cmb.ac.lk).

DOI: https://doi.org/10.4038/icter.v16i4.7270

1. External approaches (data level): attempt to balance the dataset by either removing the majority class samples (undersampling) or generating minority class samples (oversampling).
2. Internal approaches (algorithmic level): modify the underlying classification algorithm while keeping the original dataset intact through cost-sensitive learning or ensemble-based learning.
3. Hybrid approaches: combine internal and external approaches.

Among the three approaches used to address the data imbalance, external approaches are favored over internal approaches because they are more generalizable as a result of classifier independence [3]. The primary objective of any external approach is to balance the number of samples in the dataset classes, which is achieved through resampling. The imbalance that exists between the dataset classes due to the scarcity of data in one or more classes is called the between-class imbalance.

Oversampling and undersampling are two popular resampling methods used to balance imbalanced datasets by addressing the between-class imbalance. Oversampling poses a risk of overfitting by generating synthetic data resembling the original samples or incorrectly positioning them. On the other hand, undersampling may exclude crucial information and suffer from data scarcity for the minority class. To mitigate the fundamental drawbacks of these methods, numerous studies have been conducted over time. Their findings have led the advancement of resampling techniques, moving beyond basic methods like random selections and incorporating more sophisticated approaches along with statistical and probabilistic methods.

While oversampling carries the risk of overfitting, early detection during training is possible using simple approaches like a well-structured train-test split and monitoring changes in testing error compared to training error. Conversely, excluding essential information through undersampling can result in misclassifications when the model is tested on unseen data samples. Mohammed et al. [4] confirmed these observations by evaluating oversampled and undersampled datasets using advanced classifiers. Their conclusions indicate that, when compared to undersampling, oversampling typically yields more accurate classification. This serves as motivation for this study to focus on oversampling when formulating its strategy. This paper is a follow-up to our previous publication [5], focusing on addressing a number of critical gaps identified in that work.

The rest of this paper is organized as follows. Section II provides an overview of the related work, highlighting their strategies when performing oversampling. Section III presents the proposed algorithm with an in-depth explanation of its steps. Section IV presents our evaluation framework and the experimental setup. Section V provides the results obtained, followed by discussion and conclusion sections .

II. RELATED WORK

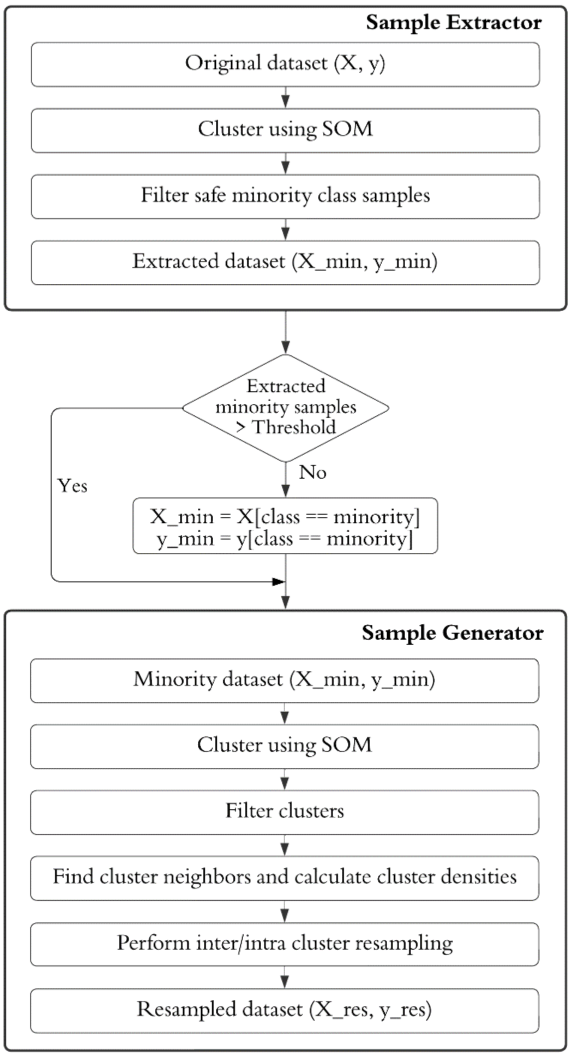
When investigating studies on oversampling, SMOTE [6] and its variants can be identified as the most extensively utilized strategies in academia. In SMOTE (Synthetic Minority Oversampling Technique), a minority class sample is randomly selected to generate a new synthetic sample that lies along the line segment that joins the selected sample and one of its K nearest neighbors. As samples are not duplicates of already existing data, SMOTE has a lower risk of overfitting compared to random oversampling. However, due to random selection, the minority regions that are densely populated get further concentrated, and regions that are sparsely populated remain sparse. SMOTE is also sensitive to noise as a sample generated by combining an existing noisy sample with its nearest neighbor is more likely to be noisy.

[7]-[10] are some of the variants of SMOTE introduced to address its limitations. When exploring these techniques, it is evident that they try to control the areas of sample generation and remove noise by enhancing the output of SMOTE. The techniques focus on preserving the decision boundary that separates the classes, which is a major limitation of vanilla SMOTE as it frequently distorts class separation, generating samples that overlap with the majority class. The evaluation results obtained by these techniques emphasize the importance of decision boundary preservation when formulating a new oversampling technique.

