



Imbalanced data oversampling through subspace optimization with Bayesian reinforcement

Mahesh Kumbhar¹ · Sunith Bandaru¹ · Alexander Karlsson²

Received: 23 December 2024 / Accepted: 30 September 2025
© The Author(s) 2025

Abstract

Many real-world machine learning classification problems suffer from imbalanced training data, where the least frequent label has high relevance and significance for the end user, such as equipment breakdowns or various types of process anomalies. This imbalance can negatively impact the learning algorithm and lead to misclassification of minority labels, resulting in erroneous actions and potentially high unexpected costs. Most previous oversampling methods rely only on the minority samples, often ignoring their overall density and distribution in relation to the other classes. In addition, most of them lack in the oversampling method's explainability. In contrast, this paper proposes a novel oversampling method that considers a subspace of the feature-set for the creation of synthetic minority samples using nonlinear optimization of a class-sensitive objective function. Suitable subspaces for oversampling are identified through a Bayesian reinforcement strategy based on Dirichlet smoothing, which may be useful for explainable-AI. An empirical comparison of the proposed method is performed with 10 existing techniques on 18 real-world datasets using two traditional machine learning classifiers and four evaluation metrics. Statistical analysis of cross-validated runs over the 18 datasets and four metrics (i.e. 72 experiments) reveals that the proposed approach is among the best performing methods in 6 and 2 instances when using random forest classifier and support vector machine classifier, thus placing it at the top. The study also reveals that some feature combinations are more important than others for minority oversampling, and the proposed approach offers a way to identify such features.

Keywords Imbalanced data · Oversampling · Nonlinear optimization · Dirichlet distribution · Bayesian reinforcement · Density-based · Features subspace · Feature importance · Explainable-AI

1 Introduction

Machine learning classification involves the prediction of a nominal target class variable (y) based on a set of features $\mathbf{x} = [x_1, x_2, \dots, x_N]$. Prediction requires an implicit or explicit approximation model of the true unknown mapping function, $y = f(x_1, x_2, \dots, x_N)$, and this model can be obtained using any of the several classification algorithms (classifiers) on training samples. All classifiers aim to minimize a so-called *loss function* that is some measure of misclassification over all training samples and classes.

A classification problem requires the classifier to learn inherent patterns that distinguish between the classes. However, the model training task can become challenging when the dataset does not adequately represent one or more classes. The model learns to favor the over-represented *majority class* over the under-represented *minority class* during training. The trained model tends to be biased in such situations, and hence it often mislabel samples belonging to the minority class. This challenging aspect of training dataset is called the *class imbalance* or *class skew* problem (Kubat and Matwin 1997); (Branco et al. 2016). Class imbalance is observed in almost all domains, such as healthcare, security, and manufacturing, and in many of these applications have the minority class is of particular interest due to its relatively high misclassification cost. Some examples include, detecting leukemia from images (Depto et al. 2023), diagnosing heart failure by capturing X-ray images (Li et al. 2023), threat detection in Internet-of-Things systems (Liang et al. 2022), and identifying faulty asset behavior (Dangut et al. 2022). The definition of class imbalance is unclear in the literature, but usually a dataset is categorized as extreme class imbalance (or rare events) when the ratio of normal samples (majority class) to rare instances (minority class) is more than 3 : 1 (Khoshgoftaar et al. 2007).

The machine learning pipeline can be divided into three distinct stages: data preparation, model building, and model deployment. Consequently, the class imbalance problem can also be addressed at any of these stages through data pre-processing, specialized model training, and post-processing of predictions (Branco et al. 2016). Firstly, *data pre-processing methods* are used before the model training stage to modify the degree and impact of class distribution based on the user preference. The main advantage of these methods is that they can be applied to any dataset, and the user can control the class proportions to be used for training. However, finding an optimal class distribution for a dataset is difficult. Secondly, *specialized learning methods* aim to modify an existing classification algorithm to take class skew into account. These methods are useful when the training dataset cannot be pre-processed for some reason. However, improving existing algorithm requires deep knowledge about their inner workings. Moreover, since such improvements are independent of the training dataset, they may not have wider applicability. Finally, *prediction post-processing methods* are used to manipulate predictions to compensate for class skewness. The main advantage with these methods is that the user does not have to think about class imbalance during model training (Branco et al. 2016).

Data pre-processing methods can be further classified based on whether they lead to a distribution change in classes or weighting of training dataset samples (Branco et al. 2016). The distribution change methods modify the data distribution by balancing frequency distributions of the target class, whereas the weighting of the training dataset uses misclassification costs to reduce prediction errors.

The specialized learning methods require a complete understanding of classifiers and the user needs. To modify a selected classifier, it is necessary to determine why it fails when the class distribution does not match with the desired preferences. Based on this information, classification algorithms are modified for the imbalanced dataset. In addition, user needs in terms of misclassification costs are also incorporated while modifying these algorithms. These methods are effective when the user understands the problem context and inputs their biases into the classification algorithms (Branco et al. 2016). Almost all classification algorithms are modified based on the user needs and misclassification costs.

Prediction post-processing methods can be further classified as threshold and cost-sensitive post-processing methods. Soft classifiers, such as logistic regression and naive Bayes, provide conditional class probabilities for samples that belongs to each class. These probabilities are used as scores, and a threshold is set to classify unknown samples. Therefore, the threshold is used as a decision criterion in threshold-based methods. In cost-sensitive learning, model predictions are adjusted with the help of misclassification costs to reduce false predictions on minority samples (Branco et al. 2016). However, these methods are rarely used in actual practice.

Almost all the pre-processing methods use Euclidean distance to measure similarity between samples across all features (Aggarwal et al. 2001). These methods may generate low quality synthetic samples due to skewed neighborhood computations during interpolation, a consequence of the curse of dimensionality. This increases variance and reduces representational accuracy of the generated instances (Fernández et al. 2018). In addition, it may be beneficial to derive value from interpretable analysis on imbalanced data to enable model transparency and support decision-making (Krawczyk 2016). The challenges that still exist are: Can representative synthetic minority samples be generated to avoid the curse of dimensionality and can an oversampling method be more interpretable and transparent to enable class imbalance explainable-AI?

The proposed method is a type of pre-processing method that considers feature subspaces to generate minority features using nonlinear optimization and uses other features as original to form a synthetic minority sample. The generation of each minority sample produces an objective value from optimization, and it is used to reinforce the future selection of feature subspaces, which are able to distinguish majority and minority samples. The optimization problem involves minimizing a function and lower objective function value produced by a feature subspace likely to get reinforced using Bayesian reinforcement with the help of Dirichlet distribution. The method may select a feature subspace with a higher objective value, as obtaining the high objective value for generating one minority sample does not undermine its superior performance over other subspaces.

The proposed method is validated through experiments on 18 real-world datasets exhibiting varying degrees of class imbalance, ranging from low to high. A comparative analysis is conducted with other pre-processing methods using random forest (RF) and support vector machine (SVM) classifiers. The performance evaluations are carried out using multiple evaluation metrics on both classifiers. The proposed method achieves 6 and 2 performance scores with RF and SVM classifiers, respectively. This score is the count of non-significant statistical comparisons across all datasets and evaluation metrics, as determined by Dunn's test, with a significant median difference observed by the Kruskal-Wallis test. The performance score is the highest for RF and SVM classifiers. Additionally, the proposed method attains a top-two average rank when compared to other pre-processing methods. To validate

these findings, Friedman and post-hoc Nemenyi statistical tests are performed, which have shown the proposed method comes in the top rank across all evaluation metrics, thereby confirming its superior performance.

1.1 Contributions and limitations

The main contributions of the paper are as follows:

- A class-sensitive objective function captures density and distribution of both minority and majority samples for effective class discrimination on a low-dimensional feature subspace, to avoid the curse of dimensionality for neighborhood computation during interpolation.
- The optimization objective values are used to reinforce the selection of feature subspaces based on the Bayesian Dirichlet distribution for class imbalance explainable-AI.

The proposed methodology has the following limitations.

- The cardinality of the feature subspace is restricted to two or three, resulting in a large number of feature combinations for high-dimensional dataset.
- The proposed method has higher computational cost when compared to existing over-sampling methods, which may impact scalability, especially when applied to high-dimensional dataset. However, unlike these existing methods, our approach provides additional insights into the importance of features.

The remainder of this paper is organized as follows. Section 2 highlights related work in the area of pre-processing methods. Next, Sect. 3 introduces the proposed method. This is followed by Sect. 4 which presents experimental settings, and Sect. 5 provides results and discussion. Finally, the last Sect. 6 gives conclusions and future research.

2 Related work

In the past two decades, much research has been done on the problem of imbalanced classification due to the widespread adoption of machine learning in real-world situations. Practical challenges, such as false predictions on rare events, have increased the research to modify existing machine learning algorithms to use them in actual practice. Many methods and improvements are developed to address imbalanced datasets, and these have been analyzed and extensively discussed in some of the literature reviews presented by (Krawczyk 2016b; Branco et al. 2016). This section presents an overview of existing literature concerning data pre-processing methods, of which our proposed method is a part of. These methods are categorized into distribution change and weighting of training dataset. The distribution change is further categorized into three types: stratified sampling, synthesizing new data, and a combination of both types.

Stratified sampling methods involve the addition or removal of samples from the original training dataset. This can be achieved through random oversampling, random undersampling, data cleaning, clustering, distance-based elimination, or evolutionary algorithms.

Random oversampling replicates randomly selected minority samples; however, as pointed out by Chawla et al. (2002), it may result in overfitting. In contrast, *random undersampling* removes random majority samples, and such undersampling may also result in worse performance (Menardi and Torelli 2014). Distance measures, such as the Euclidean distance, can be used to overcome the limitations of random oversampling and random undersampling methods. In *data cleaning* methods, (Kubat and Matwin 1997) have removed majority samples with the help of Tomek links to obtain a reasonable size of majority samples. Similarly, in the undersampling of majority samples with *distance-based elimination* methods, only those majority samples are selected which are closest or farthest from a few of the minority samples (Mani and Zhang 2003). Likewise, Tomek (1976) determined the decision boundary and removed some noisy samples only when the closest neighbors belong to the same majority class or a different class. In *clustering based methods* of oversampling minority samples, Jo and Japkowicz (2004) have proposed finding initial k -means clustering depending on number of classes, and subsequently generating random minority samples. Finally, from an *evolutionary algorithm's* perspective, García et al. (2006) have introduced a version of a genetic algorithm to generate minority samples. The method involves selecting and evaluating a subset of the training dataset using a fitness function with 0 and 1 states. If the fitness function yields 1, then the corresponding data point is saved as a new minority sample; otherwise, it is rejected. The fitness function combines the classification rate obtained with 1-NN classification and how much imbalance reduction is achieved during sequential samples generation.

Synthesizing new data is an oversampling method that involves creating artificial minority samples. New synthetic samples are generated either by *perturbation* or *interpolation* of the existing minority samples (Chawla et al. 2002; Branco et al. 2016). Perturbation simply means by adding Gaussian noise to existing minority samples. For example, Zhang et al. (2023) have proposed a perturbation method, where it generates minority samples by perturbing each feature of the minority samples. The amount of perturbation depends on a hyperparameter. In contrast, interpolation generate synthetic samples by interpolating between minority samples.

Chawla et al. (2002) proposed the first interpolation method, which is *synthetic minority oversampling technique* (SMOTE). It selects any random minority sample and its closest k neighbors. After this, one of the samples is chosen out of the k neighbors and feature difference is calculated between the chosen sample and the reference selected sample. A random number is generated and multiplied by the difference between the feature's distance, which is then added to the initially selected sample. Therefore, a created sample is interpolated between the selected sample and the one of the neighbors. This oversampling technique is successful because it can use any machine learning algorithm to form a balanced dataset by balancing the class distribution. However, the SMOTE method is unable to address within-class imbalance, wherein certain subsets contain significantly fewer minority samples than others. In addition, it fails when a dataset has small disjuncts in which a few minority samples appear inside majority of samples (Islam et al. 2022). Furthermore, SMOTE oversampling may improve classifier performance on minority samples; however, it may add noise by generating synthetic samples inside the majority samples. These limitations are reduced by its variants, which apply post-processing to remove some minority samples generated by the SMOTE. Batista et al. (2004) have used Tomek links to remove both minority and majority samples after oversampling. In addition, the author also proposed *SMOTE edited*

nearest neighbor (SMOTE-ENN), which removes a sample where a minimum of two nearest neighbor labels differs from the selected sample. The SMOTE-ENN approach removes more samples from an oversampled dataset compared to that of SMOTE-Tomek.

The second type of SMOTE variant generates synthetic samples only in the specific regions, such as on class boundaries, which are considered the most helpful for learning a classifier. But, finding a good region is not straightforward, and it is generally difficult to know exact class boundaries. Therefore, modified SMOTE methods have been proposed to precisely capture class boundaries for the creation of synthetic samples. These methods include *borderline-SMOTE* (Han et al. 2005), *adaptive synthetic sampling* (ADASYN) (He et al. 2008), *modified synthetic minority oversampling technique* (MSMOTE) (Hu et al. 2009), *majority weighted minority oversampling technique* (MWMOTE) (Barua et al. 2014), *kmeans-SMOTE* (Douzas et al. 2018), and *SVM-SMOTE* (Nguyen et al. 2011).

