



Noise-free sampling with majority framework for an imbalanced classification problem

Neni Alya Firdausanti¹ · Israel Mendonça² · Masayoshi Aritsugi²

Received: 8 September 2023 / Revised: 11 January 2024 / Accepted: 15 February 2024 /
Published online: 9 April 2024

© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2024

Abstract

Class imbalance has been widely accepted as a significant factor that negatively impacts a machine learning classifier's performance. One of the techniques to avoid this problem is to balance the data distribution by using sampling-based approaches, in which synthetic data is generated using the probability distribution of the classes. However, this process is sensitive to the presence of noise in the data, and the boundaries between the majority class and the minority class are blurred. Such phenomena shift the algorithm's decision boundary away from the ideal outcome. In this work, we propose a hybrid framework for two primary objectives. The first objective is to address class distribution imbalance by synthetically increasing the data of a minority class, and the second objective is, to devise an efficient noise reduction technique that improves the class balance algorithm. The proposed framework focuses on removing noisy elements from the majority class, and by doing so, provides more accurate information to the subsequent synthetic data generator algorithm. To evaluate the effectiveness of our framework, we employ the geometric mean (*G*-mean) as the evaluation metric. The experimental results show that our framework is capable of improving the prediction *G*-mean for eight classifiers across eleven datasets. The range of improvements varies from 7.78% on the Loan dataset to 67.45% on the Abalone19_vs_10-11-12-13 dataset.

Keywords Imbalance · Classification · Machine learning · Noisy data · Synthetic oversampling

Israel Mendonça and Masayoshi Aritsugi have contributed equally to this work.

✉ Neni Alya Firdausanti
nenialya@dbms.cs.kumamoto-u.ac.jp

Israel Mendonça
israel@cs.kumamoto-u.ac.jp

Masayoshi Aritsugi
aritsugi@cs.kumamoto-u.ac.jp

¹ Graduate School of Science and Technology, Kumamoto University, Kumamoto, Japan

² Faculty of Advanced Science and Technology, Kumamoto University, Kumamoto, Japan

1 Introduction

Class imbalance is widely acknowledged as a significant factor that interferes with the effectiveness of classification models [1–3]. In an imbalanced dataset, the amount of data belonging to a class (namely, the majority or negative class) significantly exceeds that of another class (namely, the minority or positive class) [4, 5]. When data is distributed unevenly across different classes, it becomes challenging for a classification model to accurately predict outcomes for the minority classes [6]. This, in turn, usually results in various costs associated with misclassified data. The challenging feature of imbalanced data is apparent in numerous significant classification domains where the target variable predominantly belongs to the minority class, including fraud detection [7, 8], enterprise credit evaluation [9], disease diagnosis [10, 11], image recognition [12, 13], and failure prediction [14, 15]. With this type of data, it becomes challenging to overcome the issue of class imbalance due to obstacles in acquiring balanced data, such as cost, confidentiality, and effort [1], thus encouraging researchers to develop different approaches aimed at mitigating the degradation of the classification model performance caused by class imbalance.

Over the past few decades, numerous approaches have been proposed to address the class imbalance problem. These methods are typically classified into three primary categories: (i) sampling-based approaches, (ii) algorithm-based approaches, and (iii) cost-sensitive approaches [3, 5]. Sampling-based approaches involve a preprocessing stage aimed at balancing the class distribution. The purpose of algorithm-based approaches is to adjust classification algorithms, allowing them to address the imbalance issue by favoring the discrimination process toward the minority class. Finally, cost-sensitive approaches integrate strategies in both the sampling-based approach and algorithm-level approach, allowing for varying costs associated with various misclassification categories, and assigning higher costs for misclassifying the minority class. The most extensively studied category of solutions is the sampling-based approach due to its easy implementation [2, 3, 16]. In particular, this approach aims to preprocess data by balancing the distribution between the majority and minority classes. Subsequently, the balanced dataset can be utilized as input for any traditional classification method.

Sampling-based approaches can be divided into three categories: oversampling, undersampling, and hybrid methods that combine both approaches. Undersampling methods aim to achieve a balanced class distribution by reducing the data of the majority class. On the other hand, oversampling methods generate new data that resembles those of the minority class [17]. However, these methods have their drawbacks, as they involve artificially altering the initial class distribution. For example, undersampling may remove valuable data, leading to information loss and possibly increasing the classifier's variance. In contrast, oversampling creates synthetic data to expand the dataset, which increases the risk of overfitting and places a greater computational load on any learning model [18, 19]. Although oversampling and undersampling demonstrate comparable effectiveness when applied to moderately imbalanced data, oversampling is more commonly utilized than undersampling [20]. One of the reasons is that oversampling outperforms undersampling when the data is significantly imbalanced data [21, 22].

Combining undersampling and oversampling methods enables the effective utilization of datasets with extremely imbalanced cases [23, 24]. By employing combination sampling methods, the drawbacks of undersampling, especially when dealing with a small amount of minority class data, can be mitigated. In noteworthy research conducted by [25], a synthetic oversampling framework known as SWIM (sampling with the majority class) was intro-

duced. The SWIM framework has proven to be effective in handling extremely imbalanced problems. Based on our previous study [26], combining SWIM with undersampling methods, such as TomekLinks and edited nearest neighbors (ENN), shows superiority in improving the classification performance on imbalanced cases with overlapped classes. However, an adequate study addressing the issue arising from noisy data is still needed.

The presence of noise is another significant issue that typically accompanies imbalanced data classification, leading to a substantial decrease in the classifier performance [6]. The presence of noise poses a notable issue when the features of the majority data overlap with those of the minority data, causing disruptions in the formation of decision boundaries [27]. Detecting the possible noise data and then eliminating those data from the majority class is a crucial step. A study by [28] demonstrated an improvement of SMOTE by addressing the noisy and borderline examples problem in imbalance classification. Therefore, these results have inspired us to modify the SWIM framework by integrating a procedure to detect and remove possibly noisy data from the majority class. The noise reduction step will lead to more precise majority class features for the SWIM framework to generate a precise representation of the synthetic minority data.

Clustering algorithms can be applied for the detection of noise in data due to their likelihood of producing "byproducts" that are commonly known as outliers. Moreover, utilizing clustering for noise data detection is more straightforward and intuitive [29]. Therefore, we utilized a density-based method, that is, the density-based spatial clustering of applications with noise (DBSCAN) algorithm, for identifying possibly noisy data. This algorithm identifies points in regions with low density as noise data [30] and is widely adopted in various applications because it does not require a predefined number of clusters, is insensitive to noise, and can handle high-dimensional data while disregarding the shape and size of the data [31–34].

The significant contributions of this study are outlined as follows:

1. We propose a hybrid sampling framework named noise-free sampling with majority that addresses the challenges arising from imbalanced class and noise data simultaneously. By utilizing DBSCAN, our proposed framework is able to identify potentially noisy data and eliminate them from the majority class, enhancing the accuracy of the features used to generate synthetic data for the minority class.
2. We conduct an extensive analysis of the proposed framework. For this evaluation, we employ eleven datasets with diverse imbalance ratios to ensure unbiased evaluation results. Moreover, to demonstrate the effectiveness of our proposed framework across various classifiers, we utilized eight generally recognized classifiers in the evaluation process.

The remainder of this paper is organized as follows. Section 2 describes related work. Section 3 defines the proposed methodology. Section 4 explains the experimental datasets and experimental setup. The results and discussion of applying the proposed methodology are presented in Sect. 5. Finally, Sect. 6 concludes the overall results.

2 Related work

The majority of machine learning algorithms perform optimally when applied to datasets with balanced class distributions. Nonetheless, real-world applications often involve data that exhibit unequal class distributions, also known as imbalanced data. This imbalance raises challenges in machine learning, particularly when dealing with minority classes and it

can lead to bias during training and reduced accuracy during testing. Consequently, numerous researchers are dedicated to finding solutions to address this issue. Sampling-based approaches represent a methodology for addressing imbalanced data distributions by adjusting the original class frequencies through data addition or removal. To address imbalanced data problems, three main categories are commonly employed: oversampling, undersampling, and hybrid methods combining both oversampling and undersampling.