When considering real-world datasets, there exist situations where the minority class is scattered into multiple dense or sparse disjuncts. This is called the within-class imbalance, and it can result in the lack of representation of important characteristics in the minority class, distorting its distribution. Cluster SMOTE [11], DBSMOTE [12], MWMOTE [13], and K-means SMOTE [14] are popular oversampling techniques based on clustering and density estimation. They locate areas within the data space where oversampling is most effective, making it possible to address the within-class imbalance by manipulating the spatial location of the sample distribution.

Another common observation that can be made regarding many available oversampling techniques is that they operate in the Euclidean space. However, as real-world data are often high-dimensional, the heuristics based on Euclidean distance become meaningless when operating on them. This effect is referred to as the “curse of dimensionality” [15], which can be alleviated using dimensionality reduction. Among resampling techniques that incorporate dimensionality reduction, Self-Organizing Maps (SOM) [16] based resampling techniques have been actively investigated in the scientific community in recent years.

Douzas and Bacao [17], Vannucci and Colla [1], and Zhang et al [18] use SOMs to generate a 2D representation of the input state before performing oversampling. Douzas and Bacao [17] propose Self-Organizing Map based oversampling (SOMO) that uses the density of minority class samples to filter out clusters generated by the SOM algorithm. Synthetic samples are generated within these identified clusters and between neighboring clusters using SMOTE, addressing the within and between class imbalances. [1] presents a hybrid resampling technique combining SOM and genetic algorithms, where two SOMs are used for oversampling and undersampling, respectively. Zhang et al [18] address the clutter suppression in search radars and introduce a SOM-SMOTE oversampling technique to alleviate the imbalance in the clutter dataset. Based on these studies, it can be assumed that the ability of SOMs to address the within-class imbalance as a clustering algorithm and reduce the dimensionality maintaining the input topology are the main reasons for its popularity in this context.


Sample Extractor
Original dataset (X, y)
Cluster using SOM
Filter safe minority class samples
Extracted dataset (X_min, y_min)
Extracted minority samples > Threshold
Yes
No
X_min = X[class == minority]
y_min = y[class == minority]
Sample Generator
Minority dataset (X_min, y_min)
Cluster using SOM
Filter clusters
Find cluster neighbors and calculate cluster densities
Perform inter/intra cluster resampling
Resampled dataset (X_res, y_res)
Fig. 1 - High-level overview of SOM-XG.

In summary, when analyzing oversampling techniques that address the data imbalance problem, it is possible to identify several key constraints that contribute to their success: (1) Addressing the between-class imbalance, (2) Addressing the within-class imbalance, and (3) Preserving the boundary during sample generation. Furthermore, aside from addressing the above constraints, it is also preferable to pay special attention to the curse of dimensionality. The proposed oversampling technique utilizes SOMs due to its ability to address both within-class and between-class imbalances from the perspective of a clustering algorithm and imbalance in higher-dimensional data from the perspective of a topology-preserving dimensionality reduction algorithm.

Our initial work [5] contains a more detailed description of the related research and how the aforementioned constraints were derived based on the strategies introduced by the authors. Therefore, we recommend referring to it for a comprehensive understanding of the literature, its limitations, and potential research directions.

III. METHODOLOGY

The proposed oversampling technique consists of two phases: Sample extraction, and Sample generation. During the sample extraction phase, minority class samples are extracted from the original dataset to create a new dataset that contains safe minority class samples suitable for oversampling. These minority class samples are then used to create new synthetic samples in the sample generation phase. Figure 1 gives a high-level overview of the SOM-XG oversampling technique.

A. *Sample Extractor*

The Sample Extractor is used to extract safe minority class samples suitable for generating new synthetic samples. In this context, safe minority class samples refer to the minority samples that reside in areas with high minority class density and are not noisy. Due to the topology-preserving property and the clustering nature of the SOM, it is possible to identify the occurrences of multiple disjuncts of minority class samples with varying densities more accurately if they exist. Algorithm 1 presents the pseudo-code of the sample extractor.

Algorithm 1 Sample Extractor Pseudo Code

```
Input: X, y, num_rows, num_cols, num_iterations,
learning_rate, sigma
Output: $X_{min}$, $y_{min}$
  1. Normalize the input data
  2. Cluster the normalized data using SOM
  3. Filter safe minority class samples
       3.1. Filter safe non-boundary minority class samples
       3.2. Filter safe boundary minority class samples
  4. Aggregate the filtered minority class samples to
     form a new dataset
  5. return $X_{min}$, $y_{min}$
```

Inputs: The Sample Extractor expects seven input parameters, five of which are SOM parameters. ***X*** and ***y*** represent the data samples and their corresponding labels. The SOM parameters are ***num_rows*** representing the number of rows in the grid, ***num_cols*** representing the number of columns in the grid, ***num_iterations*** representing the number of training iterations, ***learning_rate*** representing the initial learning rate, and ***sigma*** representing the spread of the neighborhood function. The grid of the map is fixed to be ‘hexagonal’, and the neighborhood function is fixed to be ‘Gaussian,’.

Normalize the input data: The input data is normalized before training the SOM such that each feature lies between [0, 1].

Cluster the normalized data using SOM: In this step, the normalized data is clustered using the SOM. Each neuron in the SOM grid is considered a separate cluster. Therefore, number of clusters = *num_cols* × *num_rows*.