(Han et al. 2005) proposed *borderline-SMOTE*, which identifies border samples and generates synthetic samples near to it. For each minority sample, the number of majority nearest neighbors (m) are determined among k -nearest neighbors (k -NN). If $k/2 \leq m \leq k$, a minority sample is at risk of misclassification. If $0 \leq m \leq k/2$, a sample is considered safe and does not require a synthetic sample. The risk of misclassifying samples are called *borderline samples*, and subsequently, the standard SMOTE is applied to generate synthetic samples. *Borderline-SMOTE* overcomes the drawback of SMOTE; however, inaccurate identification of border minority samples leads to poor generation of synthetic samples. He et al. (2008) introduced ADASYN method, which assigns higher weightage to samples that are harder to learn compared to easier to learn samples. This method is fine-tuned before using the standard SMOTE. It determines the weights corresponding to each sample with a ratio of majority samples out of k -NN samples. This ratio is normalized by considering the number of synthetic samples required for each minority sample. Based on this, new synthetic samples are generated with the SMOTE method. This method has some advantages, such as it forces the SMOTE method to learn difficult samples closely instead of non-difficult ones. To address the challenge of identifying the correct *borderline samples*, Hu et al. (2009) proposed MSMOTE method. Initially, minority samples are classified into security, noise, or border samples based on k -NN. A sample is tagged as a security sample when a minority sample and its k -NN have the same minority class. If a sample has all the majority samples close to its k -NN, then it is tagged as a noise sample; otherwise, it is categorized as a border sample. After this minority sample tagging, synthetic samples belonging to security and border samples are generated using the SMOTE method. ADASYN and MSMOTE methods use generation of minority samples based on k -NN, which may result in duplicate and wrong synthetic minority samples. To take into account, the position and distance of neighbors from the minority samples, Barua et al. (2014) proposed MWMOTE. This method consists of three steps involving finding hard to learn minority samples, giving weight to each minority sample, and finally generating synthetic samples. Difficult minority samples are found in three sequential steps. First, for each minority sample, it finds k_1 nearest minority samples, and around these samples, it filters k_2 nearest majority samples. Subsequently, on these majority samples, k_3 nearest minority samples are determined. In the second phase, for each minority sample, a selection weight is determined by combining the density and closeness of samples, to create clusters near each minority sample. Finally, a minority sample, x_1 , is selected from a cluster from the selection probability and another random sample, x_2 , is selected from the same cluster. These two samples are combined

together with, $\mathbf{x}_1 + \text{random number} * (\mathbf{x}_2 - \mathbf{x}_1)$, to form a synthetic minority sample. This method is able to generate synthetic samples inside minority class clusters.

Further continuous modifications are seen in the SMOTE method. kmeans-SMOTE is proposed by Douzas et al. (2018), consisting of three steps: clustering, filtering, and oversampling. In the first step, clustering is used to determine k groups in the training dataset. Subsequently, the filtering step refines the formed clusters by assigning more minority samples to clusters with sparse minority representation. Finally, the SMOTE method is applied to generate synthetic samples, ensuring the required balance between minority and majority samples within each cluster. Furthermore, Nguyen et al. (2011) emphasizes and leverages the decision boundary from SVM classifier. The method involves generating synthetic samples near the class decision boundary. Initially, the SVM is utilized to identify minority and majority support vectors. It identifies k -NN for each majority sample of the support vector to generate synthetic samples. If the number of majority samples is less than its k -NN, extrapolation is employed; otherwise, interpolation is used. In a recent development, Islam et al. (2022) introduced *k-nearest neighbor oversampling* (KNNOR) method, which consists of two steps: identifying a critical safe region and generating synthetic samples using multiple minority samples. Initially, a critical and safe region is determined for each minority sample by considering k -th nearest neighbor distance. These distances are then sorted in descending order and filtered based on the distance threshold, resulting in a subset of minority samples. Subsequently, synthetic samples are generated using subsets of these minority samples with the help of the SMOTE method. Instead of linear interpolation as in SMOTE, Gazzah and Amara (2008) has proposed a polynomial fitting that generates synthetic minority samples by fitting a curve using the selected minority samples. It works by selecting a minority and choosing neighboring samples based among a chosen topology, such as star, polynomial curve, bus, and mesh- and then fitting a curve. In particular, mesh topology (polyfit-mesh) forms a densely connected local graph, creating overlapping paths and patterns. Recently, Sağlam and Cengiz (2022) proposed SMOTE with boosting (SMOTEBW), which first fits AdaBoost classifier to a dataset and sample weights are computed based on prediction error - using the log odds, which compares the probability of correct classification to the probability of error. The weights are higher for the minority samples than the majority. Minority samples are filtered with greater than the reciprocal of the sample size. Finally, SMOTE is applied on these selected samples to generate new instances, enhancing the representation of informative minority samples. In addition, Barua et al. (2013) proposed Proximity Weighted Synthetic Oversampling Technique (ProWSyn), which partitions minority samples into many partitions based on distances from the decision boundary using Euclidean distance. Each level gets a proximity level with increasing distances from a decision boundary. Depending on proximity levels, synthetic samples are generated. Kumar et al. (2024) proposed entropy-based hybrid sampling (EHS), which removes the majority samples using a threshold entropy of majority samples and generates synthetic samples on the majority-minority border lines using a threshold entropy of minority samples. Dai et al. (2025) proposed graph-based generalized quadrilateral element oversampling (GQEO). It finds the global structure of minority samples using k -NN, giving a weighted adjacency matrix to represent the proximity relationship. Then, screening is done to remove remote minority samples with a planar quadrilateral constraint, sum of any three sides must be greater than the fourth side. On the remaining samples, synthetic samples are generated using a 1D interpolation function.

Minority samples can be generated through optimization, and one of the method is proposed by Shahee and Ananthakumar (2020). This approach employs Lagrange optimization in three steps. Initially, sub-clusters are identified through model-based clustering. Subsequently, normalized Jeffrey divergence is computed on both classes within these sub-clusters. Finally, a geometric mean of normalized Jeffrey divergence, a new distance metric, captures the separation between and within classes. However, this method assumes m sub-clusters determined by model-based clustering and assumes each sub-cluster follows a multivariate Gaussian distribution. In addition, it does not consider class overlap and targets redundant features. In this paper, the authors highlights the requirement of fewer important features for discriminating the imbalanced dataset.

In summary, almost all the previously described methods perform similarly by selecting among minority samples or focusing on border regions and applying the SMOTE method to generate minority samples. In addition, some methods use polynomial interpolation instead of linear interpolation to overcome the nonlinear behavior of a local region. Most of them use Euclidean distance to measure similarity between samples across all features (Aggarwal et al. 2001). To avoid the problem of the abnormal behavior of Euclidean distance and to generate representative minority samples using a nonlinear optimization search are vital for addressing the class imbalance problem. Furthermore, sometimes an insignificant feature may significantly improve the classification performance in high dimensions (Guyon and De 2003). Therefore, this paper aims to address the problem of minority samples generation using nonlinear optimization on feature subspaces and reinforces the probability of those subspaces based on their ability to distinguish between minority and majority samples.

3 Proposed method

We illustrate the working of our proposed method, Subspace Optimization with Bayesian Reinforcement (SOBER), by applying it to the two-dimensional artificial imbalanced dataset named paw02a-600-5-50-BI (Napierała et al. 2010), taken from *Knowledge Extraction based on Evolutionary Learning* (KEEL) repository (Alcalá-Fdez et al. 2011). This dataset, shown on the left in Fig. 1, has different sub-regions of minority samples with varying cardinalities, which can pose difficulties for some classification algorithms (Stefanowski 2013; Pradipta et al. 2021). SOBER is an oversampling approach that takes into account the density and distribution of both majority and minority samples for iteratively generating synthetic minority samples. The effect of the approach is seen on the right in Fig. 1, where the two classes are completely balanced.

SOBER works by defining a class-sensitive objective landscape, shown in Fig. 2, as a function of the dataset features. The objective function is designed to take low values in regions where the density of minority samples is low, and higher values where more majority samples are present. For any given sample in the feature space, the objective function can be constructed by first defining a neighborhood and then taking the ratio of the number of majority samples to that of minority samples within that neighborhood. The minimization of such an objective function can lead to a new sample resembling other minority samples. To generate multiple new samples that are different from each other, the objective function is attenuated by some random noise in SOBER. This is how the samples in Fig. 1 are generated.

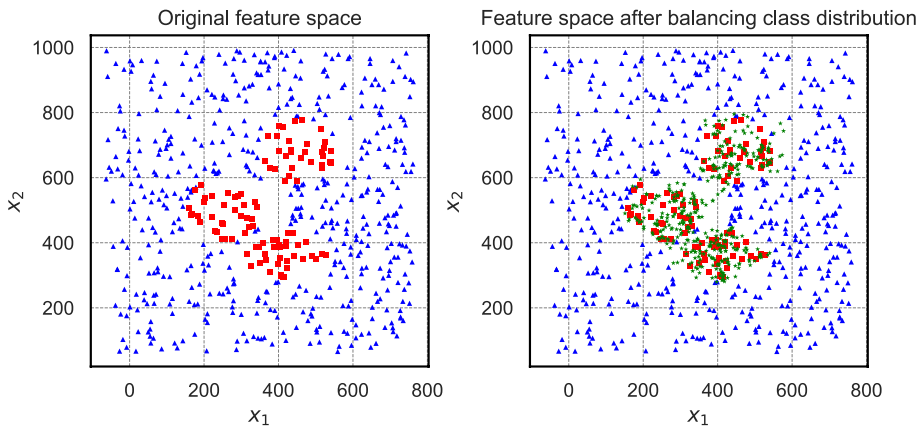


Fig. 1 An illustrative imbalanced dataset (left) and result of oversampling using SOBER (right). Blue triangles represent majority samples, red squares represent original minority samples, and green stars represent synthetic minority samples generated by SOBER

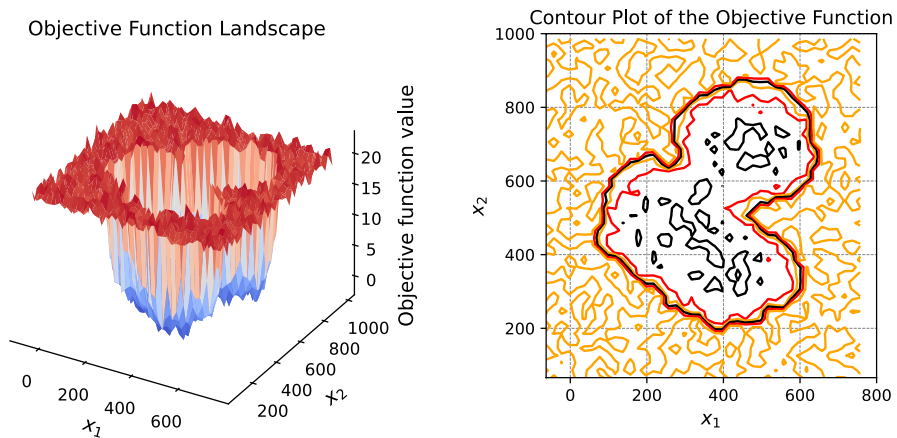


Fig. 2 The plot on left side shows objective function landscape and the plot on the right side shows contour plot on synthetic dataset

In high-dimensional feature spaces, the concept of neighborhood, as employed above, falls apart. Moreover, if the minimization of the objective function is unsuccessful due to high number of features or limited iterations of the optimization algorithm, the new sample may actually represent the majority class instead of the desired minority class. To avoid these issues, SOBER only operates in feature subspaces (that we call L -subspaces). Each new sample is generated in a subspace which is probabilistically selected based on the performance of subspaces in the generation of previous samples. This performance can be measured in terms of the achieved objective function values. The probability of subspaces with lower achieved objective function values are reinforced positively.

The core idea of SOBER approach is shown in Fig. 3 for six features and a subspace dimension of two. Starting with a random minority sample x , SOBER first selects an

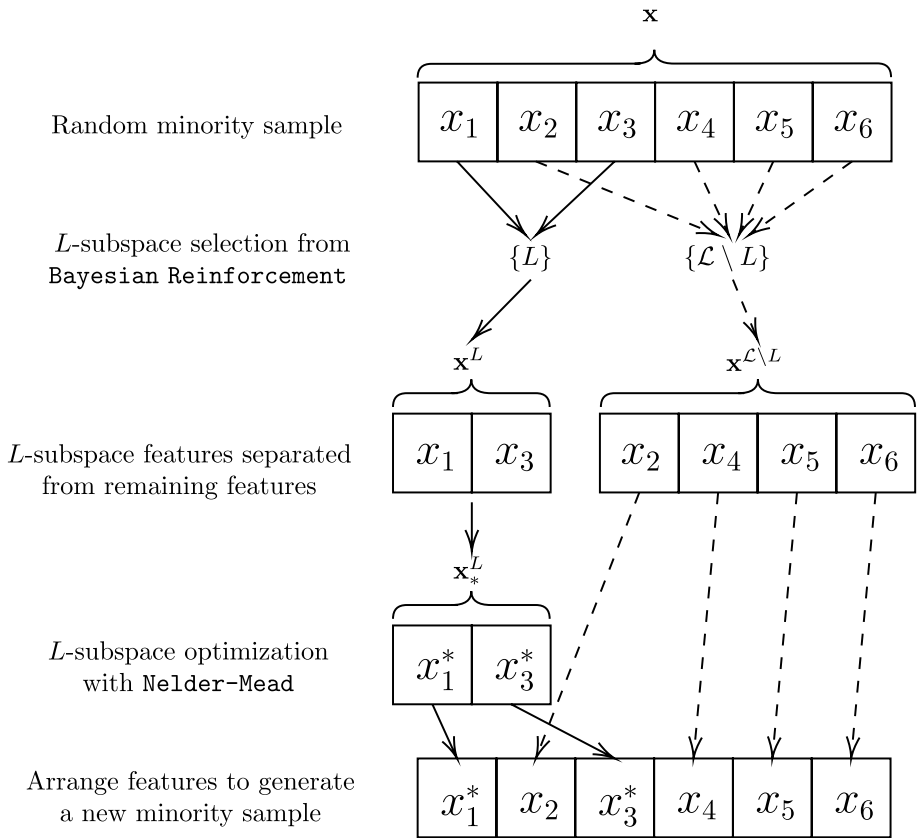


Fig. 3 Schematic representation of minority sample generation in SOBER with $N = 6$ and $d = 2$. Starting with a random minority sample x , SOBER probabilistically selects an L -subspace using Bayesian Reinforcement. The subspace feature vector, x^L , is then optimized using Nelder-Mead to obtain x_*^L . Finally, these optimized values are arranged together with the remaining features in $x^{L \setminus L}$, resulting in a new minority sample

L -subspace based on probabilities obtained from Bayesian Reinforcement (described in Sect. 3.2). With the separated feature vector, x^L , in this subspace as the initial starting point, L -subspace optimization is performed using the Nelder-Mead method (described in Sect. 3.3) to minimize the class-sensitive objective function described above. The obtained optimum point, x_*^L , is finally merged with the remaining part of the original feature vector, $x^{L \setminus L}$, to generate a new minority sample. This process is repeated until the training dataset is completely balanced. The final probabilities of the subspaces capture which features are effective at distinguishing between the minority and majority samples.

We provide a complete description of SOBER in the following sections through Algorithms 1, 2 and 3, which show pseudocodes for the objective function calculation, Bayesian reinforcement, and overall minority sample generation, respectively. The source code of SOBER is available on <https://github.com/MaheshKumbhar/SOBER>. Table 1 shows the mathematical notation used throughout the rest of the paper.