A popular technique within oversampling methods is the synthetic minority oversampling technique (SMOTE) [35], which generates new data by randomly interpolating existing pairs of nearest neighbors. However, SMOTE solely prioritizes the minority class and neglects the majority class. Consequently, this could compromise the performance, as the generated data may deviate significantly from that of the minority class [25]. Several variations of SMOTE have been proposed to improve the algorithm's performance, including Borderline-SMOTE [36]. However, the issue of SMOTE remains to be solved.

TomekLinks, a widely used undersampling method, identifies all nearest neighbors between overlapping classes and removes data from the majority class that are close to those in the minority class [37]. Another commonly applied undersampling method is the edited nearest neighbor (ENN) method [38]. The ENN method eliminates data from the majority class that exhibit different predictions from those of the majority class when using the *k*-nearest neighbors (kNN) method.

Achieving a more balanced distribution can be accomplished through either oversampling or undersampling independently. However, each method has limitations when used separately. In oversampling methods, data from the minority class is often duplicated, leading to the introduction of redundant information and resulting in overfitting. On the other hand, undersampling methods require removing data from the majority class, leading to the loss of valuable information from the original data. To address these issues effectively, a combination of both strategies has been found to be more successful. A study by [39] demonstrates that the combination of SMOTE with TomekLinks and SMOTE with ENN provides excellent results, particularly for highly imbalanced datasets with few minority examples. Moreover, Sasada et al. [40] integrated two resampling methods, known as SMOTEENN and SMOTETomek, to solve imbalanced datasets by considering noise and overlap. The results demonstrate the effectiveness of the approach in comparison with that of traditional methods.

Another significant challenge commonly associated with imbalanced data classification is the existence of noise, which has the potential to significantly degrade the performance of a classifier. Noisy data in a dataset is identified by their differentiation from the rest of the data, as they do not align with the general patterns found within the dataset [41]. As a result, accounting for noise data could lead to incorrect conclusions. In datasets with imbalanced classes, noisy data contributes to overlap scenarios and potentially increase the class imbalance ratio, while discarding them could result in a significant loss of information [42].

Various studies have been proposed to solve the noise problem in a given dataset. Fang et al. [43] introduced a clustering algorithm-based approach to detect and eliminate noise from air pollutant data gathered through mobile portable sensors. They tested and compared six clustering algorithms to determine which was the most suitable. These algorithms include the simple K-means, hierarchical clustering, cascading K-means, X-means, expectation maximization, and self-organizing map. The results of the experiment showed that the expectation maximization and cascading and simple K-means algorithms yielded the most effective results. Hao et al. [29] used classical DBSCAN to cluster the data and remove the noise. The results indicated that removing the noise using DBSCAN is significantly efficient for improving the classification performance.

As mentioned in Sect. 1, the state-of-the-art oversampling framework, namely, SWIM, generates synthetic samples regarding the information of the majority class and has demonstrated its effectiveness in managing highly imbalanced problems. Our previous study demonstrated that enhancing the representation of the majority class by utilizing under-sampling techniques to remove overlapping data results in a better generation of synthetic data in the SWIM framework and leads to subsequent improvement in the classification performance [26]. However, the problem arising from noisy data remains to be solved and needs to be addressed. Motivated by this problem, we proposed an integrated framework that improves the synthetic generation process of SWIM by incorporating a mechanism to identify and eliminate potential noise within the majority class. Detecting and eliminating noise in the majority class leads to more precise features for the SWIM framework to generate a better representation of the synthetic data. As a result, our framework is particularly efficient for classification tasks where there is limited data in the minority class and there is possibly noisy data in the majority class.

3 Noise-free sampling with majority (NF-SWIM) framework

In this section, we introduce a hybrid framework to improve the synthetic data generation process of SWIM by adding a mechanism to eliminate the noise data in the majority class. While our previous study focused on removing overlapping data between the classes using undersampling methods [26], this study highlights noise detection methods and eliminates noisy data to provide a noise-free majority class and improve the quality of the synthetic data generation process. Identifying and removing noise data from the majority class enhances the accuracy of the features used by the SWIM framework, leading to an improved representation of the synthetic data. We refer to this hybrid framework as noise-free sampling with majority (NF-SWIM).

Figure 1 illustrates the workflow of the proposed framework. There are two main components in the NF-SWIM framework: noise reduction for the majority class and synthetic data generation for the minority class to obtain a balanced dataset. We modified and improved the SWIM framework by integrating a process to reduce the noisy data in the majority class. The SWIM framework uses the distributional information of the majority class to generate synthetic data for the minority class. Therefore, the quality of the majority class plays an

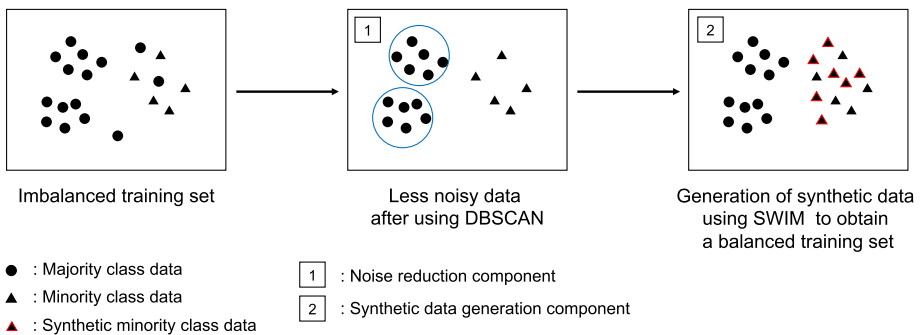


Fig. 1 Workflow of the NF-SWIM framework. There are two main components of this framework: (i) removing the noisy data in the majority class and (ii) generating synthetic data for the minority class to obtain a balanced training set

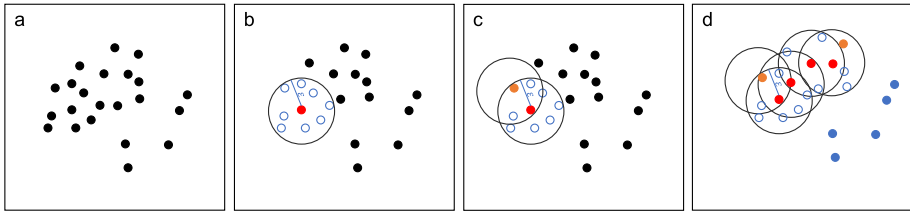


Fig. 2 Illustration of DBSCAN. **a** A cluster of a dataset. **b** A core object. **c** A border point. **d** Cluster-density reachable objects and noise points. Points with four-pointed red dots represent core points. Points with blue outlined circles represent ε -neighborhoods of corresponding core points. Points with yellow dots represent border points. Points with blue dots represent noise points

important role in generating synthetic data. By integrating this noise reduction step, we can provide the SWIM framework with more accurate features of the majority class, allowing for a more precise generation of synthetic minority data. The noise reduction component used in our study is introduced in Sect. 3.1. Then, the concept of the synthetic data generation component is described in Sect. 3.2. Finally, an explanation of how NF-SWIM works is presented in Sect. 3.3.

3.1 Noise reduction component

In this section, we explain the noise reduction component of NF-SWIM. This component removes the noisy data from the majority class and improves the SWIM framework in terms of generating more accurate synthetic data. In this study, we applied the DBSCAN clustering algorithm to detect the noisy data in the majority class. DBSCAN clusters the data from the majority class and obtains the cluster label. Data with low cluster density will be labeled as noise and eliminated from the majority class.