Filter safe minority class samples: The safe minority class samples are divided into two groups depending on where they are placed on the map.

1. Samples that reside in areas with high minority class density
2. Samples that are not noise but reside in areas with low minority class density

During the initial filtering process, the intuition is that minority class samples residing in areas with high minority class density (the number of minority samples is greater than the number of majority samples) are naturally most suitable for generating new synthetic samples. Therefore, as depicted in Figure 2 (A), clusters with high minority class density are filtered out, and minority samples that reside in the filtered clusters are extracted as safe minority class samples.

The remaining minority class samples after the initial filtering include minority class samples that reside in areas with high majority class density, minority class samples in the boundary regions, and minority class samples that are noise. In order to filter safe minority class samples out of these, a Nearest Neighbors approach is used. For each minority class sample in the remaining clusters, the three nearest neighbors are considered. If at least two of the three nearest neighbors are minority samples, the minority class sample is considered safe and filtered out.

For example, in Figure 2 (B), minority samples p and q represent a sample in a cluster with a high majority class density, and a noisy sample, respectively. When considering the three nearest neighbors of p, it can be seen that two samples are of the minority class, and one sample is of the majority class. Therefore, p is considered a safe minority class sample and extracted. However, when considering q, all three neighbors are majority class samples. Therefore, it is not considered a safe sample and hence, not extracted.

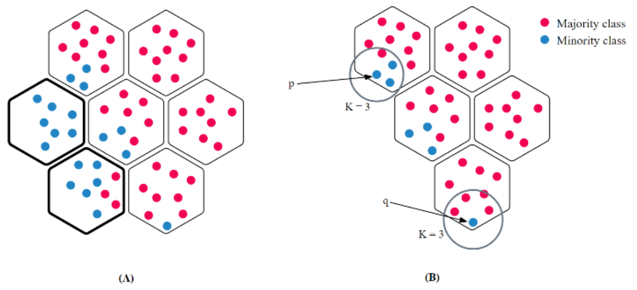


Fig. 2 - Sample Extractor. (A) Highlighted nodes are the clusters with high minority class density. (B) The remaining nodes after the clusters with high minority class samples are filtered out.

Aggregate the filtered minority class samples to form a new dataset: The filtered minority class samples are combined to form a new dataset that can be used as input to the Sample Generator.

B. *Sample Generator*

The Sample Generator uses the minority class samples extracted from the Sample Extractor and generates new synthetic samples between them using SMOTE. Similar to the extraction phase, the generation phase also utilizes a SOM when generating new synthetic samples so that the synthetically generated samples are topologically similar to minority samples in the original dataset. The pseudo-code of the Sample Generator is demonstrated in Algorithm 2.

An important design decision of the proposed algorithm is using a threshold (lower limit) to determine the number of samples required to be extracted from the Sample Extractor to use in the Sample Generator. If the number of extracted samples is less than the threshold, the entire minority class is used as input to the Sample Generator, ignoring the output of the Sample Extractor. Setting such a threshold implicitly declares the minimum number of samples required for oversampling, but there is no prior research conducted regarding this. After performing multiple experiments with the selected datasets, 25 was selected as the threshold across all datasets in the presented study.

Algorithm 2 Sample Generator Pseudo Code

```
Input: $X_{min}$, $y_{min}$, num_rows, num_cols, num_iterations, learning_rate, sigma, resample_ratio
Output: $X_{res}$, $y_{res}$
    1. Normalize the minority class samples
    2. Cluster the normalized minority class samples using SOM
    3. Filter clusters with rich minority class density
    4. Find neighboring clusters and calculate inter-cluster and intra-cluster densities
    5. Perform resampling:
        5.1. Perform intra-cluster oversampling
        5.2. Perform inter-cluster oversampling
    6. Aggregate the filtered minority class samples to form a new dataset
    7. return $X_{res}$, $y_{res}$
```

Inputs: The Sample Generator expects eight inputs, five of which are SOM parameters as same as the parameters of the Sample Extractor. Among other parameters, ***X_{min}*** and ***y_{min}*** represent the minority class samples and their corresponding labels, and ***resample_ratio*** represents the ratio of intra-cluster and inter-cluster samples to generate. The *resample_ratio* takes a value between 0 and 1, dividing the total number of samples to generate between intra-cluster and inter-cluster samples. For example, if the *resample_ratio* is 0.2, 20% intra-cluster samples and 80% inter-cluster samples are generated, such that the number of minority class and majority class samples are equal. This metric adds extra variation to the synthetic sample generation process to facilitate datasets with different properties.

Normalize the minority class samples: Features are normalized such that they lie between [0, 1].

Cluster the normalized minority class samples using SOM: In this step, the normalized minority samples are clustered using the second SOM, where each neuron is considered a separate cluster.

Filter clusters with rich minority class density: In the clusters generated after training the SOM, there can be clusters with no minority class samples or very few minority class samples. These clusters carry no useful information and are considered to be unrepresentative. Therefore, clusters with rich minority class density are filtered out using a threshold, ***filter_ratio***, which is calculated using the number of minority class samples ($|X_{min}|$) and the number of neurons in the SOM grid (*num_rows* × *num_cols*) as shown in Equation 1.

$$filter\_ratio = \frac{|X_{min}|}{num\_rows \times num\_cols} \tag{1}$$

The above equation calculates the number of minority class samples that belong to each neuron (winner neuron) if the samples are equally divided.