Table 1 Mathematical notation used in this paper

Notations	Definitions
M	Number of instances in a training dataset
N	Number of features in a training dataset
\mathbf{x}	Feature vector, i.e., $\mathbf{x} = [x_1, x_2, \dots, x_N]$
\mathbf{z}	Data vector with features and class label, i.e., $\mathbf{z} = [\mathbf{x} \quad y]$
Z	Indexed training dataset, i.e., $\mathbf{z}_i = [\mathbf{x}_i \quad y_i]$ for $i = 1, 2, \dots, M$
\mathcal{L}	Set of features, i.e., $\mathcal{L} = \{x_1, x_2, \dots, x_N\}$
L	Subset of features that form a subspace, i.e., $L \subseteq \mathcal{L}$
$\mathbf{x}^L, \mathbf{z}^L, Z^L$	L -subspace of $\mathbf{x}, \mathbf{z}, Z$ respectively
\mathcal{I}_Z	Index set for data vectors, i.e., $\mathcal{I}_Z = \{1, 2, \dots, M\}$
\mathcal{I}_L	Index set for L -subspaces
c^-	Class label (y_i) for majority samples
c^+	Class label (y_i) for minority samples
α	Concentration parameter

Require: Z^L - L -subspace of training dataset, k - Number of nearest neighbors, \mathbf{z}_t^L - L -subspace of the candidate minority sample at iteration t , i.e., $\mathbf{z}_t^L = [\mathbf{x}_t^L \quad c^+]$

- 1: $\mathcal{I}_{local} \leftarrow \{i \in \mathcal{I}_Z \mid \mathbf{x}_i^L \text{ are } k\text{-nearest neighbors of } \mathbf{x}_t^L\}$
- 2: $n^- \leftarrow |\{i \in \mathcal{I}_{local} \mid y_i = c^-\}|$ \triangleright Majority samples in the neighborhood of \mathbf{x}_t^L
- 3: $n^+ \leftarrow |\{i \in \mathcal{I}_{local} \mid y_i = c^+\}|$ \triangleright Minority samples in the neighborhood of \mathbf{x}_t^L
- 4: $objFun \leftarrow \frac{n^-}{n^++1.0} + \gamma$, where $\gamma \sim \mathcal{N}(0, 1)$

Output: $objFun$

Algorithm 1 ObjectiveFunction

3.1 Creation of feature subspaces and objective function

The first step of SOBER is to create feature subspaces within which new minority samples will be generated. The cardinality of the subspaces, d , is motivated by the use of square root of number of features in tree generation with random forests (Breiman 2001). However, we limit the subspace cardinality to three if $\sqrt{N} \geq 3$ (see Algorithm 3, line 2). Next, SOBER creates $\binom{N}{d}$ L -subspaces (see Algorithm 3, line 2).

The calculation of the objective function within any of these subspaces is shown in Algorithm 1. It requires L -subspace of the original training dataset Z^L , L -subspace of a candidate minority sample \mathbf{z}_t^L at iteration t , and k parameter for k -NN to define a neighborhood of samples around \mathbf{x}_t^L , given by \mathcal{I}_{local} . Subsequently, we count majority samples, denoted as n^- and minority samples, denoted as n^+ from the neighborhood of \mathbf{x}_t^L . Finally, the objective function value, $\frac{n^-}{n^++1.0} + \gamma$, where $\gamma \sim \mathcal{N}(0, 1)$, is calculated (see Algorithm 1, lines 2 to 4). Therefore, the defined objective function captures density and class distribution values.

Require: *subFunAvg* - Mean objective values for all L -subspaces, *ficObsCount* - Count of fictive observations for all L -subspaces, α - Concentration parameter, \mathcal{I}_L - Index set for all L -subspaces

```

1: for  $j \leftarrow 1$  to  $|\mathcal{I}_L|$  do
2:    $subFunAvg[i] \leftarrow subFunAvg[i] - \max(subFunAvg)$ 
3: end for
4:  $normFunAvg \leftarrow [0]_{1 \times |\mathcal{I}_L|}$   $\triangleright$  Data structure for normalized function values
5: for  $j \leftarrow 1$  to  $|\mathcal{I}_L|$  do
6:    $normFunAvg[i] \leftarrow \frac{subFunAvg[i]}{\sum_{i=1}^{|\mathcal{I}_L|} subFunAvg[i]}$ 
7: end for
8:  $p \leftarrow [0]_{1 \times |\mathcal{I}_L|}, n_{obs} \leftarrow [0]_{1 \times |\mathcal{I}_L|}$   $\triangleright$  Probability and fictive observation vector
9: for  $j \leftarrow 1$  to  $|\mathcal{I}_L|$  do
10:   $n_{obs}[i] \leftarrow \alpha \times normFunAvg[i]$   $\triangleright$  Incremental fictive observations
11: end for
12: for  $j \leftarrow 1$  to  $|\mathcal{I}_L|$  do
13:   $p[i] \leftarrow \frac{ficObsCount[i] + n_{obs}[i] + 1.0}{\sum_{i=1}^{|\mathcal{I}_L|} \{ficObsCount[i] + n_{obs}[i]\} + |\mathcal{I}_L|}$   $\triangleright$  Probability of occurrence
14: end for
Output:  $p, n_{obs}$ 

```

Algorithm 2 BayesianReinforcement

3.2 Bayesian reinforcement for feature subspaces

The illustration of Bayesian reinforcement on a function $f(x) : 2 \sin(x) + \sin(0.5x)$ is shown in Fig. 4. L -subspaces are generated using x values from 2 to 15 with steps of 1.5. Then SOBER is used for generating 100 samples, and Fig. 4 shows the reinforced probability of those L -subspaces having the lowest objective values. However, it is also interesting to note that we still give a fair chance for the selection of other L -subspaces having higher objective values.

In order to select a L -subspace, we apply a probabilistic schema based on Bayesian methodology (Gelman et al. 2013) to avoid zero probabilities when a L -subspace has yet to be selected and evaluated. Before evaluating the objective function, we will adopt a uniform prior (Dirichlet) to all L -subspaces and update to a posterior (Dirichlet) by thinking in terms of a fictive number of observations α which will determine how much potential contribution a certain L -subspace can receive in each draw. In principle, since we solve a minimization problem, a L -subspace with a low objective function value is interesting to select again. Hence, we would like more probability mass to concentrate around this L -subspace. To start, we will select two distinct L -subspaces using uniform distribution (see Algorithm 3, lines 13 to 14) for generating the first two minority samples, and after referenced objective function values from optimization, we can reinforce the importance of few of the L -subspaces over other L -subspaces iteratively.

We keep generating samples and accumulating objective values f^* into *subFunVal* (see Algorithm 3, lines 34 and 38). We compute average objective values (*subFunAvg*) for already occurred L -subspaces (see Algorithm 3, lines 16 to 20). For the remaining L -sub-

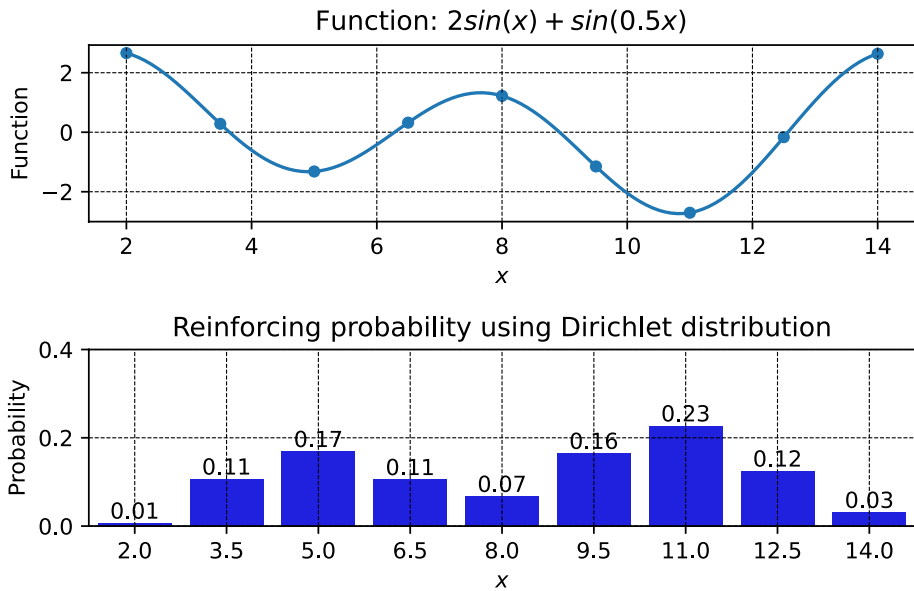


Fig. 4 Reinforcing selection of L -subspaces (discrete points) for $2\sin(x) + \sin(0.5x)$ function. The top image shows function values across and the bottom image shows the probabilities across L -subspaces

spaces, which are yet to appear, we assign the maximum average objective value from the already occurred L -subspaces (see Algorithm 3, line 21 to 26).

Following this, we reinforce and update the L -subspaces probabilities to select a next L -subspace in Algorithm 2. To obtain this, we maintain relative distances by shifting average objective function values with a maximum average objective value. This shift will ensure that all the objective values are on one side of the measuring scale (see Algorithm 2, lines 1 to 3). Subsequently, we normalize the average objective values, denoted as $normFunAvg$ (see Algorithm 2, lines 5 to 7) and is calculated as,

$$normFunAvg[i] = \frac{subFunAvg[i]}{\sum_{i=1}^{|I_L|} subFunAvg[i]} \quad (1)$$

and then we find incremental fictive observations, denoted as n_{obs} , by using normalized objective values and a concentration parameter (see Algorithm 2, lines 9 to 11). We keep updating the previous fictive observations with the new ones (see Algorithm 3, lines 28 to 30), which is given by,

$$ficObsCount[j] = ficObsCount[j] + n_{obs}[j] \quad (2)$$

Finally, these fictive observations can be used for calculating the posterior predictive distribution (see Algorithm 2, line 12 to 14):

$$p[i] = \frac{ficObsCount[i] + 1.0}{\sum_{i=1}^{|\mathcal{I}_{\mathbb{L}}|} ficObsCount[i] + |\mathcal{I}_{\mathbb{L}}|}. \quad (3)$$

Notice that when no fictive observations exist, one obtains the uniform distribution (prior predictive distribution). Now, in principle, we want a certain L -subspace to gain only a few fictive observations before properly exploring the use of other L -subspaces since that would mean a high concentration in terms of probability mass for the selected L -subspace. The chosen L -subspace may dominate the entire oversampling process, resulting in non-selection of other L -subspaces that could be potentially good but were never selected.

To address this, we use a parameter α concerning the importance of L -subspaces based on its relative objective values for generating new minority samples (n_s). Initially, a prior probability is $1/|\mathcal{I}_{\mathbb{L}}|$ for all L -subspaces. Suppose that any selected L -subspace contributes entirely to the fictive observations. In that case, the normalized objective function for that L -subspace becomes much higher, and the respective probability of that L -subspace becomes closer to 1. To avoid this scenario, the maximum fictive observations for the L -subspace is restricted to a single probability state $p[i]$ as,

$$\max p[i] = \frac{n_s \times \alpha + 1}{n_s \times \alpha + |\mathcal{I}_{\mathbb{L}}|} \leq \beta \quad (4)$$

with a maximum allowed probability concentration β which is set to 0.95. This gives the maximum value for α (see Algorithm 3, line 11) as,

$$\alpha \leq \frac{(\beta * |\mathcal{I}_{\mathbb{L}}|) - 1.0}{(1.0 - \beta) * n_s} \quad (5)$$

Subsequently, using categorical distribution, we simulate and choose a L -subspace for generating the next minority sample. (see Algorithm 2, line 29).

Input: \mathbf{Z} - Training dataset, β - Maximum allowed concentration

```

1: if  $\lfloor \sqrt{N} \rfloor \geq 3$  then  $d \leftarrow 3$  else  $d \leftarrow 2$ 
2:  $\mathbb{L} \leftarrow [L_1, \dots, L_{|\mathbb{L}|}]$ , where  $\mathbb{L}$  is an index set i.e.  $|\mathbb{L}| = \binom{N}{d}$ 
3:  $n^- \leftarrow |\{i \in \mathcal{I}_Z \mid y_i = c^-\}|$   $\triangleright$  Number of majority samples in  $\mathbf{Z}$ 
4:  $n^+ \leftarrow |\{i \in \mathcal{I}_Z \mid y_i = c^+\}|$   $\triangleright$  Number of minority samples in  $\mathbf{Z}$ 
5:  $n_s \leftarrow n^- - n^+$   $\triangleright$  Number of minority samples to be generated
6:  $samples \leftarrow [\mathbf{0}]_{n_s \times N}$   $\triangleright$  Data structure for new minority samples
7:  $subFunVal \leftarrow [\infty]_{1 \times |\mathbb{L}|}$   $\triangleright$  Data structure for summing objective values
8:  $subFunAvg \leftarrow [\infty]_{1 \times |\mathbb{L}|}$   $\triangleright$  Data structure for mean of objective values
9:  $subFunCount \leftarrow [\mathbf{0}]_{1 \times |\mathbb{L}|}$   $\triangleright$  Data structure for counting  $L$ -subspace selections
10:  $ficObsCount \leftarrow [\mathbf{0}]_{1 \times |\mathbb{L}|}$   $\triangleright$  Data structure for counting fictive observations
11:  $\alpha \leftarrow \frac{(\beta * |\mathbb{L}|) - 1.0}{(1.0 - \beta) * n_s}$   $\triangleright$  Concentration parameter
12: for  $i \leftarrow 1$  to  $n_s$  do  $\triangleright$  Minority sample generation starts here
13:   if  $\nexists \{j, k\} \mid (j \neq k \wedge subFunAvg[j] \neq \infty \wedge subFunAvg[k] \neq \infty)$  then
14:      $l_{sub} \sim \text{Uniform}(\mathbb{L})$   $\triangleright$  Index of randomly selected  $L$ -subspace
15:   else
16:     for  $j \leftarrow 1$  to  $|\mathbb{L}|$  do
17:       if  $subFunCount[j] \neq 0$  then
18:          $subFunAvg[j] \leftarrow \frac{subFunVal[j]}{subFunCount[j]}$ 
19:       end if
20:     end for
21:      $\bar{f}_{max} \leftarrow \max_{j \in \mathbb{L}} \{subFunAvg[j] \mid subFunCount[j] \neq 0\}$ 
22:     for  $j \leftarrow 1$  to  $|\mathbb{L}|$  do
23:       if  $subFunCount[j] = 0$  then
24:          $subFunAvg[j] \leftarrow \bar{f}_{max}$ 
25:       end if
26:     end for
27:      $p, n_{obs} \leftarrow \text{BayesianReinforcement}(subFunAvg, ficObsCount, \alpha)$ 
28:     for  $j \leftarrow 1$  to  $|\mathbb{L}|$  do
29:        $ficObsCount[j] \leftarrow ficObsCount[j] + n_{obs}[j]$ 
30:     end for
31:      $l_{sub} \sim \text{Categorical}(\mathbb{L}, p)$   $\triangleright$  Index of Bayesian reinforced  $L$ -subspace
32:   end if
33:    $j \sim \text{Uniform}(\{j \in \mathcal{I}_Z \mid y_j = c^+\})$   $\triangleright$  Index of random minority sample
34:    $f_*, \mathbf{x}_*^L \leftarrow \text{Nelder-Mead}(\text{ObjectiveFunction}, \mathbf{z}_j^L, \mathbf{Z}^L)$  where  $L = \mathbb{L}[l_{sub}]$ 
35:    $\mathbf{z}_s \leftarrow [\text{arrange}(\mathbf{x}_*^L, \mathbf{x}_j^{c \setminus L}) \ c^+]$   $\triangleright$  Arrange features (See Figure 3)
36:    $samples[i, :] \leftarrow \mathbf{z}_s$ 
37:    $subFunCount[l_{sub}] \leftarrow subFunCount[l_{sub}] + 1$ 
38:    $subFunVal[l_{sub}] \leftarrow subFunVal[l_{sub}] + f_*$ 
39: end for

```

Output: $samples$

Algorithm 3 Generation of minority samples

3.3 Subspace optimization with Nelder–Mead method

A direct unconstrained nonlinear optimization is necessary to find an optimal solution in a subspace of high dimensions. In the current approach, we have used Nelder-Mead (Nelder and Mead 1965), which constructs a simplex of $d + 1$ vertices polyhedron for N dimensions. This algorithm requires an objective function, k parameter for k -NN, L -subspace of original training dataset Z^L , and L -subspace of a minority sample z^L (see Algorithm 3, line 34). The output of this optimization produces an optimized L -subspace values x_*^L (See Fig. 3) and objective value f_* .