DBSCAN, a widely recognized clustering algorithm, is capable of identifying clusters with arbitrary shapes and does not rely on a predefined number of clusters [44]. This algorithm can identify noise as points with low cluster-density areas [30] and works based on two input parameters, epsilon (ε) and the minimum sample points (*minPts*) [32–34]. ε determines the radius of the neighborhood surrounding each data point, and *minPts* represents the minimum number of points within ε .

Given a dataset of n points $D = \{x_1, x_2, \dots, x_n\}$, where $x_i = \{x_{i1}, x_{i2}, \dots, x_{id}\}$ for $1 \leq i \leq n$ is a vector with d dimensions, the DBSCAN algorithm defines distinct relationships between any two different points as follows.

1. Directly cluster-density reachable. A point p is directly cluster-density reachable from a point q if it lies within an ε distance from the core point p . The values of the distance function vary when using various distance metrics, such as the Manhattan distance or Euclidean distance.
2. Cluster-density reachable. A point p is cluster-density reachable to a point q with respect to ε and *minPts*, if there is a sequence of points p_1, \dots, p_n , and $p_1 = q$, $p_n = p$, such that p_{i+1} is directly cluster-density reachable from p_i with respect to ε and *minPts*, for $1 \leq i \leq n - 1$, $p_i \in D$ (see Fig. 2 points with red four-pointed dots).

Based on the relationships mentioned above, DBSCAN classifies all points into three categories, i.e., core points, border points, and noise points. These point categories are illustrated in Fig. 2.

1. Core point. If the number of points directly cluster density reachable from point p is greater than $minPts$ within the ε -neighborhood, point p is classified as a core point (refer to Fig. 2(b), point with red four-pointed dots).
2. Border point. If the number of points within the ε -neighborhood of point p is less than or equal to $minPts$ and p is directly cluster-density reachable from a core point, then p is categorized as a border point (refer to Fig. 2c, point with yellow dots).
3. Noise point. If a point p is neither a core point nor a border point, then p is a noise point (refer to Fig. 2d, point with blue dots).

The DBSCAN process begins by selecting an unvisited starting point and calculating its cluster density, representing the number of points in its ε -neighborhood [45]. If the point qualifies as a core point, DBSCAN labels it as a new cluster and includes all points that are cluster-density reachable to it, labeling them with the same cluster label. Otherwise, the selected point is labeled as a noise point. For a point that has not been labeled, it undergoes a neighborhood function to receive a label with the following properties: (i) Neighborhood functions are executed solely for points labeled as *undefined*. (ii) When a neighborhood function is conducted on a point, it is subsequently labeled either as a cluster label or *noise*. (iii) The point is relabeled only when its label changes from *noise* to a cluster label. Consequently, DBSCAN iteratively gathers point cluster density reachable from core points, continuing until no new point can be added to any cluster. DBSCAN examines the dataset only once and computes the distance between any pair of objects in the dataset.

3.2 Sampling with the majority class (SWIM)

In this section, we explain how we integrated SWIM into our framework. The SWIM framework is known for utilizing the density of each minority class data in relation to the majority class distribution to determine the location for generating synthetic data [25]. However, in scenarios where there is considerable noisy data in the majority class and overlap between classes, the SWIM framework will introduce misleading information about the majority class distribution and generate a poor representation of synthetic data for the minority class. As a result, classifiers built with SWIM data may underperform in domains where the noisy data and overlap problems are prevalent. However, we expect that our noise reduction component can overcome these problems.

The principal components of SWIM are density estimation and shift procedures. Any appropriate technique for estimating density can be used. The SWIM framework can be implemented using the Mahalanobis distance (SWIM-MD) and radial basis function (SWIM-RBF) [25]. In this study, we focused on improving SWIM-MD which has less time complexity than SWIM-RBF when it is applied to a dataset. The SWIM framework effectively employs the Mahalanobis distance (MD) to calculate the distance between the mean of the majority class distribution and a minority class seed point while considering the density along the path connecting them. Accordingly, points lying on the same hyperelliptical density contour will have an equal MD from the mean. This differs from the concept of the Euclidean distance, where the distance is mainly the straight-line distance between two points in Euclidean space. The MD accounts for the correlations within the dataset, which is crucial when the variables are correlated. In scenarios where the variables are correlated, the Euclidean distance may not provide an accurate representation of the actual distance between observations. Figure 3 shows an illustration of the Mahalanobis distance between two points p and q from the mean. Within the SWIM framework, any point in the nearby regions of the data space with the same MD as a minority seed can be sampled as synthetic minority training data. The

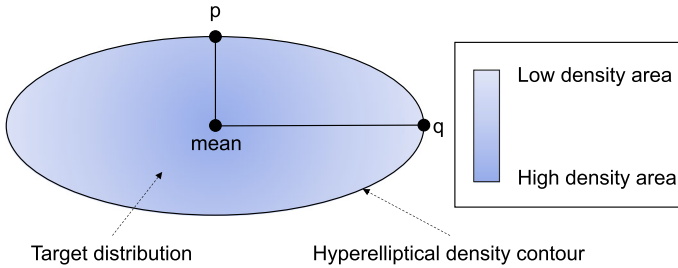


Fig. 3 Illustration of the Mahalanobis distance between two points p and q from the mean. Both points have the same Mahalanobis distance

MD calculation requires the known mean μ and covariance matrix Σ of the distribution. In practice, these parameters are estimated using $\bar{\mu}$ and $\bar{\Sigma}$ of a sample population.

The process of the SWIM framework using MD is described in lines 5–14 of Algorithm 1. Line 5 includes the application of preprocessing measures to streamline both data generation and $\bar{\mu}$ and $\bar{\Sigma}$ estimation from the majority class examples. The steps of this process are as follows.

1. Center the majority and minority classes: Let μ^- be the feature mean vector of the majority class. Initially, the majority class is adjusted to have a $\mathbf{0}$ mean and subsequently aligns the minority class with the mean vector of the majority class.

$$\begin{aligned} D_c^- &= D^- - \mathbf{0} \\ D_c^+ &= D^+ - \mu^- \end{aligned} \tag{1}$$

2. Whiten the minority class: Let Σ denote the estimated covariance matrix of D_c^- , and Σ^{-1} denote its inverse. $\Sigma^{-\frac{1}{2}}$ is the square root of Σ^{-1} . The center minority class is whitened as follows:

$$D_w^+ = D_c^+ \Sigma^{-\frac{1}{2}}. \tag{2}$$

In the whitened space of a distribution, the MD is analogous to the Euclidian distance.

3. Find the feature limits: Feature limits are employed to limit the dispersion of the synthetic samples. Find the mean μ_f and standard deviation σ_f for each feature f in D_w^+ . Then, calculate its upper and lower limits.

$$\begin{aligned} u_f &= \mu_f + \alpha \sigma_f \\ l_f &= \mu_f - \alpha \sigma_f \end{aligned} \tag{3}$$

$\alpha \in \mathbb{R}$ regulates the number of standard deviations for the limits.

Then, the process described in lines 8-13 of Algorithm 1 is iterated until the desired number of synthetic samples is generated, and in practice, these lines are repeated until a balanced class distribution is achieved. During each iteration, a minority staple d_i is randomly chosen and the shift procedure is employed to generate synthetic data. The shifting procedure for SWIM is described as follows.

1. Generate a new sample. A random number is generated for each feature f between u_f and l_f . Accordingly, a sample point d'_i is obtained in the whitened space, where each feature $d'_{i,f}$ is $l_f \leq d'_{i,f} \leq u_f$. A sample is generated at the same Euclidean distance

from the mean of the majority class. As a result of centering the data, the new sample will have an identical Euclidian norm to that of the minority seed d_i .

$$d_i^{\text{norm}} = d_i' \frac{\|d_i\|_2}{\|d_i'\|_2} \tag{4}$$

2. Return the scaled sample to its original space.

$$d_i'' = (\Sigma^{-\frac{1}{2}})^{-1} d_i^{\text{norm}} \tag{5}$$

The synthetic sample d_i'' will be located in the same density contour as its corresponding minority seed data d_i .