Find neighboring clusters and calculate inter and intra-cluster densities: From the filtered clusters, the neighboring clusters are identified based on the Euclidean coordinates of the clusters (neurons). Since the topology of the map is 'hexagonal,' each cluster can have up to a maximum of six neighboring clusters. These neighboring clusters are processed in pairs as the inter-cluster sample generation is performed between two neighboring clusters.

The ***intra-cluster density*** is calculated similarly to the approach followed by [17] in the SOMO algorithm based on the number of samples in the cluster ($|C_i|$) and the average Euclidean distance between all the samples belonging to the cluster. The intra-cluster density of a cluster *i* can be represented as shown in Equation 2.

$$d_{intra}(i) = \frac{|C_i|}{\left[\frac{1}{|C_i|} \times \sum_{p,q\,\in\,C_i}\left(\sqrt{\sum_{j=1}^{n}(p_j - q_j)^2}\right)\right]^2} \tag{2}$$

Where, $|C_i|$ = size of the cluster *i* (number of samples), *n* = number of features in the dataset, and *p* and *q* are two samples in cluster C_i.

When calculating the ***inter-cluster density*** between two neighboring clusters, the individual intra-cluster densities of the two clusters are simply added together. Equation 3 demonstrates how the inter-cluster density between clusters *i* and *j* are calculated.

$$d_{inter}(i,j) = d_{intra}(i) + d_{intra}(j) \tag{3}$$

Perform resampling: The resampling of the minority class is performed in two phases: intra-cluster oversampling, where new samples are generated within each filtered cluster, and inter-cluster oversampling, where new samples are generated between neighboring clusters if present.

***Intra-cluster oversampling*:** Intra-cluster oversampling refers to the sample generation in each filtered cluster by taking the density of the cluster into account. Each filtered cluster is assigned a weight in such a way that the weight is directly proportional to the density of the cluster. i.e., clusters with higher density will be assigned a higher weight. More samples are generated in clusters with higher weights compared to clusters with lower weights as new samples can be generated with more confidence in areas where the minority class density is high. The weight (W) assignment rule of a cluster *i* can be expressed as shown in Equation 4, Where *n = num_rows × num_cols.*

$$W_i = \frac{d_{intra}(i)}{\sum_{j=1}^{n} d_{intra}(j)} \tag{4}$$

The number of samples (S) to generate in a cluster is determined based on the weight assigned to the cluster. The number of samples generated in cluster i can be calculated using Equations 5 and 6.

$$S_i = int(W_i \times T) \tag{5}$$

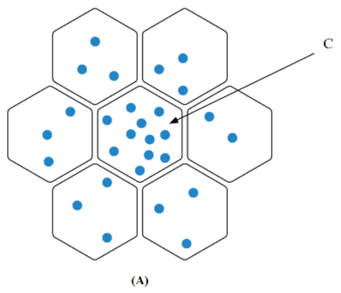
Where

$$T = \begin{cases} |X_{maj}|, & if\ neighbors = \emptyset \\ |X_{maj}| \times resample\_ratio, & otherwise \end{cases} \tag{6}$$

As the total number of samples that need to be generated (T) is divided among inter-cluster and intra-cluster oversampling, *resample_ratio* is set as a hyperparameter and tuned during experiments.

Finally, the required number of minority samples for each cluster is generated by applying SMOTE. Depending on the number of minority class samples in the cluster, the *k_neighbors* parameter of SMOTE is changed between 1, 3, and 5.

The intra-cluster sample generation process is illustrated in Figure 3. It can be observed that there are 28 minority class samples present with 7 neurons in Fig. 3 (A). Based on the *filter_ratio* calculation, a cluster should have at least 4 minority class samples to become a candidate for intra-cluster oversampling. As a result, only cluster *C* is filtered out of the initial set of clusters. Fig. 3 (B) represents how SMOTE is applied to generate new synthetic samples in a cluster. Since the number of samples in cluster *C* is greater than 5, SMOTE is applied with the *k_neighbors* parameter set to 5.



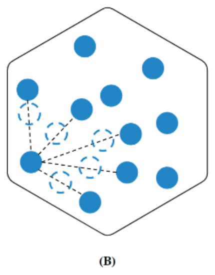


Fig. 3 - Intra-cluster sample generation.

Inter-cluster oversampling: In inter-cluster oversampling, minority samples are generated between pairs of neighboring clusters. Similar to the weight assignment step in intra-cluster oversampling, each pair of neighboring clusters is assigned a weight based on the density between them, and synthetic samples are generated so that more samples are generated between neighboring pairs with higher weights. Both weight and sample calculations can be expressed with Equations 7 and 8.

$$W_{i,j} = \frac{d_{inter}(i,j)}{\sum_{j=1}^{n} d_{intra}(j)} \tag{7}$$

$$S_{i,j} = int(W_{i,j} \times [|X_{maj}| \times (1 - resample\_ratio)]) \tag{8}$$

During sample generation between two neighboring clusters, a random sample is selected from each cluster, and a new synthetic sample is generated between the selected samples by applying SMOTE. This action is graphically represented in Fig. 4.