The Nelder-Mead search algorithm exhibits direct search capabilities without requiring the computation of function derivatives. Let v_0, v_1, \dots, v_d be the d vertices of a simplex, and objective values ($f(v_d)$) are determined for all these vertices. All vertices are sorted, such that $f(v_0) \leq f(v_2) \leq \dots \leq f(v_d)$. From these objective values, a vertex v_d with the highest objective function value is identified by $f_h = f(v_d)$, while v_0 is a vertex with the lowest objective function value, denoted as $f_l = f(v_0)$.

Consider \bar{v} as the centroid of all vertices. The algorithm consists of three operations: reflection, expansion, and contraction. The standard parameter values are $\theta_r = 1$, $\theta_e = 2$, and $\theta_c = 0.5$, respectively. During each of these operations, the algorithm replaces v_d with a new vertex.

- The algorithm computes a reflection point, denoted as $v_r = \bar{v} + \theta_r(\bar{v} - v_d)$, and resulting in a function value of f_r . If $f_l \leq f_r < f_h$, then the algorithm replaces v_d with v_r and restarts with the new simplex.
- If $f_r < f_l$, the algorithm expands further to find a new vertex $v_e = \bar{v} + \theta_e(v_r - \bar{v})$, and resulting in a function value of f_e . If $f_e < f_r$, then the algorithm replaces v_d with v_e ; otherwise, the algorithm replaces v_d with v_r and restarts with the new simplex.
- If $f_r > f_h$ for all $i \neq h$, the algorithm chooses the old vertex with f_h or f_r , whichever is minimum, and finds a contraction, denoted as $v_c = \bar{v} + \theta_c(v_d - \bar{v})$, resulting in a function value of f_c . The algorithm then accepts v_c for v_d and restarts the process.

Thus, this algorithm explores the multidirectional space in search of a critical point with the lowest minimum objective function value. It achieves this by iteratively obtaining new vertices with monotonically decreasing function values. However, this algorithm performs well with lower dimensions, but becomes inefficient for large dimensions Gao and Han (2012). Therefore, we have considered L -subspaces while doing nonlinear optimization.

3.4 Formation of a new minority sample

The complete minority sample (z_s) is formed by arranging the optimized values from Nelder-Mead in the selected L -subspace (x_*^L) together with the remaining feature values ($x^{\mathcal{L} \setminus L}$) of the original minority sample and minority class label. This is done through the arrange function in Algorithm 3 (line 35), as illustrated in Fig. 3. This sample is appended to a data structure called *samples*. Next, the count for the number of times the corresponding L -subspace is selected, i.e., *subFunCount*, is incremented by one. And finally, the optimum objective function value, f_* , is added to *subFunVal* as a measure of L -subspace reinforcement. These three steps can be seen in Algorithm 3, lines 36 to 38.

4 Experimental methodology

The proposed method is evaluated on representative real-world imbalanced datasets. We selected pre-processing methods such as data distribution and data weighting. Subsequently, machine learning classifiers are trained after balancing a training dataset using pre-processing methods and evaluating them on the testing dataset. Here, we chose two classifiers to compare SOBER with existing pre-processing methods. This section summarizes the characteristics of datasets, various evaluation metrics, the selection of classifiers, parameter settings of the compared methods, and the evaluation process.

4.1 Datasets

The experiment was performed on openly-available imbalanced datasets obtained from KEEL and some of them are from UCI machine learning repository (UCI 2021). We have selected datasets from the KEEL repository and have screened datasets with the below selection criteria. Firstly, we excluded datasets containing categorical features, since the proposed SOBER method is designed for continuous features. From the remaining datasets, we choose those with a sample size of ≤ 2000 . Within these datasets, we gave preference to those exhibiting a wider range of imbalance levels, from low to high imbalance ratios. Through this process, we have selected a total of 18 datasets having sizes varying from 112 to 1599 and imbalance ratios (IR) ranging from 0.02 to 0.54. Dai et al. (2024) classified datasets into mildly, moderately, and highly imbalanced types. A dataset is considered mildly imbalanced if the IR is ≥ 0.2 . It is classified as moderately imbalanced when $0.2 > \text{IR} \geq 0.1$, and as highly imbalanced if the IR is < 0.1 . Based on these, 62% selected datasets have an $\text{IR} < 0.1$, indicating a high degree of class imbalance. Table 2 summarizes these datasets, including their respective imbalance ratios.

4.2 Evaluation metrics

Machine learning algorithms typically search for an optimal solution guided by an evaluation metric that decides the model parameters (Branco et al. 2016). In a classification problem with imbalanced dataset, the user prefers to have high performance on a poorly represented class. Usual model evaluation metric, such as accuracy, may impact user needs and generates a suboptimal classification model. Therefore, selection of evaluation metric is important while working with an imbalance classification problem.

A confusion matrix is shown in Table 3, for two class binary problems, consisting of True Negatives (TN), True Positives (TP), and cases with misclassification, such as False Positive (FP) and False Negative (FN). Here, the accuracy is defined as error rate and is calculated as $(TP + TN)/(TP + TN + FP + FN)$. Most classifiers perform well when there are approximately equal proportions of samples from both classes, resulting in a correct classification. In these cases, the average accuracy metric is sufficient to measure the performance of a classifier (Kononenko and Bratko 1991; Kubat et al. 1997; Kubat and Matwin 1997). However, accuracy metric is not suitable for imbalanced datasets. For example, in a problem where 5% of the samples belong to a minority class, an accuracy of 95% can be achieved by only predicting the majority class. This evaluation metric to search for an optimal solution is useless in practice because the user is more interested in the minority class.

Table 2 Openly-available imbalanced datasets used in this study, sorted by their imbalance ratio

Dataset	Acronym	No. Features	No. Minority (n^+)	No. Majority (n^-)	Imbalance Ratio $IR \leftarrow n^+/n^-$
Diabetes	D1	8	268	500	0.54
Wisconsin	D2	9	239	444	0.54
Haberman	D3	3	81	225	0.36
Vehicle	D4	18	212	634	0.33
Ecoli3	D5	7	35	301	0.12
Fertility	D6	9	12	100	0.12
Yeast-2_vs_4	D7	8	51	463	0.11
Climate	D8	20	46	494	0.09
Page-blocks-1-3_vs_4	D9	10	28	444	0.06
Oil	D10	48	41	896	0.05
Shuttle-6_vs_2-3	D11	9	10	220	0.05
Glass5	D12	9	9	205	0.04
Winequality-red-4	D13	11	53	1546	0.03
Yeast5	D14	8	44	1440	0.03
Winequality-red-8_vs_6	D15	11	18	637	0.03
Yeast6	D16	8	35	1449	0.02
Poker-8-9_vs_6	D17	10	25	1460	0.02
Winequality-white-3-9_vs_5	D18	11	25	1457	0.02

Table 3 Confusion matrix for binary classification

	Predicted (\hat{y})	
	$\hat{y}_i = c^+$	$\hat{y}_i = c^-$
True (y)	c^+ $TP = \sum_{i=1}^M \mathbb{I}_{c^+}(y_i) \mathbb{I}_{c^+}(\hat{y}_i)$ c^- $FP = n^- - TN$	$FN = n^+ - TP$ $TN = \sum_{i=1}^M \mathbb{I}_{c^-}(y_i) \mathbb{I}_{c^-}(\hat{y}_i)$

$\mathbb{I}_{c^+}(x)$ and $\mathbb{I}_{c^-}(x)$ are indicator functions that are equal to 1 when $x = c^+$ and $x = c^-$, respectively, and 0 otherwise

Therefore, a metric selection is vital for the imbalance classification problem and should consider user class preference and class distribution.

Different metrics are proposed, such as F_β , *geometric mean (GM)*, and *receiver operating characteristic (ROC)* curve. Each metric is derived from the confusion matrix as shown in Table 3. The completeness of the model is defined as *recall*, *sensitivity*, or *true positive rate (TPR)*, $TP/(TP + FN)$. In contrast, a model exactness is defined as *precision*, which is $TP/(TP + FP)$. Here, TP is a minority class that interests a user. In contrast to TPR, *false positive rate (FPR)* determines $FP/(FP + TN)$. In addition, *specificity* is defined as $TN/(TN + FP)$. A classification model needs both exactness and precision on an unseen dataset. Therefore, F_β , as shown in Eq. 6, is a combination of both, and β decides the trade-off between precision and recall. If an equal weightage ($\beta = 1$) is given to both precision and recall, then it is called as a F_1 measure.

$$F_{\beta} = \frac{(1 + \beta)^2 * recall * precision}{\beta^2 * precision + recall} \quad (6)$$

Likewise, *GM* computes the geometric mean by obtaining a good balance of the accuracies with both classes, namely specificity and sensitivity. It is calculated as shown in Eq. 7.

$$GM = \sqrt{\frac{TP}{TP + FN} * \frac{TN}{TN + FP}} = \sqrt{sensitivity * specificity} \quad (7)$$

In addition, *Matthew correlation coefficient* metric (*MCC*) is introduced, which is based on the Pearson correlation coefficient, which is calculated by Eq. 8 (Baldi et al. 2000; Gorodkin 2004). In this, \bar{y} and $\hat{\bar{y}}$ are the averages and σ_y and $\hat{\sigma}_y$ are standard deviations of actual and predicted class. This value ranges from +1, indicating a total agreement, and -1, which indicates complete disagreement between the actual and predicted class.

$$MCC = \sum_{i=1}^M \frac{(y_i - \bar{y})(\hat{y}_i - \hat{\bar{y}})}{\sigma_y * \hat{\sigma}_y} \quad (8)$$

In addition to numerical evaluation metrics, some graphical metrics are also proposed, such as the ROC curve (Fawcett 2006) and *precision-recall curve* (Davis and Goadrich 2006). ROC curve shows the relative importance of TPR and *false positive rate (FPR)* in a two-dimensional space. Each classifier provides a TPR-FPR pair, indicating a single point on the ROC curve. The primary objective in the ROC space is to have a solution in the upper left corner that corresponds to a better classifier than the others. ROC curve is translated to a single scalar value range between 0 to 1.0, which is called as *area under the ROC curve*. The ROC curve shows progress of classifying correctly true positive samples over misclassifying negative samples (Fawcett 2006).

GM, *MCC*, and *AUC* metrics evaluate the overall performance of a classifier because they aggregate accuracies from both the classes and F_{β} focuses on minority class performance. Therefore, these four evaluation metrics are considered while comparing pre-processing methods.

4.3 Classification algorithms

SOBER is evaluated using different classification algorithms to ensure its robustness across different classifiers. In the literature, there are many machine learning algorithms (Hastie et al. 2009), but we specifically chose RF and SVM classifiers (Breiman 2001; Cortes and Vapnik 1995). RF and SVM are widely recognized as standard benchmark classifiers in the machine learning literature. RF is particularly valued for its ability to handle high-dimensional and noisy data, while SVM is known for its effectiveness in handling nonlinear decision boundaries. RF classifier has initial tuning parameters, such as the number of trees in the forest set as 200, 'Gini' index as decision tree impurity measure, and the number of features for tree split as the square root of a dataset features. SVM classifier is configured with the kernel as a radial basis function.

4.4 Parameter settings

We have used various pre-processing methods to handle class imbalance problems. The pre-processing methods involve distribution change and data weighting, which are available in `Python imbalanced-learn` and `smote-variants` implementation (Lemaître et al. 2017; Kovács 2019). Among the undersampling methods, we selected RUS method, which randomly removes majority samples. We chose SMOTE, borderline-SMOTE, ADASYN, kmeans-SMOTE, and SVM-SMOTE methods from the interpolation methods. In addition, we have selected the top two algorithms, ProWSyn and polyfit-Mesh, from `smote-variants` and SMOTEWB as the latest ones from the package. Each oversampling method has a set of parameters that can be adjusted during an experiment. For all interpolation methods, the hyperparameter k for k -NN is selected from $k \in \{5, 7, 9, 11\}$ and for Borderline-SMOTE and SVM-SMOTE, the hyperparameter m for m -NN is selected from $m \in \{8, 10, 12\}$. A deformation factor for polyfit-mesh algorithm is chosen from $\{1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0\}$. In the case of ProWSyn, $\{1, 2, 3\}$ synthetic samples are generated for a minority sample. SMOTEWB takes a depth parameter from $\{3, 5, 7\}$ for AdaBoost algorithm. On the other hand, the weighting of a training dataset involves misclassification costs, where the cost is inversely proportional to respective class frequencies. All these methods are tested against *Baseline* method with no data pre-processing. The SOBER considers the hyperparameter k for k -NN from $k \in \{5, 7, 9, 11\}$ used in the objective function.