Algorithm 1 NF-SWIM

Input:

- D^+ : original minority class training set
- D^- : original majority class training set
- n^+ : the number of minority class data
- ε : radius of cluster
- $minPts$: density threshold, the minimum number of samples for a cluster
- $dist$: distance function

Output:

- D' : balanced training set

/ NF-SWIM sampling method */*

```

1: label ← GetClusterLabel( $D^-$ ,  $\varepsilon$ , minPts, dist) //DBSCAN Algorithm
2:  $D_0^- \leftarrow D^-$  with label = Noise
3:  $D_1^- \leftarrow D^- \setminus D_0^-$ 
4:  $n_1^- \leftarrow$  the number of data in  $D_1^-$ 
5: Estimate a density function  $\hat{P}_{D_1^-}$  from  $D_1^-$ 
6:  $n \leftarrow n_1^- - n^+$  // Number of samples to generate
7:  $k = 1$ 
8: while  $k \leq n$  do
9:    $d \leftarrow$  select a random data from  $D^+$  set
10:   $p \leftarrow \hat{P}_{D_1^-}(d)$  // Get the density of  $d$  with the center density function of  $D_1^-$ 
11:   $d' \leftarrow shift(d, p)$  // Shift  $d$  to neighboring region with density  $p$ 
12:   $D_{new}^+ \leftarrow d' \cup D^+$ 
13:   $k = k + 1$ 
14: end while
15: return  $D' \leftarrow D_{new}^+ \cup D_1^-$ 

```

3.3 Implementation of NF-SWIM

In this section, we explain how we used our noise reduction technique to improve the quality of synthetic data generation from SWIM. As previously mentioned, the performance of SWIM is degraded when there are noisy data and overlapping classes. As mentioned in Sect. 3, the framework has two main components: the first component is responsible for removing noise data, and the second component is responsible for the synthetic data generation process. The process of the framework is described in Algorithm 1.

The first part of the framework (lines 1–6) is responsible for removing noise. First, DBSCAN is used to detect noisy data in the majority class D^- (line 1). The function

GetClusterLabel is called using the DBSCAN method to discover noisy data in the majority class regions, characterized by low cluster-density data. The data points that are believed to contain noise are labeled and added to a group named D_0^- (line 2). We then subtract D_0^- from D^- , resulting in a noise-free group named D_1^- (line 3). The variable that holds the length of the majority class is updated (line 4). We use SWIM to estimate the probability density function (PDF) $\hat{P}_{D_1^-}$ from D_1^- (line 5) to capture the distributional characteristics of the noise-free majority class. The process of estimating the PDF $\hat{P}_{D_1^-}$ is inspired by using the MD of the SWIM framework and described in detail in Sect. 3.1. The process starts by estimating $\bar{\mu}$ and $\bar{\Sigma}$ from D_1^- . $\bar{\mu}$ is estimated by calculating the feature mean vector of D_1^- . Then, D_1^- is centered to have a $\mathbf{0}$ mean, as shown in equation 1, resulting in D_c^- . The $\bar{\Sigma}$ is then estimated by calculating the covariance matrix of D_c^- . After estimating the parameters of $\hat{P}_{D_1^-}$, we calculate the number of samples to be generated n (line 6).

After the majority class is noise-free, our framework proceeds to the next step, which is to iteratively generate data for the minority class (lines 8–14). First, we select a random data point from the minority class d . Then, the algorithm estimates the PDF of the selected data point and stores it in p with the center of the density vector of D_1^- . The process of estimating the PDF of d begins by aligning with the mean vector of D_1^- using the Eq. 1. Then, the center of d is whitened, as described in Eq. 2, by using the estimated covariance matrix of D_c^- . After that, the algorithm shifts the selected point d to the neighboring region with density p , resulting in a shifted point d' . The inspiration for the shifting procedure is comprehensively explained in Sect. 3.1. The generated point d' is then inserted into the minority class. This process is repeated until both classes are balanced.

4 Experimental setup

An extensive study was carried out to evaluate the effectiveness of the proposed NF-SWIM framework. First, we introduce the dataset used in the experiment in Sect. 4.1. Then, we describe the data preprocessing step before employing the proposed method in Sect. 4.2. The evaluation metrics for this experiment are defined in Sect. 4.3. The parameter settings used in this experiment are described in Sect. 4.4. Finally, the technical specifications used in this experiment are presented in Sect. 4.5.

4.1 Dataset

In this experiment, we test the NF-SWIM framework on a real-case imbalanced dataset, which is available online.¹ The real-case dataset is obtained from the Landing Club company and primarily focuses on the variable determining loan defaults. This imbalanced dataset contains recorded transaction data from 2007 to 2015. We also used ten benchmark datasets from the KEEL repository [46] with highly imbalanced cases that are commonly used in other studies [25, 26]. These datasets are selected to test the generalizability of the proposed framework and compare with other sampling-based methods.

Table 1 describes the summary of the datasets, including N_{ins} denoting the total number of data, N_{var} representing the number of variables, and IR , denoting the imbalance ratio, which is the ratio of minority data to the total of data. A low IR means that a particular class is represented much less than others in the dataset, leading to an imbalanced dataset. An IR

¹ <https://github.com/sarathi-tech/lending-club>

Table 1 Summary of the datasets

| Dataset | N_{ins} | N_{var} | IR | degOver |
|----------------------------|------------------|------------------|------|------------------|
| Loan data | 9578 | 14 | 0.16 | 0.61 |
| Abalone19_vs_10-11-12-13 | 1622 | 8 | 0.02 | 0.41 |
| Abalone20_vs_8-9-10 | 1916 | 8 | 0.01 | 0.04 |
| Abalone19 | 4174 | 8 | 0.01 | 0.07 |
| Abalone9-18 | 731 | 8 | 0.06 | 0.41 |
| Winequality-red-3_vs_5 | 691 | 11 | 0.01 | 0.04 |
| Winequality-white-3-9_vs_5 | 1482 | 11 | 0.02 | 0.06 |
| Yeast-0-5-6-7-9_vs_4 | 528 | 8 | 0.10 | 0.19 |
| Yeast-1_vs_7 | 459 | 7 | 0.07 | 0.15 |
| Yeast-1-2-8-9_vs_7 | 947 | 8 | 0.03 | 0.07 |
| Yeast-1-4-5-8_vs_7 | 693 | 8 | 0.04 | 0.10 |

close to zero indicates a highly imbalanced dataset. The overlap ratio, calculated using the *degOver* formulation, measures the overlap of data of both classes relative to the total data in the data space [47]. A larger *degOver* value signifies a greater overlapping area. These datasets exhibit varying imbalance ratios, ranging from 0.01 to 0.16, and overlap ratios that span from 0.04 to 0.61.

4.2 Data preprocessing

Prior to using the NF-SWIM framework, a preprocessing step is employed. The sampling-based method and several classification methods, such as kNN and QDA, rely on distance metrics. Consequently, these methods cannot process string-based categorical variables, and these variables must be converted into numerical values. Therefore, we transform all the categorical variables into binary variables using one-hot encoding. Additionally, since the numeric variables have a distinct measurement scale, it is necessary to normalize the variables to ensure a uniform measurement scale, which is achieved by individually scaling and transforming each variable to fall within the range[0,1] using min-max scaling [48].