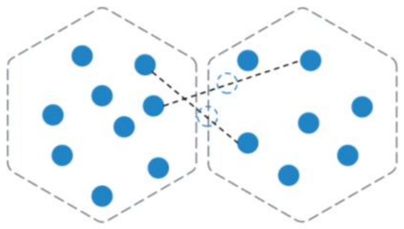


Fig. 4 - Inter-cluster sample generation.

The conclusion of the sample generation phase results in a new oversampled dataset with a balanced distribution of majority and minority classes.

IV. Evaluation Framework

A. Metrics

As it is required consider the independent accuracy of both the majority and minority classes when evaluating imbalanced data, this study uses three imbalanced learning metrics to evaluate the performance of classifiers on resampled datasets: (1) F-Measure, (2) Area under the Receiver Operating Characteristics curve (AUC-ROC), and (3) Matthews Correlation Coefficient (MCC) [19]. Among the three metrics, the AUC-ROC score is used as the primary metric.

B. Datasets

Ten publicly available datasets from UCI [20], KDD Cup [21], and LIBSVM [22] repositories are selected to conduct the experiments and evaluations. As a preprocessing step, all multivariate classification datasets are converted into binary classification by identifying a single minority class and combining the rest of the classes to form the majority class. An overview of the selected datasets, with the selected minority class (Target), repository, number of features, number of samples, and the imbalanced ratio (IR), are presented in Table I.

C. Oversampling Methods and Classifiers

The performance of the formulated oversampling technique is compared and evaluated against Random Oversampling, SMOTE, K-means SMOTE, and SOMO. Among the above techniques, SOMO is considered the primary oversampling technique that SOM-XG is compared against, as both techniques are based on SOMs. The number of K nearest neighbors of SMOTE, K-means SMOTE, and SOMO are tuned with *k_neighbors* ∈ {2, 3, 4, 5}, and the *distribution_ratio* of the SOMO, which determines the ratio of intracluster/intercluster points generated, is set in the range [0.1, 0.9] with an interval of 0.1. For SOM-XG, apart from tuning the parameters of the SOMs, the *resample_ratio* is tuned in the range [0.1, 0.9] with an interval of 0.1.

The primary advantage of any data level resampling technique is the inherent classifier independence. Therefore, after oversampling using the aforementioned techniques, the balanced datasets are classified using three different classifier

TABLE I
DATASETS

Id	Dataset	Repository and Target	IR	Samples	Features
1	liver	UCI, target: 1	1.38	345	6
2	iris	UCI, target: 2	2.00	150	4
3	haberman	UCI, target: 2	2.78	306	3
4	libras_move	UCI, target: 1, 2, 3	4.00	360	90
5	ecoli	UCI, target: pp	5.46	336	7
6	web_page	LIBSVM, target: minority	33.00	34,780	300
7	ozone_level	UCI, target: ozone	34.00	2,536	72
8	Mammography	UCI, target: minority	42.00	11,183	6
9	protein_homo	KDD CUP, target: minority	111.00	145,751	74
10	abalone_19	UCI, target: 19	130.00	4,177	10

models: Logistic Regression Classifier (LRC), Gradient Boosting Classifier (GBC), and Decision Tree Classifier (DTC). Among the selected classifiers, LRC does not require any hyperparameters to be tuned. For GBC, *max_depth* ∈ {3, 6, 9} and *n_estimators* ∈ {50, 100, 200}. In DTC, *max_depth* ∈ {3, 6, 9}.

D. Experimental Setup

The experiments are conducted in two stages due to computational limitations. Datasets 1 - 5 with smaller imbalanced ratios and dimensions (***low imbalanced datasets***), and datasets 6 - 10 with higher imbalanced ratios and dimensions (***high imbalanced datasets***) are evaluated separately in stages one and two, respectively. During low imbalanced dataset evaluation, a one-on-one comparison is conducted between SOM-XG and SOMO, and during high imbalanced dataset evaluation, SOM-XG is compared against other state-of-the-art oversampling techniques to determine its generalizability to higher-dimensional datasets

However, in the sample generator, as the SOM is trained only on minority data, the lower and the upper bounds are set to $\sqrt{X_{min} - 4}$ and $\sqrt{X_{min} + 4}$, respectively, where X_{min} represents the minority class samples extracted through the sample extractor.

In the first evaluation stage, performance results are obtained for each dataset using stratified K-fold cross validation with K = 5. However, for the second stage of evaluation, results are derived using a stratified train test split due to the magnitude of data and the limitations in computational resources. The classifier parameters are tuned in each fold using the original data from the K-1 folds, forming a separate 5-fold cross validation, maximizing the AUC-ROC score. Finally, the tuned classifiers are trained on oversampled data and evaluated on the withheld fold of test data. The experimental procedure is carried out for all possible hyperparameters, maximizing the AUC-ROC score.

E. Tuning the parameters of Self-Organizing Maps

The SOM implementation used in the algorithm is based on MiniSOM [23], and the parameters of the sample extractor and generator are tuned identically except for the grid size. The SOM is considered to have an equal number of rows and columns. The lower bound of this value is set to $\sqrt{|X\_minority|}$ such that each neuron contains at least one minority class sample, and the upper bound is set to $\sqrt{|X\_majority|}$. These bounds are similar to the SOM training conducted in SOMO [17].