4.5 Cross validation and evaluation

The performance of all compared methods are evaluated through five repetitions to reduce randomness involving data split, thus confirming the generalizability of the proposed method. Each iteration consists of stratified 5-fold cross-validation. A dataset is partitioned into 80% training and the remaining 20% for testing, maintaining an equal proportion of class imbalance. Subsequently, the training dataset is further divided into 5 folds, ensuring a similar proportion of the class ratio in each fold. Each pre-processing method is used to balance an imbalanced training dataset, and then a classifier is trained and fitted using 4 of the folds, while the remaining fold is kept for validation to determine the best parameter combination. If the number of minority class instances in the training dataset is less than the values in the k -NN list $\{5, 7, 9, 11\}$, then Leave-One-Out cross-validation is used instead of 5-fold cross-validation.

The selection of the best parameter setting is based on the average F_1 score across the 5 folds or Leave-One-Out. The final best parameters of a pre-processing method are applied to the test dataset, and the performance is assessed using various evaluation metrics.

5 Results and discussion

The results presented in this section represent the average outcomes from five repetitions of an evaluation metric for each classifier and dataset. These findings provide readers with a general understanding of learning from imbalanced datasets. First, the statistical comparison of results is presented in Sect. 5.1, encompassing 10 pre-processing methods and

baseline method applied to 18 datasets using two machine learning classifiers. Second, Sect. 5.2 delves into the ranking of the compared methods. Third, secondary results in terms of Bayesian reinforcement are discussed in Sect. 5.3. Fourth, the effect of the class imbalance ratio is explored in Sect. 5.4. Finally, the computational complexity of SOBER is discussed in Sect. 5.5.

5.1 Statistical comparison

Tables 4 and 5 present the performance of RF and SVM classifiers, evaluated on all pre-processing methods, using F_1 , GM , MCC , and AUC as evaluation metrics. It is important to note that kmeans-SMOTE results are excluded due to difficulties encountered while training on highly imbalanced datasets. The best evaluation metric for each dataset is highlighted in bold. On Shuttle-6_vs_2-3 (D11) dataset, all compared methods exceeded except ADASYN, irrespective of the evaluation metrics and classifiers. Haberman (D3), Yeast-2_vs_4 (D7), Winequality-red-4 (D13), and Winequality-white-3-9_vs_5 (D18) datasets, our method exceeded on F_1 and MCC evaluation metrics using RF classifier. On Winequality-red-4 (D13) and Yeast5 (D14) datasets, our method outperformed on GM and AUC evaluation metrics using SVM classifier. RF classifier performed better on the Glass5 (D12) dataset across all evaluation metrics. In the case of Winequality-red-8_vs_6 (D15) dataset, our method exceeded on F_1 and GM evaluation metrics using RF and SVM classifiers, respectively. Our method outperformed on some evaluation metrics using RF classifier for the Winequality-white-3-9_vs_5 (D18), a highly imbalanced dataset. From these observations, our method outperformed using both classifiers with different evaluation metrics for highly imbalanced datasets.

However, it is worth mentioning that RUS method performs better using RF classifier on some of the datasets like, Diabetes (D1), Vehicle (D4), Yeast-2_vs_4 (D7), Climate (D8), Winequality-red-4 (D13), Yeast5 (D14), Winequality-red-8_vs_6 (D15), Yeast6 (D16), Winequality-white-3-9_vs_5 (D18). These results support previous research on class imbalance, where Tarawneh et al. (Tarawneh et al. 2022) emphasized that synthetically generated samples often fail to represent the characteristics of the original minority class accurately. It is important to note that these observations do not undermine the existing research, as the proposed method explores subspaces to generate minority samples. Based on empirical results, it can be concluded that RF classifier, with its default settings, is an effective algorithm for classifying imbalanced datasets across all evaluation metrics.

Overall, the results highlight that a classifier may perform best according to one metric but may not perform best when compared to other metrics. Hence, relying on a single evaluation metric is not sufficient for selecting the best classifier Ferri et al. (2009). To validate these significance of findings, it is crucial to evaluate the performance of classifiers on different datasets. However, since each dataset is unique, comparisons based on parametric tests are insufficient, which rely on assumptions of normality and homogeneity of variance. Therefore, we have used non-parametric tests, such as Kruskal-Wallis and Friedman, which are non-parametric counterparts of ANOVA, to determine if there are any significant differences between pre-processing methods. Kruskal-Wallis test compares the ranks of pre-processing methods instead of their averages, making it suitable for scenarios where

parametric test assumptions are not satisfied across multiple datasets. These statistical tests are accessed through `scikit-posthocs` Python package (Terpilowski 2019), while ensuring that all necessary assumptions for statistical inference are met.

The results of the statistical tests are presented in Tables 4 and 5. The symbol ‘*’ denotes methods that exhibit non-significant pairwise differences based on Dunn’s test within a particular dataset and metric, while demonstrating significant differences in method performance using Kruskal-Wallis test. The aggregate count of non-significant differences on Dunn’s test over each method across all datasets is defined with *performance score*. SOBER method appears at the top, Tables 4 and 5, with a performance score of 6 and 2 for RF and SVM classifiers, respectively.

5.2 Ranking of compared methods

Table 6 demonstrates varying differences in the average ranking with standard deviation of classifiers across different metrics. SOBER method comes in second position for F_1 and MCC evaluation metrics on RF classifier, while it stood in second position for GM evaluation metric on SVM classifier. ProWSyn method outperformed for GM , MCC , and AUC evaluation metrics on SVM classifier. The boldface shows the highest average rank across a metric and a classifier.

As we evaluated SOBER method on 18 datasets using 11 methods, we applied the Friedman test to rank pre-processing methods with an F-distribution of 119 degrees of freedom. The Friedman test yields a $p \leq 0.05$ for all evaluation metrics on both classifiers, leading to the rejection of the null hypothesis. Following the determination of differing average ranks, various procedures are employed to identify pre-processing methods that have different mean ranks compared to the others. Hence, the Nemenyi post-hoc test, similar to the Tukey test for ANOVA, is used to determine significant pairs (Demšar 2006). This test uses estimates of model performance and their ranking in each dataset. The results of these model comparisons are visualized using Critical Difference (CD) diagram depicted in Figs. 5 and 6 for RF and SVM classifiers, respectively. The vertical lines indicate the average rank of each pre-processing method across all 18 datasets. Connected methods do not show clear evidence of being different from each other. Conversely, methods that are not connected show statistically significant different ranks. For example, in Fig. 5, which presents the CD diagrams for RF classifier, statistically significant differences can be observed between SOBER method with Cost and RUS methods, using F_1 evaluation metric. In addition, we see statistically significant differences in ranks between SOBER with Cost, Baseline, SMOTEWB, and Borderline-SMOTE methods for GM evaluation metric with RF classifier. Similarly, for AUC evaluation metric with RF classifier, we see statistically significant differences in ranks between SOBER with tCost, Baseline, and SMOTEWB methods. Furthermore, in relation to the evaluation metric MCC , SOBER method is statistically significant compared to Cost method. In the case of SVM classifier, the CD diagrams reveal that SOBER method shows statistically significant differences compared to Baseline method in all evaluation metrics, as shown in Fig. 6.

Table 4 Comparison of RF classifier using pre-processing methods on 18 real-world datasets in terms of F_1 , GM , MCC , and AUC evaluation metrics

Dataset	Metric	Baseline	Cost	RUS	SMOTE	Borderline-SMOTE	ADASYN	SVM-SMOTE	polyfit-Mesh	ProWSyn	SMOTEWB	SOBER
D1	F1	0.744	0.755	0.753	0.761	0.743	0.742	0.749	0.739	0.75	0.767	0.751
	GM	0.728	0.740	0.770	0.767	0.750	0.749	0.759	0.746	0.754	0.765	0.753
	MCC	0.493	0.517	0.525	0.527	0.493	0.492	0.506	0.486	0.505	0.537	0.505
	AUC	0.739	0.749	0.772	0.768	0.751	0.751	0.760	0.748	0.756	0.769	0.756
D2	F1	0.973	0.974	0.976	0.976	0.971	0.973	0.976	0.975	0.973	0.975	0.973
	GM	0.977	0.977	0.981	0.980	0.975	0.978	0.982	0.979	0.978	0.978	0.978
	MCC	0.947	0.949	0.953	0.953	0.943	0.947	0.954	0.95	0.947	0.95	0.947
	AUC	0.977	0.977	0.981	0.980	0.975	0.978	0.982	0.979	0.978	0.978	0.978
D3	F1	0.542	0.518	0.576	0.545	0.571	0.558	0.565	0.604	0.553	0.599	0.612
	GM	0.469	0.421	0.625	0.562	0.589	0.578	0.573	0.638	0.594	0.51	0.625
	MCC	0.095	0.049	0.231	0.124	0.171	0.146	0.154	0.247	0.215	0.118	0.249
	AUC	0.544	0.519	0.631	0.569	0.595	0.582	0.585	0.639	0.615	0.56	0.633
D4	F1	0.685	0.668	0.741	0.740	0.732	0.733	0.715	0.751	0.74	0.728	0.734
	GM	0.614	0.594	0.792	0.746	0.741	0.745	0.702	0.778	0.738	0.701	0.731
	MCC	0.409	0.366	0.524	0.487	0.473	0.476	0.433	0.521	0.483	0.461	0.471
	AUC	0.668	0.654	0.794	0.753	0.748	0.751	0.717	0.781	0.747	0.721	0.741
D5	F1	0.827	0.751	0.693	0.769	0.742	0.766	0.780	0.782	0.765	0.784	0.772
	GM	0.765	0.680	0.851	0.822	0.778	0.833	0.823	0.852	0.821	0.767	0.846
	MCC	0.669	0.514	0.492	0.555	0.502	0.556	0.574	0.587	0.548	0.571	0.572
	AUC	0.792	0.726	0.857	0.829	0.799	0.838	0.834	0.856	0.827	0.791	0.851
D6	F1	0.679	0.606	0.472	0.672	0.660	0.741	0.663	0.643	0.659	0.656	0.633
	GM	0.424	0.283	0.456	0.471	0.465	0.663	0.465	0.416	0.42	0.42	0.416
	MCC	0.398	0.245	0.041	0.378	0.368	0.496	0.375	0.318	0.349	0.334	0.278
	AUC	0.644	0.589	0.533	0.678	0.672	0.772	0.678	0.628	0.639	0.633	0.622
D7	F1	0.818	0.771	0.826	0.824	0.838	0.817	0.820	0.829	0.836	0.791	0.842
	GM	0.732	0.664	0.853	0.802	0.786	0.767	0.793	0.845	0.837	0.717	0.826
	MCC	0.667	0.584	0.668	0.663	0.688	0.650	0.656	0.673	0.689	0.609	0.708
	AUC	0.776	0.725	0.863	0.824	0.811	0.799	0.814	0.857	0.851	0.76	0.845

Table 4 (continued)

Dataset	Metric	Baseline	Cost	RUS	SMOTE	Borderline-SMOTE	ADASYN	SVM-SMOTE	polyfit-Mesh	ProW'Syn	SMOTEWB	SOBER
D8	F1	0.495	0.477	0.590	0.641	0.628	0.624	0.676	0.648	0.62	0.475	0.637
	GM	0.066	0.000	0.802	0.574	0.540	0.540	0.575	0.605	0.548	0.000	0.571
	MCC	0.030	-0.012	0.361	0.288	0.259	0.255	0.371	0.306	0.234	-0.023	0.279
	AUC	0.508	0.498	0.809	0.656	0.635	0.634	0.660	0.674	0.63	0.494	0.643
D9	F1	0.982	0.979	0.810	0.969	0.990	0.990	1.000	0.979	0.979	0.969	0.992
	GM	0.981	0.963	0.960	0.946	0.983	0.963	0.963	0.983	1.000	0.946	0.999
	MCC	0.966	0.961	0.679	0.943	0.982	0.982	1.000	0.962	0.962	0.943	0.984
	AUC	0.982	0.967	0.961	0.950	0.983	0.983	1.000	0.967	0.967	0.95	0.999
D10	F1	0.649	0.644	0.558	0.760	0.733	0.766	0.755	0.754	0.76	0.699	0.739
	GM	0.441	0.441	0.695	0.664	0.628	0.664	0.663	0.665	0.664	0.515	0.625
	MCC	0.402	0.375	0.221	0.537	0.483	0.553	0.523	0.521	0.542	0.488	0.501
	AUC	0.599	0.598	0.711	0.720	0.695	0.721	0.719	0.719	0.72	0.637	0.695
D11	F1	1.000	1.000	1.000	1.000	1.000	0.800	1.000	1.000	1.000	1.000	1.000
	GM	1.000	1.000	1.000	1.000	1.000	0.800	1.000	1.000	1.000	1.000	1.000
	MCC	1.000	1.000	1.000	1.000	1.000	0.800	1.000	1.000	1.000	1.000	1.000
	AUC	1.000	1.000	1.000	1.000	1.000	0.800	1.000	1.000	1.000	1.000	1.000
D12	F1	0.808*	0.793*	0.585	0.853*	0.853*	0.821*	0.853*	0.853*	0.853*	0.853*	0.887*
	GM	0.759	0.758	0.870	0.873	0.873	0.815	0.873	0.873	0.873	0.873	0.932
	MCC	0.653	0.611	0.375	0.740	0.740	0.674	0.740	0.74	0.74	0.74	0.800
	AUC	0.793	0.790	0.880	0.890	0.890	0.840	0.890	0.890	0.890	0.890	0.940
D13	F1	0.491	0.491	0.482	0.537	0.542*	0.561*	0.528	0.554*	0.573*	0.495	0.619*
	GM	0.000	0.000	0.711	0.296	0.264	0.406	0.180	0.566	0.552	0.059	0.636
	MCC	-0.002	0.000	0.169*	0.076	0.093	0.125*	0.074	0.153*	0.175*	-0.002	0.264*
	AUC	0.500	0.500	0.714	0.545	0.537	0.570	0.522	0.63	0.635	0.501	0.687
D14	F1	0.830	0.883	0.779	0.888	0.899	0.908	0.891	0.903	0.887	0.879	0.880
	GM	0.765	0.827	0.977	0.901	0.912	0.936	0.923	0.935	0.912	0.841	0.946
	MCC	0.670	0.775	0.626	0.780	0.803	0.821	0.788	0.812	0.779	0.761	0.773
	AUC	0.797	0.843	0.977	0.907	0.919	0.940	0.928	0.94	0.917	0.853	0.948