4.3 Evaluation metric

The geometric mean (*G-mean*) provides a unified value that effectively evaluates both the majority and minority classes together and it surpasses traditional metrics in performance [49]. The *G-mean* accurately captures the level of inductive bias by precisely assessing the qualities of the majority and minority classes. To compute the *G-mean* for a classification model, one must consider the accuracy on the target positive class and the accuracy on the negative class. The formula for calculating the *G-mean* is as follows:

$$G\text{-mean} = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}}. \quad (6)$$

TP (true positive) refers to a scenario where the model makes a correct prediction of the positive class. Correspondingly, *TN* (true negative) is an outcome in which the model accurately predicts the negative class. On the other hand, *FP* (false-positive) occurs when the

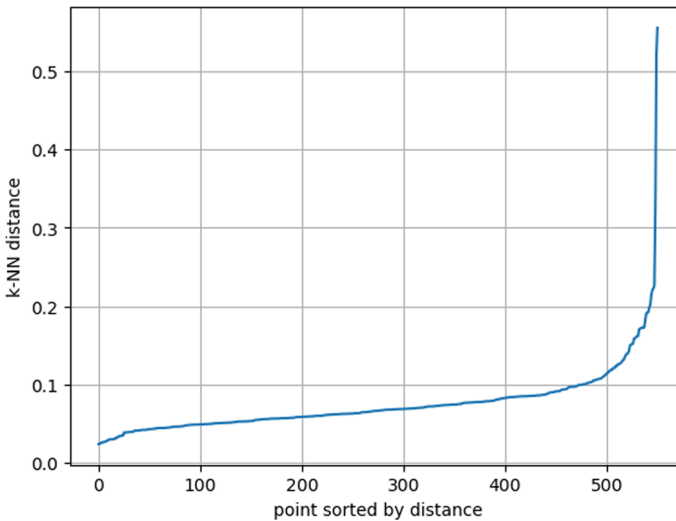


Fig. 4 Example of a k -distance graph for the Abalone9-18 dataset when $k = 5$. The optimum ε is approximately 0.1 with $minPts = 5$

model incorrectly predicts the positive class, and FN (false-negative) arises when the model makes an incorrect prediction of the negative class.

4.4 Determining the parameters

The DBSCAN performance relies on two parameter values that must be carefully chosen to conform to the input dataset, the minimum sample points ($minPts$) and epsilon (ε). The number, shape, and size of the identified clusters are significantly influenced by the two parameter values. It is essential to note that $minPts$ depends on ε , as it represents the minimum number of observations expected within a given distance around each point. In this study, we apply a heuristic method to determine the appropriate input parameters for DBSCAN. However, determining the two parameters is a challenging task. Higher $minPts$ values resulted in increased fragmentation since small features with a number of points below $minPts$ are considered noise. On the other hand, lower $minPts$ values lead to excessive connectivity between clusters and a high number of detected small features [50]. It is common to select the $minPts$ parameter empirically based on the specific dataset under investigation [51].

In general, selecting smaller values for the ε parameter is often preferred. One heuristic method to determine the appropriate ε value is based on the distances within the dataset. To achieve this, the k -distance graph is utilized [52]. Specifically, we calculate the mean of the distances between each point and its k nearest neighbors, where k corresponds to $minPts$. By plotting these k -distances in ascending order, we observe a distinct point known as a “knee” or “valley.” The value of this knee point represents the optimal ε parameter. In this study, we set the $minPts$ value to 5. Then, we use the $minPts$ value as k for the k -distance graph to obtain the optimum ε in each dataset. Figure 4 shows an example of the k -distance graph on the Abalone9-18 dataset when $k = 5$. In the graph, we can approximate that the

Table 2 Summary of optimum ε

| Dataset | Optimum ε |
|----------------------------|-----------------------|
| Loan Data | 0.35 |
| Abalone19_vs_10-11-12-13 | 0.125 |
| Abalone20_vs_8-9-10 | 0.1 |
| Abalone19 | 0.175 |
| Abalone9-18 | 0.1 |
| Winequality-red-3_vs_5 | 0.3 |
| Winequality-white-3-9_vs_5 | 0.25 |
| Yeast-0-5-6-7-9_vs_4 | 0.2 |
| Yeast-1_vs_7 | 0.2 |
| Yeast-1-2-8-9_vs_7 | 0.9 |
| Yeast-1-4-5-8_vs_7 | 0.25 |

Table 3 Hyperparameters of the classifiers

| Classifiers | Hyperparameter |
|------------------------|---|
| k-Nearest neighbors | n_neighbors = 3; weight = uniform; leaf_size = 30; p = 2; metric = minkowski |
| SVC with linear kernel | C = 0.025 |
| SVC with RBF kernel | C = 2; gamma = 1/(n=feature * X.var()) |
| Random forest | n_estimators = 4; max_depth = 5; min_sample_split = 2; min_sample_leaf = 1; min_weight_fraction_leaf = 0; max_features = 5 |
| MLP | hidden_layer_sizes = 100; alpha = 0.1; batch_size = min(200, n_samples); learning_rate_init = 0.001; power_t = 0.5; max_iter = 200; tolerance = 1e-4; momentum = 0.9; validation_fraction = 0.1; beta_1 = 0.9; beta_2 = 0.999; epsilon = 1e-8; n_iter_no_change = 10; max_fun = 15000 |
| AdaBoost | n_estimators = 50; learning_rate = 1.0 |
| Naive Bayes | var_smoothing = 1e-9 |
| QDA | - |

optimum ε is 0.1 when *minPts* equal to 5. Table 2 shows the summary of the optimum ε of the eleven datasets after examining the knee point of each k -distance graph when $k = 5$.

In this experiment, we assess the classification performance on a sampling-based method training set using eight different classifiers. These classifiers are the k -nearest neighbor (kNN), support vector classification (SVC) with both linear and radial basis function (RBF) kernels, random forest, multilayer perceptron (MLP), AdaBoost, naive Bayes classifier, and quadratic discriminant analysis (QDA). The hyperparameters of the classifiers used in this study are initially tuned on the Abalone19 dataset. Subsequently, the same hyperparameters are applied across all datasets, as described in Table 3. It is important to note that we do

not anticipate these hyperparameters to significantly influence our results, as our primary focus is on the preprocessing step. This involves developing a method for balancing the class distribution and testing the balanced data with the original model, rather than optimizing the classifier performance through hyperparameter tuning.

4.5 Technical specifications

The experiment is performed on an Intel Core i9-990K CPU, equipped with 64 GB of RAM. The operating system used is Linux Mint 21.2 “Victoria,” chosen for its wide community support and stability. We utilize Python 3.10 as the programming language. Additionally, the libraries used for this experiment are Pandas 2.1.0 [53], Numpy 1.25 [54], Scipy 1.11.2 [55], Matplotlib 3.6.0 [56], scikit-learn 1.3.0 [57], and imbalanced-learn (imblearn) 0.11.0 library from scikit-learn [58].

5 Results and discussion

In this section, we conduct a comprehensive evaluation of the proposed framework to analyze the efficiency and effectiveness of solving the imbalanced class problem in the presence of noisy data in the majority class. We compare the performance of NF-SWIM with that of other sampling-based methods. Table 4 shows a summary of other sampling-based methods and their categories.

To validate the improvement in classifier performance for imbalanced datasets, achieved through sampling-based approaches, we compare the classification results derived from original imbalanced datasets against those from balanced datasets generated by sampling-based methods. To enhance readability and facilitate understanding, in this study, we use the term “baseline” to refer to the classification results obtained from the original imbalanced data. Additionally, to ensure a robust evaluation, the datasets are divided into training and testing

Table 4 Sampling-based methods for the comparison analysis

| Method | Category | | |
|---------------------------------------|--------------|---------------|--------|
| | Oversampling | Undersampling | Hybrid |
| SMOTE [35] | ✓ | | |
| ADASYN [59] | ✓ | | |
| BL-SMOTE [60] | ✓ | | |
| SWIM-MD [25] | ✓ | | |
| SWIM-RBF [25] | ✓ | | |
| NearMiss [61] | | ✓ | |
| ENN [38] | | ✓ | |
| TomekLinks [37] | | ✓ | |
| Neighborhood Cleaning Rule (NCR) [62] | | ✓ | |
| SMOTEENN [39] | | | ✓ |
| SMOTETomek [39] | | | ✓ |
| Tomek-SWIM [26] | | | ✓ |
| ENN-SWIM [26] | | | ✓ |

data using fivefold cross-validation, which helps assess the model performance, reduce overfitting, and provide a more reliable estimate of how well a model is likely to generalize to new, unseen data. Subsequently, we computed the average of the evaluation metrics for each run.