The number of iterations is set with a lower bound of $|X|$ and an upper bound of $(|X| \times 10)$, where X is the training dataset. For the initial learning rate, the lower bound is set to 0.001 and the upper bound is set to 1. Sigma determines the spread of the neighborhood function, and its lower bound of is set to 1 and the upper bound is set to $\frac{max(num\_rows, num\_cols)}{2.01}$. Dividing by 2.01 ensures that sigma is strictly less than half the dimensions of the map. For all three parameters, ten evenly placed values in the give interval are considered during tuning.

The sample extractor is tuned with the above parameters to reduce the Topographic Error, which assesses how well the SOM preserves the topology in a given dataset by determining the number of data samples not having the first and the second best matching units (BMU) adjacent to each other. Whereas the sample generator is tuned to maximize the AUC-ROC value.

V. RESULTS

In the given tables, AUC represents the area under the ROC curve, F1 represents the weighted F1 score, and MCC represents the Matthews Correlation Coefficient. The bold figures indicate the best value of each metric.

A. Low Imbalanced Dataset Evaluation

SOM-XG is evaluated on the low imbalanced datasets and compared with SOMO [17], which has outperformed other oversampling techniques during its evaluation. Table II present the results obtained by the two techniques when classified using LRC, GBC, and DTC.

B. High Imbalanced Dataset Evaluation

In order to determine the generalizability and the adaptability to varying imbalance ratios, SOM-XG is evaluated on high imbalanced datasets and compared against existing state-of-the-art oversampling techniques. Tables III, IV, and V demonstrate the evaluation results.

Based on the evaluation results, especially with respect to the target metric AUC-ROC, it can be observed that SOM-XG performs well on both low and high imbalanced datasets. It can also be seen that SOM-XG consistently outperforms all the other techniques when evaluated on the most imbalanced dataset (abalone_19), and datasets with the highest number of features (webpage), and samples (protein). This demonstrates the generalizability of SOM-XG to high dimensional and high imbalanced datasets.

TABLE II
RESULTS OF LOW IMBALANCED DATASET EVALUATION

Oversampler	Metric	Dataset				
		liver	iris	haberman	libras	ecoli
			LRC			
SOMO	AUC	0.669	**0.750**	0.640	0.611	0.897
	F1	0.666	**0.664**	0.744	**0.739**	**0.901**
	MCC	0.335	**0.502**	0.324	0.214	0.684
SOM-XG	AUC	**0.676**	**0.750**	**0.729**	**0.629**	**0.910**
	F1	**0.677**	0.600	**0.745**	0.595	0.900
	MCC	**0.393**	0.500	**0.437**	**0.239**	**0.714**
			GBC			
SOMO	AUC	0.720	**0.960**	0.644	0.776	0.911
	F1	**0.727**	**0.961**	**0.733**	0.861	0.935
	MCC	0.442	**0.914**	0.333	0.584	0.803
SOM-XG	AUC	**0.722**	**0.960**	**0.653**	**0.831**	**0.935**
	F1	**0.727**	**0.961**	0.720	**0.896**	**0.957**
	MCC	**0.446**	**0.914**	**0.344**	**0.690**	**0.844**
			DTC			
SOMO	AUC	0.689	0.955	0.627	**0.851**	0.863
	F1	0.686	0.960	0.723	**0.910**	0.897
	MCC	0.393	0.911	0.279	**0.722**	0.648
SOM-XG	AUC	**0.712**	**0.965**	**0.668**	0.840	**0.923**
	F1	**0.706**	**0.967**	**0.724**	0.880	**0.951**
	MCC	**0.420**	**0.930**	**0.351**	0.661	**0.828**

TABLE III
RESULTS FOR LRC-CLASSIFIED OVERSAMPLING METHODS ON HIGH IMBALANCED DATASETS

Oversampler	Metric	Dataset				
		webpage	ozone	mammography	protein	abalone_19
Random	AUC	0.920	0.617	0.853	0.919	0.861
	F1	0.964	0.724	0.896	0.969	0.832
	MCC	0.561	0.080	0.280	0.347	0.142
SMOTE	AUC	**0.923**	0.619	**0.855**	0.921	0.866
	F1	0.964	0.727	0.898	0.970	0.839
	MCC	0.567	0.081	0.285	0.355	0.146
K-means SMOTE	AUC	0.865	0.621	0.852	0.844	0.677
	F1	**0.972**	**0.730**	0.938	**0.981**	0.851
	MCC	**0.578**	0.082	0.351	**0.382**	0.073
SOM-XG	AUC	**0.923**	**0.691**	0.842	**0.926**	**0.885**
	F1	0.966	0.701	**0.949**	0.966	**0.863**
	MCC	0.575	**0.129**	**0.378**	0.336	**0.161**

TABLE IV
RESULTS FOR GBC-CLASSIFIED OVERSAMPLING METHODS ON HIGH IMBALANCED DATASETS

Oversampler	Metric	Dataset				
		webpage	ozone	mammography	protein	abalone_19
Random	AUC	0.889	0.580	0.795	0.912	0.498
	F1	0.973	0.954	0.977	**0.997**	0.986
	MCC	0.605	0.168	0.532	**0.832**	-0.006
SMOTE	AUC	0.888	0.671	**0.864**	0.935	0.638
	F1	0.977	**0.962**	0.964	0.995	0.978
	MCC	0.641	0.226	0.472	0.753	0.153
K-means SMOTE	AUC	0.853	0.543	0.807	0.897	0.586
	F1	**0.982**	0.960	0.974	0.996	0.975
	MCC	0.678	0.162	0.503	0.776	0.090
SOM-XG	AUC	**0.890**	**0.712**	0.770	**0.936**	**0.696**
	F1	**0.982**	0.959	**0.981**	0.995	**0.988**
	MCC	**0.691**	**0.355**	**0.567**	0.752	**0.320**