Table 4 (continued)

Dataset	Metric	Baseline	Cost	RUS	SMOTE	Borderline-SMOTE	ADASYN	SVM-SMOTE	polyfit-Mesh	ProWSyn	SMOTEWB	SOBER
D15	F1	0.599	0.613	0.490	0.649	0.630	0.661	0.606	0.684	0.699	0.64	0.671
	GM	0.299	0.300	0.732	0.574	0.398	0.576	0.300	0.656	0.696	0.478	0.692
	MCC	0.235	0.296	0.185	0.325	0.308	0.346	0.262	0.377	0.412	0.307	0.360
	AUC	0.573	0.575	0.734	0.661	0.596	0.664	0.573	0.711	0.734	0.642	0.729
D16	F1	0.668	0.737	0.600	0.758	0.796	0.767	0.796	0.74	0.762	0.707	0.730
	GM	0.440	0.595	0.861	0.746	0.748	0.746	0.748	0.798	0.817	0.519	0.778
	MCC	0.376	0.524	0.337	0.525	0.602	0.542	0.602	0.5	0.54	0.438	0.477
	AUC	0.627	0.684	0.866	0.779	0.782	0.780	0.782	0.817	0.832	0.669	0.802
D17	F1	0.529	0.496	0.429	0.851	0.700	0.863	0.752	0.885	0.871	0.672	0.722
	GM	0.089	0.000	0.666	0.713	0.432	0.734	0.550	0.802	0.778	0.355	0.586
	MCC	0.089	0.000	0.093	0.712	0.430	0.733	0.548	0.8	0.776	0.354	0.476
	AUC	0.520	0.500	0.671	0.820	0.660	0.840	0.700	0.840	0.82	0.66	0.678
D18	F1	0.496*	0.496*	0.470	0.491	0.495*	0.491	0.494*	0.497	0.49	0.493*	0.559*
	GM	0.000	0.000	0.564	0.000	0.000	0.000	0.000	0.088	0.000	0.000	0.354
	MCC	0.000*	0.000*	0.070*	-0.018	-0.008*	-0.018	-0.010*	-0.001	-0.02*	-0.011*	0.121*
	AUC	0.500*	0.500*	0.616*	0.490	0.498	0.491	0.497	0.5	0.488	0.495	0.569*
PS		4	4	3	1	4	3	3	3	4	3	6

The boldface represents the highest metric for each corresponding dataset. The symbol '*' denotes methods that are statistically indistinguishable from each other ($p > 0.05$) as determined by Dunn's test, which is a post-hoc test used to detect significant differences following a Kruskal-Wallis test. The count of '*' for each method represents the performance score (PS)

Table 5 Comparison of SVM classifier using pre-processing methods on 18 real-world datasets in terms of F_1 , GM , MCC , and AUC evaluation metrics

Dataset	Metric	Baseline	Cost	RUS	SMOTE	Borderline-SMOTE	ADASYN	SVM-SMOTE	polyfit-Mesh	ProWSyn	SMOTEWB	SOBER
D1	F1	0.731	0.736	0.739	0.737	0.718	0.732	0.734	0.728	0.726	0.738	0.729
	GM	0.695	0.748	0.752	0.747	0.744	0.751	0.747	0.737	0.731	0.743	0.734
	MCC	0.483	0.480	0.487	0.481	0.475	0.482	0.478	0.462	0.455	0.479	0.461
	AUC	0.719	0.748	0.752	0.748	0.748	0.752	0.747	0.737	0.732	0.745	0.735
D2	F1	0.971	0.968	0.967	0.972	0.966	0.967	0.966	0.973	0.973	0.972	0.971
	GM	0.976	0.976	0.974	0.979	0.975	0.976	0.975	0.978	0.980	0.977	0.977
	MCC	0.944	0.938	0.935	0.945	0.934	0.936	0.934	0.947	0.948	0.944	0.944
	AUC	0.976	0.976	0.974	0.979	0.975	0.976	0.975	0.978	0.980	0.977	0.977
D3	F1	0.455	0.635	0.634	0.643	0.605	0.630	0.637	0.662	0.653	0.650	0.648
	GM	0.085	0.576	0.570	0.599	0.576	0.591	0.590	0.631	0.615	0.619	0.602
	MCC	0.046	0.279	0.289	0.290	0.233	0.262	0.281	0.327	0.31	0.303	0.302
	AUC	0.514	0.627	0.631	0.637	0.608	0.626	0.630	0.657	0.647	0.646	0.641
D4	F1	0.428	0.652	0.647	0.650	0.647	0.642	0.646	0.648	0.654	0.653	0.648
	GM	0.000	0.679	0.674	0.680	0.683	0.678	0.669	0.687	0.682	0.675	0.675
	MCC	0.000	0.326	0.317	0.326	0.327	0.318	0.311	0.332	0.331	0.324	0.319
	AUC	0.500	0.681	0.676	0.682	0.684	0.679	0.672	0.687	0.684	0.678	0.677
D5	F1	0.798	0.752	0.696	0.759	0.763	0.748	0.771	0.77	0.766	0.822	0.751
	GM	0.756	0.896	0.862	0.899	0.891	0.884	0.894	0.894	0.892	0.891	0.896
	MCC	0.601	0.578	0.506	0.588	0.586	0.563	0.599	0.597	0.59	0.664	0.576
	AUC	0.784	0.898	0.868	0.901	0.892	0.885	0.895	0.895	0.893	0.893	0.898
D6	F1	0.474	0.471	0.417	0.503	0.544	0.518	0.527	0.498	0.597	0.539	0.502
	GM	0.000	0.385	0.455	0.378	0.395	0.382	0.412	0.426	0.374	0.426	0.414
	MCC	0.000	0.045	0.013	0.050	0.112	0.077	0.215	0.121	0.041	0.129	0.079
	AUC	0.500	0.544	0.511	0.528	0.556	0.533	0.578	0.583	0.528	0.606	0.561
D7	F1	0.777	0.811	0.811	0.833	0.809	0.802	0.819	0.809	0.828	0.815	0.836
	GM	0.655	0.810	0.841	0.825	0.851	0.859	0.822	0.8	0.832	0.78	0.826
	MCC	0.611	0.635	0.637	0.686	0.637	0.625	0.653	0.642	0.680	0.655	0.685
	AUC	0.718	0.828	0.852	0.843	0.858	0.864	0.840	0.820	0.850	0.805	0.844

Table 5 (continued)

Dataset	Metric	Baseline	Cost	RUS	SMOTE	Borderline-SMOTE	ADASYN	SVM-SMOTE	polyfit-Mesh	ProWSyn	SMOTEWB	SOBER
D8	F1	0.478	0.533	0.517	0.528	0.543	0.520	0.571	0.537	0.535	0.544	0.536
	GM	0.000	0.746	0.737	0.741	0.749	0.734	0.733	0.75	0.748	0.625	0.749
	MCC	0.000	0.292	0.284	0.287	0.293	0.279	0.284	0.296	0.293	0.190	0.294
	AUC	0.500	0.760	0.755	0.756	0.758	0.749	0.740	0.763	0.761	0.652	0.762
D9	F1	0.748	0.688	0.661	0.689	0.696	0.683	0.700	0.686	0.69	0.696	0.675
	GM	0.591	0.735	0.697	0.754	0.755	0.752	0.757	0.753	0.754	0.756	0.749
	MCC	0.580	0.403	0.345	0.410	0.421	0.400	0.428	0.406	0.412	0.421	0.389
	AUC	0.683	0.762	0.724	0.775	0.777	0.773	0.779	0.774	0.775	0.777	0.769
D10	F1	0.489	0.554	0.542	0.545	0.541	0.544	0.552	0.544	0.543	0.579	0.543
	GM	0.000	0.472	0.418	0.489	0.463	0.508	0.375	0.524	0.524	0.433	0.463
	MCC	0.000	0.125	0.096	0.117	0.104	0.121	0.105	0.127	0.126	0.165	0.108
	AUC	0.500	0.582	0.562	0.585	0.577	0.593	0.554	0.602	0.602	0.581	0.576
D11	F1	1.000	1.000	1.000	1.000	1.000	0.800	1.000	1.000	1.000	1.000	1.000
	GM	1.000	1.000	1.000	1.000	1.000	0.800	1.000	1.000	1.000	1.000	1.000
	MCC	1.000	1.000	1.000	1.000	1.000	0.800	1.000	1.000	1.000	1.000	1.000
	AP	1.000	1.000	1.000	1.000	1.000	0.800	1.000	1.000	1.000	1.000	1.000
D12	F1	0.488	0.488	0.526	0.535	0.521	0.532	0.488	0.538	0.544	0.535	0.510
	GM	0.000	0.000	0.554	0.605	0.596	0.474	0.000	0.605	0.609	0.605	0.303
	MCC	0.000	0.000	0.152	0.193	0.177	0.175	0.000	0.196	0.203	0.193	0.084
	AUC	0.500	0.500	0.655	0.695	0.683	0.682	0.500	0.700	0.702	0.695	0.594
D13	F1	0.491	0.452	0.379	0.459	0.461	0.456	0.526	0.358	0.361	0.485	0.464
	GM	0.000	0.571*	0.520	0.594*	0.597*	0.574*	0.542*	0.584*	0.588*	0.214	0.613*
	MCC	0.000	0.066	0.033	0.083	0.085	0.070	0.110	0.076	0.078	0.014	0.100
	AUC	0.500	0.584*	0.544	0.605*	0.608*	0.588*	0.606*	0.603*	0.606*	0.524	0.626*
D14	F1	0.697	0.751	0.678	0.773	0.761	0.767	0.775	0.779	0.782	0.821	0.767
	GM	0.504	0.951	0.933	0.934	0.932	0.943	0.944	0.935	0.935	0.907	0.975
	MCC	0.478	0.574	0.469	0.598	0.579	0.594	0.605	0.606	0.612	0.664	0.607
	AUC	0.633	0.952	0.934	0.935	0.933	0.944	0.946	0.936	0.936	0.91	0.975

Table 5 (continued)

Dataset	Metric	Baseline	Cost	RUS	SMOTE	Borderline-SMOTE	ADASYN	SVM-SMOTE	polyfit-Mesh	ProWSyn	SMOTEWB	SOBER
D15	F1	0.492	0.401	0.382	0.428	0.486	0.423	0.489	0.425	0.424	0.46	0.440
	GM	0.000	0.505	0.460	0.529	0.578	0.525	0.581	0.568	0.565	0.339	0.601
	MCC	0.000	0.016	0.035	0.039	0.093	0.034	0.097	0.064	0.064	0.03	0.078
	AUC	0.500	0.524	0.557	0.555	0.616	0.549	0.616	0.591	0.59	0.551	0.606
D16	F1	0.494	0.658	0.610	0.669	0.685	0.645	0.697	0.646	0.655	0.757	0.645
	GM	0.000	0.889	0.870	0.833	0.821	0.858	0.823	0.843	0.859	0.761	0.871
	MCC	0.000	0.416	0.350	0.406	0.423	0.384	0.442	0.379	0.399	0.519	0.390
	AUC	0.500	0.892	0.874	0.843	0.834	0.863	0.836	0.851	0.866	0.792	0.876
D17	F1	0.496	0.531	0.345	0.523	0.529	0.528	0.571	0.502	0.504	0.512	0.547
	GM	0.000	0.442	0.369	0.380	0.358	0.490	0.339	0.466	0.547	0.350	0.567
	MCC	0.000	0.105	-0.023	0.094	0.091	0.119	0.149	0.103	0.115	0.071	0.144
	AUC	0.500	0.609	0.460	0.595	0.591	0.632	0.591	0.651	0.668	0.584	0.647
D18	F1	0.496	0.466	0.509	0.441	0.489	0.428	0.500	0.456	0.518	0.476	0.497
	GM	0.000	0.416	0.441	0.456	0.118	0.415	0.087	0.414	0.447	0.375	0.437
	MCC	0.000	0.031	0.076	0.028	-0.001	0.009	0.005	0.02	0.088	0.046	0.060
	AUC	0.500	0.545	0.591	0.546	0.510	0.514	0.509	0.532	0.595	0.053	0.580
PS		0	2	0	2	2	2	2	2	2	0	2

The boldface represents the highest metric for each corresponding dataset. The symbol '**' denotes methods that are statistically indistinguishable from each other ($p > 0.05$) as determined by Dunn's test, which is a post-hoc test used to detect significant differences following a Kruskal-Wallis test. The count of '**' for each method represents the performance score (PS)

Table 6 Results for mean and standard deviation ranking of pre-processing methods across the datasets

ML Classifier	Metric	Baseline	Cost	RUS	SMOTE	Borderline-SMOTE	ADASYN	SVM-SMOTE	polyfit-Mesh	ProWSyn	SMOTEB	SOBER
RF	F1	7.72 ± 2.98	2.28 ± 2.44	2.44 ± 2.29	5.72 ± 2.56	5.56 ± 3.21	4.81 ± 2.63	4.08 ± 2.54	4.08 ± 2.63	4.81 ± 2.47	4.00 ± 2.62	4.44 ± 2.84
	GM	9.28 ± 1.95	9.81 ± 1.54	2.69 ± 2.62	5.33 ± 2.47	6.42 ± 2.97	5.75 ± 2.64	5.33 ± 2.44	3.83 ± 2.65	5.06 ± 1.83	8.06 ± 2.04	4.44 ± 2.40
	MCC	7.72 ± 3.10	8.33 ± 2.47	6.86 ± 4.18	5.25 ± 2.44	5.81 ± 2.54	5.61 ± 3.16	4.72 ± 2.71	4.44 ± 2.60	5.22 ± 2.80	7.47 ± 2.38	4.56 ± 2.85
AUC		8.97 ± 2.28	9.50 ± 1.89	3.47 ± 3.34	5.42 ± 2.60	6.33 ± 1.97	5.61 ± 2.95	5.03 ± 2.58	4.08 ± 2.41	5.25 ± 2.44	8.03 ± 2.28	4.31 ± 2.33
	F1	7.44 ± 3.86	6.64 ± 2.48	8.31 ± 3.07	5.19 ± 1.98	6.39 ± 2.92	8.00 ± 2.09	4.06 ± 3.08	5.81 ± 2.88	4.97 ± 3.16	3.25 ± 1.93	5.94 ± 2.48
	GM	10.42 ± 1.56	5.78 ± 2.54	6.75 ± 3.27	4.75 ± 2.12	5.83 ± 2.76	5.83 ± 2.53	6.94 ± 2.96	4.28 ± 2.48	4.19 ± 2.56	6.94 ± 3.14	4.28 ± 2.72
MCC		8.61 ± 3.63	6.83 ± 2.07	7.97 ± 2.97	5.08 ± 1.53	6.17 ± 3.00	7.50 ± 2.23	5.33 ± 3.48	4.42 ± 2.65	4.14 ± 2.71	4.67 ± 3.12	5.28 ± 2.82
	AUC	10.36 ± 1.55	5.89 ± 2.68	7.08 ± 3.15	5.11 ± 1.82	5.81 ± 2.93	6.22 ± 2.67	6.47 ± 3.04	4.11 ± 2.54	3.75 ± 2.61	6.64 ± 3.00	4.56 ± 2.64

Boldface indicates the highest average rank within each metric and classifier combination

5.3 Effect of Bayesian reinforcement

The effect of Bayesian reinforcement for L -subspace feature selection is shown in Fig. 7. Yeast5 dataset is selected to show Bayesian reinforcement of L -subspaces on a selected repetition of the RF classifier. The initial dominant L -subspaces are simply due to chance. After each generation of a new minority sample, optimized objective values are obtained to reinforce L -subspaces, which may select L -subspaces having higher probabilities to distinguish minority and majority samples using the Bayesian Dirichlet distribution.