The NF-SWIM performance evaluation on a loan dataset and ten benchmark datasets is first presented in Sect. 5.1. Then, a discussion on the capability of noise reduction components compared to that of undersampling methods within the majority class to improve the synthetic data generation process is described in Sect. 5.2. The evaluation of the effect of detected noise using NF-SWIM on the majority class is discussed in Sect. 5.3. The significance tests of the NF-SWIM performance against that of other sampling-based methods are presented in Sect. 5.4. The effect of the data distribution on the performance of NF-SWIM is evaluated in Sect. 5.5. Then, the effect of changing the hyperparameters of NF-SWIM on the variation in the classification performance is discussed in Sect. 5.6. Finally, the impact of utilizing the proposed NF-SWIM on the selective training partition is discussed in Sect. 5.7.

5.1 NF-SWIM performance evaluation

We test the proposed NF-SWIM on imbalanced datasets with various imbalanced ratios and overlapping issues. The experiment aims to verify the performance of the proposed NF-SWIM framework and validate the impact of removing noise from the majority class training set on the synthetic data generation process. For the analysis, the loan and KEEL datasets were evaluated independently. The loan data evaluation is described in Sect. 5.1.1. The performance of NF-SWIM on the KEEL benchmark datasets was evaluated and described in Sect. 5.1.2.

5.1.1 Performance evaluation with the loan dataset

In this section, we evaluate the performance of NF-SWIM against that of other sampling-based methods on a real imbalance problem. The loan dataset is evaluated to test the performance of NF-SWIM in terms of improving the classification of an actual imbalanced case, specifically in a case on the determination of loan default transactions. Table 5 shows the percentage average and the standard deviation G -mean of the results for all sampling-based methods across different classifiers. The highest percentage of the average G -mean for each classifier is shown in bold, while the second-best percentage of the average G -mean is underlined. Moreover, the highest percentage of the average G -mean across eight classifiers and all sampling-based methods is shown in bold with asterisk(*).

Compared to the performance of the baseline, the performance of all classifiers is significantly improved by implementing the sampling-based method to rebalance the class distribution, as shown by the increase in the percentage average of the G -mean after balancing the class distribution. Moreover, the standard deviation of the G -mean for each sampling-based method is relatively small, indicating low variability among the balanced training set across all five folds.

Furthermore, we evaluated the performance of each sampling-based method with respect to each classifier. According to the optimal G -mean pairing between each sampling-based technique and the classifier in Table 5, the proposed NF-SWIM outperforms existing sampling-based methods. NF-SWIM achieves the highest G -mean percentage compared to the other sampling-based techniques across all classifiers, shown in bold with asterisk numbers. This result shows that the proposed NF-SWIM is able to simultaneously remove noise

Table 5 Percentage average and standard deviation of *G*-mean on the loan dataset

| Method | Classifier | | | | | | | |
|------------|-------------------------|-------------------------|-------------------------|-------------------------|--------------------------|-------------------------|-------------------------|-------------------------|
| | KNN | SVC Linear | SVC RBF | Random forest | MLP | AdaBoost | Naive bayes | QDA |
| | Avg (Std.) | Avg (Std.) | Avg (Std.) | Avg (Std.) | Avg (Std.) | Avg (Std.) | Avg (Std.) | Avg (Std.) |
| Baseline | 31.14% (0.02) | 0.00% (0.) | 21.28% (0.02) | 8.46% (0.05) | 12.16% (0.03) | 18.24% (0.02) | 52.39% (0.02) | 47.85% (0.02) |
| SMOTE | 54.34% (0.02) | 57.11% (0.02) | 51.98% (0.02) | 58.34% (0.02) | 60.41% (0.01) | 50.34% (0.03) | 59.89% (0.02) | 58.54% (0.02) |
| ADASYN | 54.55% (0.02) | 57.16% (0.03) | 51.32% (0.03) | 57.68% (0.02) | 59.62% (0.02) | 47.26% (0.03) | 59.83% (0.02) | 59.28% (0.02) |
| BL-SMOTE | 53.77% (0.02) | 57.12% (0.01) | 50.08% (0.02) | 58.50% (0.02) | 60.03% (0.01) | 51.73% (0.01) | 60.57% (0.01) | 58.30% (0.02) |
| NearMiss | 54.59% (0.01) | 50.48% (0.01) | 52.09% (0.01) | 51.84% (0.03) | 52.83% (0.02) | 51.94% (0.01) | 54.61% (0.02) | 0.50% (0.01) |
| ENN | 50.30% (0.01) | 0.00% (0.) | 48.38% (0.03) | 38.85% (0.03) | 46.27% (0.02) | 46.46% (0.03) | 58.84% (0.02) | 56.75% (0.02) |
| Tomek | 36.34% (0.04) | 0.00% (0.) | 26.10% (0.01) | 7.58% (0.05) | 14.83% (0.01) | 23.13% (0.03) | 53.01% (0.03) | 49.39% (0.02) |
| NCR | 51.74% (0.01) | 0.00% (0.) | 46.83% (0.03) | 33.82% (0.05) | 44.75% (0.06) | 44.28% (0.02) | 57.25% (0.02) | 55.64% (0.02) |
| SMOTEENN | 56.54% (0.01) | 50.31% (0.02) | 55.99% (0.02) | 54.04% (0.03) | 56.43% (0.01) | 60.46% (0.01) | 60.82% (0.01) | 60.49% (0.01) |
| SMOTETomek | 54.52% (0.01) | 57.33% (0.02) | 52.48% (0.01) | 58.82% (0.02) | 60.66% (0.02) | 49.14% (0.04) | 60.45% (0.02) | 58.44% (0.02) |
| SWIM-MD | 58.87% (0.02) | 58.69% (0.01) | 58.53% (0.01) | 58.99% (0.02) | 58.81% (0.01) | 58.70% (0.01) | 58.68% (0.02) | 58.87% (0.02) |
| SWIM-RBF | 58.06% (0.02) | 58.21% (0.01) | 58.30% (0.02) | 57.91% (0.03) | 58.23% (0.02) | 58.47% (0.02) | 58.25% (0.02) | 57.48% (0.02) |
| Tomek-SWIM | 53.49% (0.02) | 57.73% (0.02) | 53.55% (0.02) | 59.48% (0.01) | 62.84% (0.02) | 61.70% (0.02) | 59.72% (0.03) | 58.45% (0.02) |
| ENN-SWIM | 56.78% (0.01) | 58.62% (0.01) | 58.58% (0.01) | 59.84% (0.02) | 62.56% (0.01) | 61.88% (0.02) | 60.41% (0.) | 59.52% (0.01) |
| NF-SWIM | 58.09% (0.02) | 58.72% (0.03) | 57.94% (0.02) | 60.38% (0.02) | 62.99%* (0.02) | 61.57% (0.01) | 60.17% (0.02) | 58.99% (0.01) |

The values presented in the table represent the percentage of the average *G*-mean along with its corresponding standard deviation (in the bracket) obtained from fivefold cross-validation. The bold numbers represent the highest percentage of the average *G*-mean for each classifier, while the italic numbers represent the second-best percentage of the average *G*-mean. The bold with * number represents the highest percentage of the average *G*-mean across eight classifiers and all sampling-based methods

from the majority class and generate a precise representation of synthetic data for the minority class for the real loan dataset, subsequently improving the classification performance, as demonstrated by the superior average *G*-mean percentage of 62.99%.