TABLE V
RESULTS FOR DTC-CLASSIFIED OVERSAMPLING METHODS ON HIGH IMBALANCED DATASETS

Oversampler	Metric	Dataset				
		webpage	ozone	mammography	protein	abalone_19
Random	AUC	0.795	0.678	0.835	0.929	0.776
	F1	0.946	0.893	0.885	0.979	0.851
	MCC	0.363	0.166	0.256	0.425	0.113
SMOTE	AUC	0.793	0.705	**0.850**	0.932	0.775
	F1	0.953	0.899	0.886	0.978	0.850
	MCC	0.390	0.194	0.268	0.419	0.112
K-means SMOTE	AUC	0.775	0.668	0.788	0.878	0.602
	F1	**0.958**	**0.907**	0.934	**0.989**	**0.882**
	MCC	0.402	0.168	0.289	**0.532**	0.046
SOM-XG	AUC	**0.819**	**0.777**	0.809	**0.940**	**0.893**
	F1	0.956	**0.907**	**0.947**	0.979	0.874
	MCC	**0.431**	**0.265**	**0.344**	0.438	**0.169**

VI. DISCUSSION

The algorithm proposed in this study is based on the three constraints discovered during our initial review of oversampling techniques [5]. The three constraints, addressing the within-class imbalance, between-class imbalance, and preserving the decision boundary when generating new synthetic samples, define the conditions that should be satisfied to achieve high classification performance when formulating an oversampling technique.

In this study, the between-class imbalance is addressed by generating minority class samples such that the number of minority and majority class samples is equal in the dataset. Irrespective of the technique used, addressing the between-class imbalance is the fundamental motivation of data level approaches. When addressing the within-class imbalance, it is evident from previous studies that clustering-based techniques are more prominent in the scientific community [11]-[14]. Making use of the inherent clustering nature of the SOM, this research addresses the within-class imbalance by using intra-cluster oversampling to generate synthetic samples within independent disjuncts of minority class samples and inter-cluster oversampling to generate synthetic samples within neighboring disjuncts of minority class samples.

Among the SOM-based oversampling techniques proposed to address the data imbalance problem, SOMO [17] has been successful in addressing both between and within-class imbalances through inter-cluster and intra-cluster oversampling. However, in this research, we adopt a different strategy to perform oversampling by using only the minority class samples to train the SOM, which allows us to get a more pure representation of the minority class without the influence of the majority class. Furthermore, contrary to the procedure followed by SOMO when determining the number of samples to generate in areas with low and high minority class densities, in this study, more samples are generated in areas with high minority class density as there is a high certainty of minority samples in these areas.

Another important aspect of this research is to determine the ability of the proposed oversampling technique to handle datasets of high dimensions. It can be observed from the consistent results obtained by the proposed oversampling technique on highly imbalanced datasets that it is undoubtedly capable of handling datasets with a large number of features as well as a large number of samples.

A major concern that arises when using SOMs or any other clustering algorithm for oversampling is determining how to preserve the decision boundary when generating synthetic minority class samples in the boundary regions.

In the proposed oversampling technique, the decision boundary preservation is achieved through the sample extraction phase. The Sample Extractor only extracts the samples occurring in areas of high minority class density and ignores the minority samples that could potentially be noisy or occur in the vicinity of majority class samples (samples in the decision boundary). Furthermore, minority class samples are again filtered during sample generation, making sure synthetic samples are generated only among topologically similar samples that occur in areas with rich density. This procedure preserves the decision boundary by preventing the generation of noisy samples around it.

In summary, based on the evaluation results achieved by the proposed oversampling technique, it can be concluded that it is indeed possible to use SOMs to formulate more robust oversampling techniques, addressing the constraints that could lead to high classification performances along with the curse of dimensionality. Even though boundary preservation is not a straightforward task in the proposed algorithm, we believe this could be a good foundation for future research to address the problem with more mathematical approaches.

VII. CONCLUSION

This research presents a SOM-based oversampling technique that attempts to address the imbalance in datasets with varying imbalance ratios and dimensions. Ten publicly available datasets with imbalance ratios ranging from 1.38 to 130, the number of features ranging from 3 to 300, and the number of samples ranging from 150 to 145,751 are evaluated on SOM-XG and compared with existing state-of-the-art techniques. The results obtained demonstrate the superiority of SOM-XG over existing techniques.

Two major limitations of SOM-XG are the inherent limitations related to SOMs, such as deciding the grid size, number of iterations and sigma, and the computational complexity introduced due to the use of two SOMs. Apart from having two SOMs, the SOM implementation not being GPU optimized also hinders the performance of the algorithm. Another limitation of the study is using a threshold to determine the number of samples required to perform sample generation. Even though the lower limit works with the datasets used for evaluation, this is not a generalized value that could be used with every dataset. Based on these limitations, it is evident that the use of two SOMs is a major bottleneck of the algorithm that affects its performance. One solution to this would be to replace the sample extraction phase with a more efficient technique. As the main intuition of the sample extraction phase is to preserve the decision boundary, using a mathematical/statistical approach that can achieve boundary preservation would also limit the algorithm to use a single SOM. Furthermore, it is also worth investigating other clustering techniques that can replace SOM in the proposed algorithm.