5.4 Effect of imbalance ratio

The performance of pre-processing methods varies with the amount of class imbalance ratio in a dataset. To validate robustness of the proposed method, Fig. 8 and Fig. 9 show the mean of each evaluation metric, using both classifiers over the 18 openly-available imbalanced datasets. It includes five comparisons from all to 5%: the former consists of all datasets, while the latter considers the most seven imbalanced datasets with an imbalanced ratio of 5%.

Figure 8 shows that, regardless of the imbalanced ratio, our method performed better and stood in second position using RF classifier with all evaluation metrics. For SVM classifier, our method stood in second position for all the imbalance ratios with GM evaluation metric, as shown in Fig. 9. In addition, for highly imbalanced datasets with 5% imbalance ratio, our method outperformed the rest of pre-processing methods and ranked third in terms of F_1 evaluation metric.

5.5 Computational complexity

SOBER time complexity depends on the number of minority samples required to generate using Nelder-Mead optimization with feature subspaces. In the worst case scenario, the L -subspace requires $\binom{N}{3}$, expressed as $\mathcal{O}(N^3)$. Therefore, for n_s minority sample generations, the L -subspace complexity is $\mathcal{O}(n_s * N^3)$. After selecting a L -subspace, each Nelder-Mead optimization performs three operations, each with t iterations. During a particular iteration of an optimization, it computes k -NN from M observations on the subspace having d dimensions, with a $\mathcal{O}(d * M * \log(M))$ complexity. Here, the maximum dimensions of the L -subspace (d) are assumed to be constant at 3. In the worst case, Nelder-Mead takes $\mathcal{O}(n_s * t * M * \log(M))$ time for generating minority samples. Therefore, the overall time complexity is dominated by Nelder-Mead and iterating over L -subspace possibilities, as given by Eq. 9.

$$\mathcal{O}(n_s * N^3 + n_s * t * M * \log(M)) = \mathcal{O}(n_s(N^3 + t * M * \log(M))) \quad (9)$$

6 Conclusions

We have proposed a novel oversampling method, named SOBER, which uses feature subspaces (feature-sets) to generate minority samples by using nonlinear optimization of a class-sensitive objective function. SOBER uses objective function values from optimization

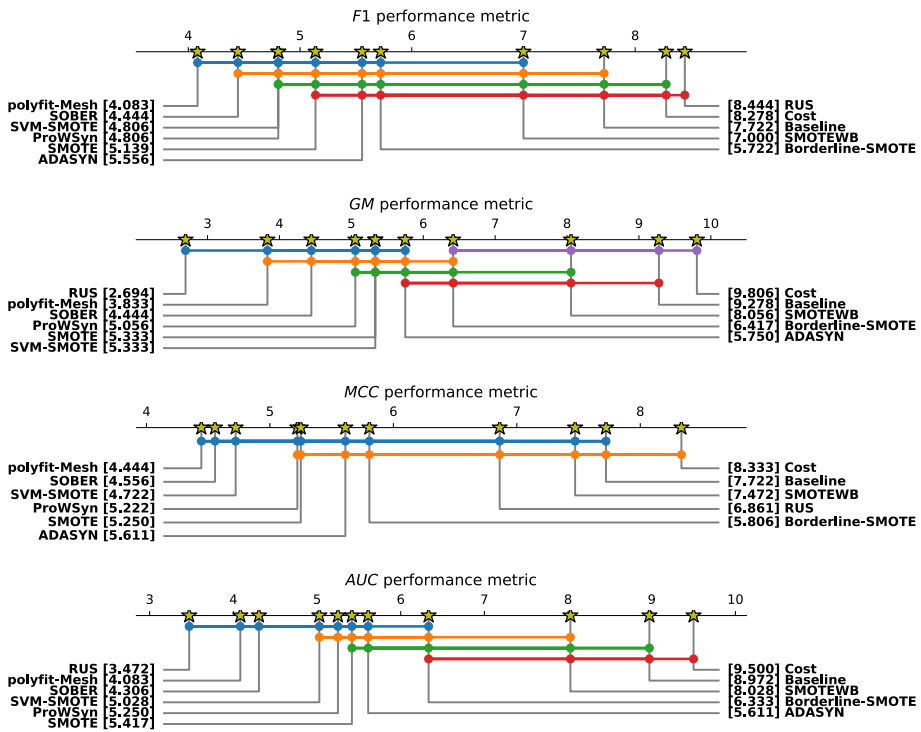


Fig. 5 Critical Difference (CD) diagrams demonstrating the results of the Nemenyi post-hoc test for RF classifier with each evaluation metric exhibiting $p \leq 0.05$. The performance of pre-processing methods is significantly different if the corresponding average ranks differ by at least the critical difference. Groups of pre-processing methods that are not significantly different (with a $p > 0.05$) are connected

to reinforce those subspaces that can distinguish between minority and majority samples. The reinforcement of subspaces is performed with the help of the Bayesian reinforcement strategy with Dirichlet distribution.

Our empirical studies on real-world datasets have shown that SOBER outperforms existing methods with varying class distributions and overlap, as evidenced by the performance score and mean ranking from statistical comparisons. SOBER performance score is at the highest for RF and SVM classifiers. Furthermore, it performs well when measured over the mean ranking of evaluation metrics for highly imbalanced datasets for at least two evaluation metrics on both classifiers. Furthermore, no single method outperforms all evaluation metrics for both classifiers.

Additionally, SOBER provides insights into the importance of subspaces. The proposed method uses the objective value from each generation of minority samples and continuously updates the probability of feature subspaces. The feature subspaces are useful to distinguish between minority and majority samples over the entire space, and there is a possibility that more than one of these subspaces are important in the given imbalanced dataset. Overall, a few of the top subspaces give the most important features that are effective in distinguishing between minority and majority samples.

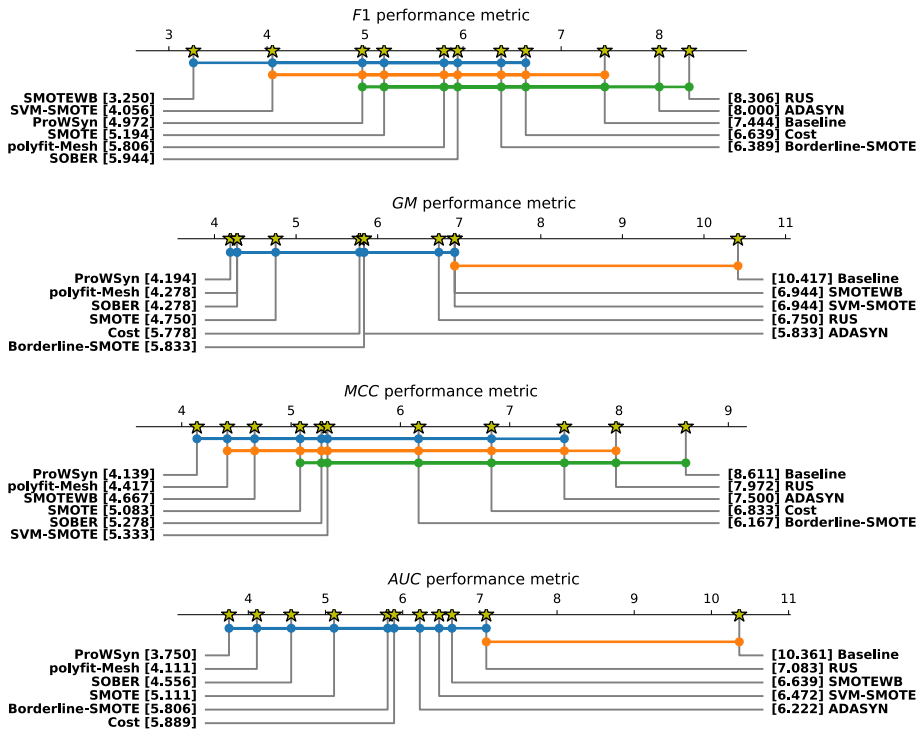


Fig. 6 Critical Difference (CD) diagrams demonstrating the results of the Nemenyi post-hoc test for SVM classifier with each evaluation metric exhibiting $p \leq 0.05$. The performance of pre-processing methods is significantly different if the corresponding average ranks differ by at least the critical difference. Groups of pre-processing methods that are not significantly different (with $p > 0.05$) are connected

SOBER has some limitations that future works can address. The existing nonlinear optimization algorithm considers continuous features, and having a heterogeneous distance could help to measure heterogeneous features similarity between samples. Additionally, the optimization process takes time to converge for each sample generation, and it may be more efficient to train and evaluate the classifiers on partially balanced data distributions rather than balancing the entire distribution. The proposed method can be improved by finding border samples on feature subspaces and then applying SOBER for generating synthetic minority samples. SOBER uses a specific objective function, and exploring new objective functions that target class boundaries may be helpful in generating representative minority samples. Finally, The SOBER can be extended for multi-class classification. This can be achieved using the One-vs-All (OVA) approach. In a dataset, those with frequencies lower than the most represented class are selected for targeted minority classes. For each such class, the data is relabeled by treating it as the minority class, while all remaining classes are grouped as the majority class. Oversampling is then applied to balance the distribution, and then perform classification. This enables focused enhancement of minority class representation within the OVA framework, improving classifier performance on underrepresented categories.

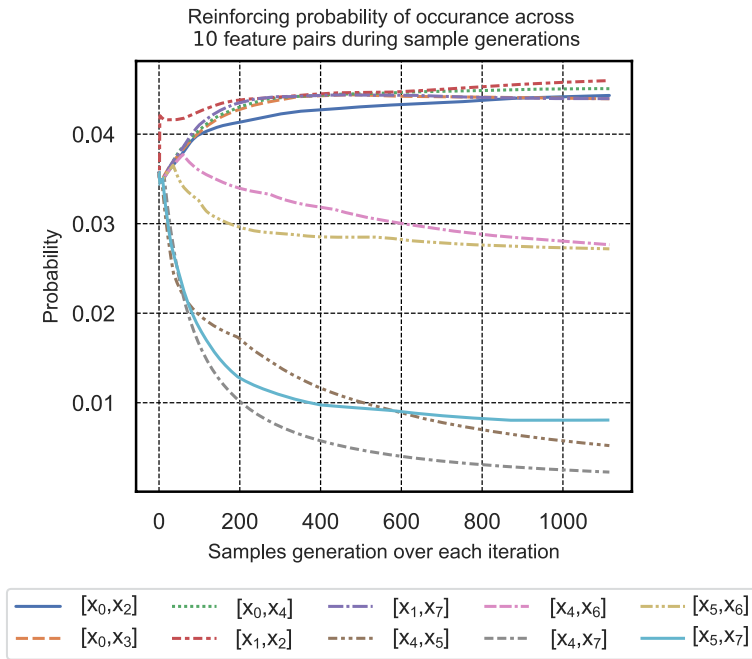


Fig. 7 Yeast5: Effect of Bayesian reinforcement on a selected repetition of RF classifier

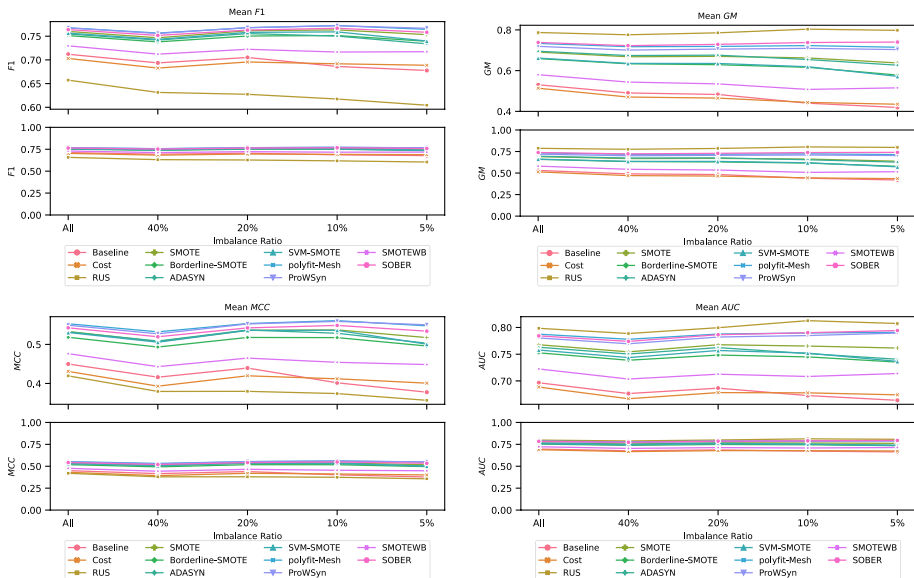


Fig. 8 Mean of evaluation metrics for RF classifier as a function of the imbalance ratio. Each metric is represented by two plots, where the y-axis of the top plot is zoomed in, and the bottom plot's y-axis ranges from 0 to 1. 'All' category represents the evaluation metric calculated across all 18 datasets. Additionally, for imbalance ratios of 40%, 20%, 10%, and 5%, metrics are derived from 16, 14, 11, and 7 datasets, respectively