In a comprehensive manner, the SWIM-MD oversampling method achieved the highest percentage average G -mean with the kNN classifiers. The hybrid method, SMOTEENN, achieves the best percentage average G -mean with the naive Bayes and QDA classifiers. The ENN-SWIM method yields the highest percentage average G -mean with the SVC RBF kernel and AdaBoost classifiers. Furthermore, the proposed NF-SWIM outperforms the other sampling-based methods with three of eight classifiers, namely, SVC linear kernel, random forest, and MLP classifiers.

Upon analyzing to determine the most effective sampling-based method, as indicated by the highest average G -mean, further examination reveals the second-best outcomes noted by the underlined numbers in Table 5. Specifically, BL-SMOTE gave a good result performance with the naive Bayes classifier, ranking as the second-best method. In the context of SVC classifiers employing both Linear and RBF kernels, SWIM-MD demonstrated commendable efficacy. Additionally, Tomek-SWIM and ENN-SWIM emerged as the second-best methods for the MLP and AdaBoost classifiers, and the KNN, random forest, and QDA classifiers, respectively. Despite variability in results, the notable performance of BL-SMOTE, Tomek-SWIM, and KNN-SWIM underscores the enhancement in data quality, achieved by eliminating overlap and borderline data in the majority class, thereby fostering improved classifier performance.

From the comparisons of ENN-SWIM and NF-SWIM with SWIM-MD, we notice an improvement in the classification performance for most of the classifiers. Thus, removing noise and overlapping data in the majority class provides a more relevant majority class for SWIM-MD to generate a better representation of synthetic data for the minority class.

5.1.2 Performance evaluation with the KEEL datasets

In this section, we evaluate the performance of NF-SWIM against that of other sampling-based methods on the ten benchmark datasets from the KEEL repository. The KEEL benchmark datasets are evaluated to validate the generalizability across diverse conditions and a general comparison of the proposed framework with other sampling-based methods. To evaluate the performance of NF-SWIM on the KEEL benchmark datasets, we select the classifiers that pair best with each sampling-based method from eight tested classifiers (KNN, SVC linear kernel, SVC RBF kernel, random forest, MLP, AdaBosst, naive Bayes, and QDA).

The results are shown in Table 6, with the best percentage average G -mean values displayed in bold, while the second-best values are underlined. The results indicated that the proposed NF-SWIM framework outperforms the other sampling-based methods on nine of the ten datasets, while SWIM-RBF achieved superior performance on one dataset. Similar to the standard deviation on the loan dataset, the standard deviation of the G -mean for each sampling-based method across all KEEL datasets is relatively small. This result suggests minimal variability among the balanced training sets resulting from each sampling-based method for all five folds.

The results also show that NF-SWIM outperforms the existing hybrid methods, namely, SMOTETomek and SMOTEENN. These existing hybrid methods generate synthetic data using SMOTE and then remove noisy data. In contrast, our proposed framework eliminates noisy data from the original majority class set instead of removing noisy data from the balanced dataset. This result proves the effectiveness of removing noise from the majority class in improving the quality of the synthetic data generated using SWIM.

Following the identification of the optimal sampling-based method based on the highest average G -mean, we then analyze the second-best performing methods as outlined in Table 6. BL-SMOTE was the second-best on one dataset. Both SWIM-MD and SWIM-RBF gave

Table 6 Percentage average and standard deviation G -mean on the KEEL dataset

| Method | Dataset | | | | | | | | | | | |
|----------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------|------------|
| | ab1 | ab2 | ab3 | ab4 | wr | ww | y1 | y2 | y3 | y4 | Avg (Std.) | Avg (Std.) |
| Baseline | 7.54% (0.17) | 72.47% (0.11) | 38.95% (0.23) | 68.25% (0.14) | 42.22% (0.39) | 55.25% (0.1) | 65.49% (0.14) | 55.32% (0.17) | 41.84% (0.26) | 27.47% (0.04) | | |
| SMOTE | 72.23% (0.09) | 86.56% (0.15) | 69.21% (0.06) | 79.20% (0.14) | 72.53% (0.12) | 73.47% (0.12) | 79.96% (0.08) | 77.06% (0.07) | 72.07% (0.07) | 61.95% (0.08) | | |
| ADASYN | 70.25% (0.12) | 86.19% (0.15) | 72.48% (0.12) | 79.09% (0.14) | 71.98% (0.12) | 73.25% (0.12) | 79.47% (0.07) | 75.29% (0.09) | 71.47% (0.08) | 60.80% (0.07) | | |
| BL-SMOTE | 62.21% (0.19) | 85.28% (0.15) | 59.86% (0.12) | 77.61% (0.16) | 77.29% (0.14) | 61.03% (0.17) | 78.95% (0.04) | 77.43% (0.07) | 72.30% (0.08) | 66.10% (0.04) | | |
| NearMiss | 54.68% (0.07) | 62.63% (0.07) | 52.72% (0.16) | 62.79% (0.03) | 47.06% (0.3) | 50.92% (0.07) | 73.28% (0.13) | 67.18% (0.14) | 67.90% (0.05) | 58.10% (0.15) | | |
| ENN | 15.58% (0.21) | 77.01% (0.13) | 38.65% (0.23) | 69.55% (0.14) | 42.11% (0.38) | 58.83% (0.08) | 77.77% (0.1) | 64.74% (0.17) | 41.66% (0.26) | 27.47% (0.04) | | |
| Tomek | 31.36% (0.18) | 69.52% (0.11) | 49.42% (0.08) | 71.36% (0.09) | 42.22% (0.39) | 58.81% (0.08) | 68.32% (0.09) | 60.47% (0.19) | 31.06% (0.29) | 43.02% (0.25) | | |

Table 6 continued

| Method | Dataset | | | | | | | | | |
|------------|------------------------------------|------------------------------------|-----------------------------------|------------------------------------|------------------------------------|------------------------------------|------------------------------------|------------------------------------|------------------------------------|------------------------------------|
| | ab1 | ab2 | ab3 | ab4 | wr | ww | y1 | y2 | y3 | y4 |
| NCR | Avg (Std.) 15.60% (0.21) | Avg (Std.) 76.93% (0.13) | Avg (Std.) 38.94% (0.23) | Avg (Std.) 71.30% (0.14) | Avg (Std.) 42.16% (0.38) | Avg (Std.) 58.76% (0.08) | Avg (Std.) 77.12% (0.08) | Avg (Std.) 57.98% (0.12) | Avg (Std.) 45.07% (0.27) | Avg (Std.) 27.47% (0.04) |
| SMOTEENN | Avg (Std.) 69.95% (0.08) | Avg (Std.) 85.89% (0.14) | Avg (Std.) 71.58% (0.08) | Avg (Std.) 80.31% (0.08) | Avg (Std.) 69.61% (0.11) | Avg (Std.) 72.14% (0.12) | Avg (Std.) 77.91% (0.08) | Avg (Std.) 75.72% (0.03) | Avg (Std.) 69.93% (0.05) | Avg (Std.) 62.58% (0.09) |
| SMOTETomek | Avg (Std.) 72.56% (0.09) | Avg (Std.) 86.30% (0.15) | Avg (Std.) 69.14% (0.06) | Avg (Std.) 82.42% (0.09) | Avg (Std.) 72.79% (0.12) | Avg (Std.) 73.29% (0.12) | Avg (Std.) 79.37% (0.11) | Avg (Std.) 76.47% (0.05) | Avg (Std.) 71.91% (0.08) | Avg (Std.) 61.73% (0.11) |
| SWIM-MD | Avg (Std.) 70.57% (0.11) | Avg (Std.) 90.94% (0.05) | Avg (Std.) 75.10% (0.09) | Avg (Std.) 87.15% (0.07) | Avg (Std.) 74.16% (0.14) | Avg (Std.) 76.88% (0.06) | Avg (Std.) 82.40% (0.05) | Avg (Std.) 75.94% (0.13) | Avg (Std.) 73.62% (0.07) | Avg (Std.) 65.20% (0.11) |
| SWIM-RBF | Avg (Std.) 64.76% (0.14) | Avg (Std.) 88.15% (0.09) | Avg (Std.) 75.63% (0.05) | Avg (Std.) 85.78% (0.03) | Avg (Std.) 61.81% (0.37) | Avg (Std.) 73.38% (0.14) | Avg (Std.) 83.12% (0.09) | Avg (Std.) 78.28% (0.09) | Avg (Std.) 74.17% (0.11) | Avg (Std.) 68.52% (0.18) |
| Tomek-SWIM | Avg (Std.) 72.69% (0.08) | Avg (Std.) 88.02% (0.09) | Avg (Std.) 76.90% (0.05) | Avg (Std.) 82.99% (0.12) | Avg (Std.) 73.24% (0.13) | Avg (Std.) 75.83% (0.15) | Avg (Std.) 81.03% (0.09) | Avg (Std.) 76.36% (0.07) | Avg (Std.) 73.72% (0.07) | Avg (Std.) 65.40% (0.14) |
| ENN-SWIM | Avg (Std.) 71.64% (0.09) | Avg (Std.) 90.31% (0.1) | Avg (Std.) 74.39% (0.1) | Avg (Std.) 85.50% (0.11) | Avg (Std.) 66.18% (0.13) | Avg (Std.) 72.24% (0.17) | Avg (Std.) 83.07% (0.08) | Avg (Std.) 76.59% (0.12) | Avg (Std.) 73.87% (0.09) | Avg (Std.) 66.00% (0.06) |
| NF-SWIM | Avg (Std.) 74.99% (0.12) | Avg (Std.) 94.82% (0.04) | Avg (Std.) 77.54% (0.1) | Avg (Std.) 87.53% (0.06) | Avg (Std.) 82.64% (0.13) | Avg (Std.) 79.85% (0.08) | Avg (Std.) 83.15% (0.04) | Avg (Std.) 78.70% (0.12) | Avg (Std.) 75.51% (0.05) | Avg (Std.) 66.16% (0.09) |