REFERENCES

[1] M. Vannucci and V. Colla, Imbalanced datasets resampling through self organizing maps and genetic algorithms. Springer International Publishing, 2019, vol. 1000. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-20257-6_34

[2] S. Maheshwari, J. Agrawal, and S. Sharma, "A New approach for Classification of Highly Imbalanced Datasets using Evolutionary Algorithms," International Journal of Scientific & Engineering Research, vol. 2, no. 7, 2011. [Online]. Available: http://www.ijser.org

[3] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," Information Sciences, vol. 250, pp. 113–141, Nov. 2013, doi: 10.1016/j.ins.2013.07.007.

[4] R. Mohammed, J. Rawashdeh, and M. Abdullah. 2020. Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results. 2020 11th International Conference on Information and Communication Systems, ICICS 2020 May (2020), 243–248. https://doi.org/10.1109/ICICS49469.2020.239556

[5] T. D. Piyadasa and K. Gunawardana, "A Review on Oversampling Techniques for Solving the Data Imbalance Problem in Classification," International journal on advances in ICT for emerging regions, vol. 16, no. 1, pp. 22–31, Jun. 2023, doi: https://doi.org/10.4038/icter.v16i1.7260.

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research 16, 2 (jun 2002), 321–357. https://doi.org/10.1613/jair.953

[7] G. E. A. P. A. Batista, A. L. C. Bazzan, and M. C. Monard. 2003. Balancing Training Data for Automated Annotation of Keywords: a Case Study. In Proceedings of the Second Brazilian Workshop on Bioinformatics January (2003), 35–43. http://www.cs.waikato.ac.nz/

[8] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard. 2004. A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explorations Newsletter 6, 1 (2004), 20–29. https://doi.org/10.1145/1007730.1007735

[9] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap. 2009. Safe-level-SMOTE: Safe-level-synthetic minority oversampling technique for handling the class imbalanced problem. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 5476 LNAI (2009), 475–482. https://doi.org/10.1007/978-3-642-01307-2_43

[10] H. Han, W. Y. Wang, and B. H. Mao. 2005. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In Advances in Intelligent Systems and Computing. Vol. 683. 878–887. https://doi.org/10.1007/11538059_91

[11] D. A. Cieslak, N. V. Chawla, and A. Striegel. 2006. Combating imbalance in network intrusion datasets. 2006 IEEE International Conference on Granular Computing JANUARY 2006 (2006), 732–737. https://doi.org/10.1109/grc.2006.1635905

[12] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap. 2012. DBSMOTE: Density-based synthetic minority over-sampling technique. Applied Intelligence 36, 3 (2012), 664–684. https://doi.org/10.1007/s10489-011-0287-y

[13] S. Barua, M. M. Islam, X. Yao, and K. Murase. 2014. MWMOTE - Majority weighted minority oversampling technique for imbalanced data set learning. IEEE Transactions on Knowledge and Data Engineering 26, 2 (2014), 405–425. https://doi.org/10.1109/TKDE.2012.232

[14] F. Last, G. Douzas, and F. Bacao. 2017. Oversampling for Imbalanced Learning Based on K-Means and SMOTE. (2017), 1–19. https://doi.org/10.1016/j.ins.2018.06.056

[15] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. 2001. On the surprising behavior of distance metrics in high dimensional space. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 1973 (2001), 420–434. https://doi.org/10.1007/3-540-44503-x_27

[16] T. Kohonen. 1982. Self-organized formation of topologically correct feature maps. Biological Cybernetics 43, 1 (1982), 59–69. https://doi.org/10.1007/BF00337288

[17] G. Douzas and F. Bacao. 2017. Self-Organizing Map Oversampling (SOMO) for imbalanced data set learning. Expert Systems with Applications 82 (2017), 40–52. https://doi.org/10.1016/j.eswa.2017.03.073

[18] X. Zhang, W. Wang, X. Zheng, Y. Ma, Y. Wei, M. Li, and Y. Zhang. 2019. A Clutter Suppression Method Based on SOM-SMOTE Random Forest. In 2019 IEEE Radar Conference (RadarConf). IEEE, 1–4. https://doi.org/10.1109/RADAR.2019.8835836

[19] B. W. Matthews, "Comparison of the predicted and observed secondary structure of T4 phage lysozyme," Biochimica et Biophysica Acta (BBA) - Protein Structure, vol. 405, no. 2, pp. 442–451, Oct. 1975, doi: 10.1016/0005-2795(75)90109-9.

[20] D. Dua and C. Graff. (2017). UCI Machine Learning Repository. [Online]. Available: http://archive.ics.uci.edu/ml

[21] K. Siddique, Z. Akhtar, F. Aslam Khan, and Y. Kim, ''KDD cup 99 data sets: A perspective on the role of data sets in network intrusion detection research,'' Computer, vol. 52, no. 2, pp. 41–51, Feb. 2019.

[22] LIBSVM: A Library for Support Vector Machines, C.-C. Chang and C.-J. Lin. (2001). [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm

[23] G. Vettigli, "Minisom: Minimalistic and numpy-based implementation of the self organizing map," GitHub, May 27, 2020. https://github.com/JustGlowing/minisom