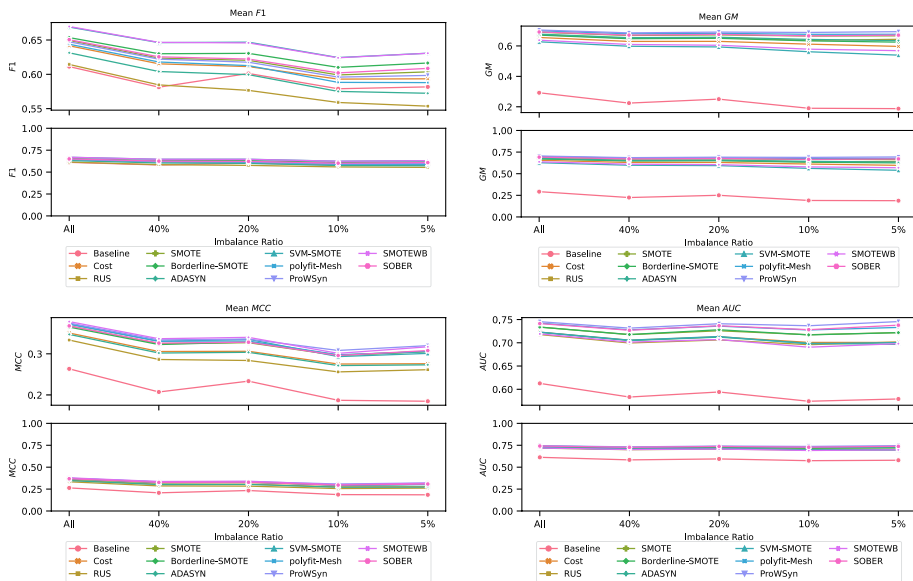


Fig. 9 Mean of evaluation metrics for SVM classifier as a function of the imbalance ratio. Each metric is represented by two plots, where the y-axis of the top plot is zoomed in, and the bottom plot's y-axis ranges from 0 to 1. 'All' category represents the evaluation metric calculated across all 18 datasets. Additionally, for imbalance ratios of 40%, 20%, 10%, and 5%, metrics are derived from 16, 14, 11, and 7 datasets, respectively

Acknowledgements The authors acknowledge the financial support received from KK-stiftelsen (The Knowledge Foundation, Stockholm, Sweden) and VINNOVA (Sweden Innovation Agency, Stockholm, Sweden) for the research projects 'TOPAZ - Towards Prescriptive Analytics in Virtual Factories through Structured Data Mining and Optimization' under grant 20200011 and 'Integrated Manufacturing Analytics Platform for Predictive Maintenance with IoT' under grant 2021-02537.

Author contributions Kumbhar Mahesh: Methodology, Software, Investigation, Validation, Formal analysis, Writing—Original draft. Bandaru Sunith: Conceptualization, Methodology, Writing—Reviewing and Editing. Karlsson Alexander: Conceptualization, Methodology, Writing—Original draft, Writing—Reviewing and Editing.

Funding Open access funding provided by University of Skövde.

Data availability This study is conducted using benchmark public datasets from repositories <http://www.kee1.es/> and <https://archive.ics.uci.edu/datasets>.

Declarations

Conflict of interest The authors declare no Conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Aggarwal CC, Hinneburg A, Keim DA (2001) On the surprising behavior of distance metrics in high dimensional space. In: International Conference on Database Theory, pp. 420–434. https://doi.org/10.1007/3-540-44503-X_27
- Alcalá-Fdez J, Fernández A, Luengo J, Derrac J, García S, Sánchez L, Herrera F (2011) Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *J Multiple Valued Logic Soft Comput* 17:255–287
- Baldi P, Brunak S, Chauvin Y, Andersen CAF, Nielsen H (2000) Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics* 16:412–424. <https://doi.org/10.1093/bioinformatics/16.5.412>
- Barua S, Islam MM, Yao X, Murase K (2014) Mwmote-majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Trans Knowl Data Eng* 26:405–425. <https://doi.org/10.1109/TKDE.2012.232>
- Barua S, Islam MM, Murase K (2013) Prowsyn: Proximity weighted synthetic oversampling technique for imbalanced data set learning. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 317–328. Springer
- Batista GEAPA, Prati RC, Monard MC (2004) A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsl* 6(1):20–29. <https://doi.org/10.1145/1007730.1007735>
- Branco P, Torgo L, Ribeiro RP (2016) A survey of predictive modeling on imbalanced domains. *ACM Comput Surv* 49:1–50. <https://doi.org/10.1145/2907070>
- Breiman L (2001) Random forests *Machine learning* 45:5–32. <https://doi.org/10.1023/A:1010933404324>
- Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) Smote: synthetic minority over-sampling technique. *J Artif Intell Res* 16:321–357. <https://doi.org/10.1613/jair.953>
- Cortes C, Vapnik V (1995) Support-vector networks *Machine learning* 20:273–297. <https://doi.org/10.1007/BF00994018>
- Dai Q, Liu J-W, Wang L-H (2024) Imbalanced instance selection based on Laplacian matrix decomposition with weighted k-nearest-neighbor graph. *Neural Comput Appl* 36(20):12397–12425
- Dai Q, Wang L, Zhang J, Ding W, Chen L (2025) Gqeo: nearest neighbor graph-based generalized quadrilateral element oversampling for class-imbalance problem. *Neural Netw* 184:107107
- Dangut MD, Skaf Z, Jennions IK (2022) Handling imbalanced data for aircraft predictive maintenance using the Bache algorithm. *Appl Soft Comput* 123:108924. <https://doi.org/10.1016/j.asoc.2022.108924>
- Davis J, Goadrich M (2006) The relationship between precision-recall and roc curves. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 233–240. <https://doi.org/10.1145/1143844.1143874>
- Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. *J Mach Learn Res* 7:1–30
- Depto DS, Rizvee MM, Rahman A, Zunair H, Rahman MS, Mahdy MRC (2023) Quantifying imbalanced classification methods for leukemia detection. *Comput Biol Med* 152:106372. <https://doi.org/10.1016/j.combiomed.2022.106372>
- Douzas G, Bacao F, Last F (2018) Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. *Inf Sci* 465:1–20. <https://doi.org/10.1016/j.ins.2018.06.056>
- Fawcett T (2006) An introduction to roc analysis. *Pattern Recogn Lett* 27:861–874. <https://doi.org/10.1016/j.patrec.2005.10.010>
- Fernández A, García S, Herrera F, Chawla NV (2018) Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *J Artif Intell Res* 61:863–905
- Ferri C, Hernández-Orallo J, Modroiu R (2009) An experimental comparison of performance measures for classification. *Pattern Recogn Lett* 30(1):27–38. <https://doi.org/10.1016/j.patrec.2008.08.010>
- Gao F, Han L (2012) Implementing the Nelder–Mead simplex algorithm with adaptive parameters. *Comput Optim Appl* 51:259–277. <https://doi.org/10.1007/s10589-010-9329-3>
- García S, Cano JR, Fernández A, Herrera F (2006) A proposal of evolutionary prototype selection for class imbalance problems. In: International Conference on Intelligent Data Engineering and Automated Learning, pp. 1415–1423. https://doi.org/10.1007/11875581_168. Springer
- Gazzah S, Amara NEB (2008) New oversampling approaches based on polynomial fitting for imbalanced data sets. In: 2008 the Eighth Iapri International Workshop on Document Analysis Systems, pp. 677–684. IEEE
- Gelman A, Carlin JB, Stern HS, Dunson DB, Vehtari A, Rubin DB (2013) Bayesian Data Analysis. Chapman and Hall/CRC, New York. <https://doi.org/10.1201/b16018>
- Gorodkin J (2004) Comparing two k-category assignments by a k-category correlation coefficient. *Comput Biol Chem* 28:367–374. <https://doi.org/10.1016/j.compbiolchem.2004.09.006>

- Guyon I, De AM (2003) An introduction to variable and feature selection. *J Mach Learn Res* 3:1157–1182. <https://doi.org/10.1162/15324430322753616>
- Han H, Wang W-Y, Mao B-H (2005) Borderline-smote: a new over-sampling method in imbalanced data sets learning. *Borderline-SMOTE A New Over-Sampling Method in Imbalanced Data Sets Learning* 3644:878–887. https://doi.org/10.1007/11538059_91
- Hastie T, Tibshirani R, Friedman JH, Friedman JH (2009) *The elements of statistical learning: data mining, inference, and prediction*, vol 2. Springer, New York
- He H, Bai Y, Garcia EA, Li S (2008) Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In: *IEEE International Joint Conference on Neural Networks*, pp. 1322–1328. <https://doi.org/10.1109/IJCNN.2008.4633969>
- Hu S, Liang Y, Ma L, He Y (2009) Msmote: Improving classification performance when training data is imbalanced. *Int Workshop Comput Sci Eng* 3:13–17. <https://doi.org/10.1109/WCSE.2009.756>
- Islam A, Belhaouari SB, Rehman AU, Bensmail H (2022) Knnor: an oversampling technique for imbalanced datasets. *Appl Soft Comput* 115:108288. <https://doi.org/10.1016/j.asoc.2021.108288>
- Jo T, Japkowicz N (2004) Class imbalances versus small disjuncts. *ACM SIGKDD Explorations Newsl* 6:40–49. <https://doi.org/10.1145/1007730.1007737>
- Khoshgoftaar TM, Seiffert C, Hulse JV, Napolitano A, Folleco A (2007) Learning with limited minority class data. In: *International Conference on Machine Learning and Applications*, pp. 348–353. <https://doi.org/10.1109/ICMLA.2007.76>
- Kononenko I, Bratko I (1991) Information-based evaluation criterion for classifier's performance. *Mach Learn* 6:67–80. <https://doi.org/10.1023/A:1022642017308>
- Kovács G (2019) smote-variants: a python implementation of 85 minority oversampling techniques. *Neuro-computing* 366:352–354. <https://doi.org/10.1016/j.neucom.2019.06.100>
- Krawczyk B (2016) Learning from imbalanced data: open challenges and future directions. *Progress Artif Intell* 5:221–232. <https://doi.org/10.1007/s13748-016-0094-0>
- Kubat M, Holte R, Matwin S (1997) Learning when negative examples abound. In: *European Conference on Machine Learning*, pp. 146–153. https://doi.org/10.1007/3-540-62858-4_79
- Kubat M, Matwin S (1997) Addressing the curse of imbalanced training sets : One-sided selection, pp. 179–186. Morgan Kaufmann. <https://cir.nii.ac.jp/crid/1571135650135717888>
- Kumar A, Singh D, Shankar Yadav R (2024) Class overlap handling methods in imbalanced domain: a comprehensive survey. *Multimed Tools Appl* 83(23):63243–63290
- Kumar A, Singh D, Yadav RS (2024) Entropy-based hybrid sampling (ehs) method to handle class overlap in highly imbalanced dataset. *Expert Syst* 41(11):13679
- Lemaître G, Nogueira F, Aridas CK (2017) Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning. *J Mach Learn Res* 18(17):1–5
- Liang P, Liu G, Xiong Z, Fan H, Zhu H, Zhang X (2022) A fault detection model for edge computing security using imbalanced classification. *J Syst Archit*. <https://doi.org/10.1016/j.sysarc.2022.102779>
- Li D, Zheng C, Zhao J, Liu Y (2023) Diagnosis of heart failure from imbalance datasets using multi-level classification. *Biomed Signal Process Control*. <https://doi.org/10.1016/j.bspc.2022.104538>
- Mani I, Zhang J (2003) knn approach to unbalanced data distributions: a case study involving information extraction. *Proc Workshop Learn Imbalanced Datasets* 126:1–7
- Menardi G, Torelli N (2014) Training and assessing classification rules with imbalanced data. *Data Min Knowl Disc* 28:92–122. <https://doi.org/10.1007/s10618-012-0295-5>
- Napierała K, Stefanowski J, Wilk S (2010) Learning from imbalanced data in presence of noisy and borderline examples. In: *Rough Sets and Current Trends in Computing*, pp. 158–167. https://doi.org/10.1007/978-3-642-13529-3_18
- Nelder JA, Mead R (1965) A simplex method for function minimization. *Comput J* 7:308–313. <https://doi.org/10.1093/comjnl/7.4.308>
- Nguyen HM, Cooper EW, Kamei K (2011) Borderline over-sampling for imbalanced data classification. *Int J Knowledge Eng Soft Data Paradigms* 3:4. <https://doi.org/10.1504/IJKESDP.2011.039875>
- Pradipta GA, Wardoyo R, Musdholifah A, Sanjaya INH (2021) Radius-smote: a new oversampling technique of minority samples based on radius distance for learning from imbalanced data. *IEEE Access* 9:74763–74777. <https://doi.org/10.1109/ACCESS.2021.3080316>
- Sağlam F, Cengiz MA (2022) A novel smote-based resampling technique through noise detection and the boosting procedure. *Expert Syst Appl* 200:117023
- Shahee SA, Ananthakumar U (2020) An effective distance based feature selection approach for imbalanced data. *Appl Intell* 50:717–745. <https://doi.org/10.1007/s10489-019-01543-z>
- Stefanowski J (2013) Overlapping, rare examples and class decomposition in learning classifiers from imbalanced data. *Emerg Paradigms Mach Learn*. https://doi.org/10.1007/978-3-642-28699-5_11
- Tarawneh AS, Hassanat AB, Altarawneh GA, Almuhaimeed A (2022) Stop oversampling for class imbalance learning: A review. *IEEE Access* 10:47643–47660. <https://doi.org/10.1109/ACCESS.2022.3169512>

- Terpilowski M (2019) Scikit–Posthocs: pairwise multiple comparison tests in python. *J Open Sourc Softw* 4(36):1169. <https://doi.org/10.21105/joss.01169>
- Tomek I (1976) Two modifications of cnn. *IEEE Transactions on Systems, Man, and Cybernetics SMC-6*, 769–772 <https://doi.org/10.1109/TSMC.1976.4309452>
- UCI Machine Learning Repository. <http://archive.ics.uci.edu/ml>. Accessed: May 10, 2023 (2021)
- Zhang J, Wang T, Ng WW, Pedrycz W (2023) Perturbation-based oversampling technique for imbalanced classification problems. *Int J Mach Learn Cybern* 14(3):773–787

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Mahesh Kumbhar¹ · Sunith Bandaru¹ · Alexander Karlsson²

✉ Mahesh Kumbhar
mahesh.kumbhar@his.se

Sunith Bandaru
sunith.bandaru@his.se

Alexander Karlsson
alexander.karlsson@his.se

¹ Intelligent Production Systems, School of Engineering Science, University of Skövde, 541 28 Skövde, Sweden

² Skövde Artificial Intelligence Lab, School of Informatics, University of Skövde, 541 28 Skövde, Sweden