The values in the table represent average and standard deviation (in the bracket) *G*-mean of fivefold cross-validation. The values of average *G*-mean are chosen from the classifiers that pair best with each sampling-based method from eight tested classifiers. The bold numbers represent the highest percentage of the average *G*-mean for each dataset, while the italic numbers represent the second-best percentage of the average *G*-mean
 ab1 = abalone19_vs_10-11-12-13; ab2 = abalone20_vs_8-9-10; ab3 = abalone19; ab4 = abalone9-18; wr = winequality-red-3_vs_5; ww = winequality-white-3-9_vs_5; y1 = yeast-0-5-6-7-9_vs_4; y2 = yeast-1_vs_7; y3 = yeast-1-2-8-9_vs_7; y4 = yeast-1-4-5-8_vs_7

the second-best results on three distinct datasets each. Tomek-SWIM attained this rank on two datasets, and NF-SWIM achieved similar results on one dataset. These findings indicate a diversity in the efficacy of alternative sampling-based methods that exhibit high performance, distinct from our proposed approaches. Notably, our proposed method consistently outperformed others in nine datasets.

5.2 Undersampling versus noise reduction

In this section, we discuss the capability of the noise reduction component compared to that of the undersampling method within the majority class to improve the synthetic data generation process using SWIM. The SWIM framework is recognized for utilizing the density of data from each minority class, relative to the distribution of the majority class, to determine where synthetic data should be generated [25]. However, when a substantial amount of noisy and overlapping data is presented in the majority class, SWIM tends to produce inaccurate information regarding the majority class distribution, leading to an inadequate generation of synthetic data for the minority class.

In our previous study, we attempted to solve the problem of overlapping data between classes by merging SWIM with undersampling methods, namely, Tomek-SWIM and ENN-SWIM [26]. This study demonstrates enhanced efficiency in improving the classification performance in an imbalanced problem where the classes overlap. Tomek-SWIM and ENN-SWIM use the TomekLinks and ENN methods, respectively, for detecting overlap between the majority and minority class data before generating synthetic data for the minority class. However, TomekLinks and ENN are not capable of detecting possibly noisy data in the majority class.

By removing noise from the majority class, we obtain more precise information about the distribution of the majority class set, leading to better-defined class groups. This noise-free majority class enhances the SWIM method's ability to generate more valuable synthetic minority data. The combination of the precise majority data and the valuable synthetic data in the balanced dataset contributes to an improved training process for the classification model.

5.3 Effect of noise on the majority class

In this study, we are interested in the noise detection performance of the sampling-based algorithm, i.e., the ability of the algorithm to detect possibly noisy data based on the density of the data and remove these data before generating the synthetic data to obtain a balanced class distribution. Therefore, in this section, we evaluate the effect of detected noisy data using NF-SWIM on the majority class. Through this evaluation, we aim to gain insight into how the removal of noisy data on the majority class influences the classification performance.

To answer this question, we test the classification performance on two different scenarios. The first scenario is training the classifier on new balanced dataset resulting from the proposed NF-SWIM method. The second scenario is retaining the detected noisy data and including them in the classification process. During the noise detection process, the detected noisy data are eliminated for the synthetic data generation process. However, these data are then restored in the classification process to preserve all the majority class information.

Figure 5 shows a comparison of the two scenarios, i.e., training on the new balanced dataset with or without the noisy data. The percentage average G -mean values when retaining the noisy data in the classification process are lower than those when the detected noisy data are removed. Therefore, removing the noisy data from the majority class increases

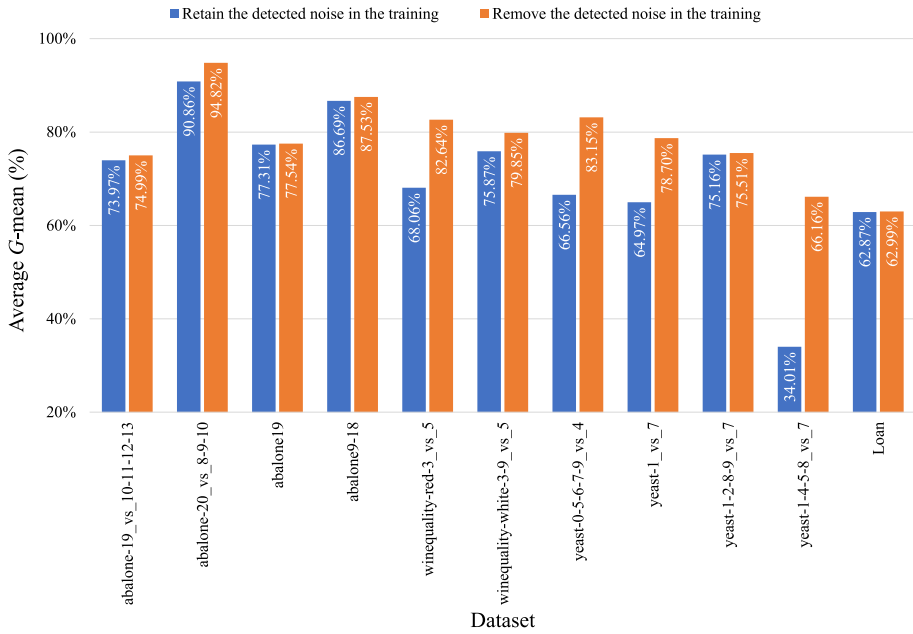


Fig. 5 Comparison of the percentage average *G*-mean between retaining and removing detected noisy data from the majority class. The graph shows that removing detected noisy data increases the classification performance

Table 7 The percentage of noise removed from the majority class

| Dataset | degOver | Noise removed (%) |
|----------------------------|---------|-------------------|
| Loan data | 0.61 | 3.94 |
| Abalone19_vs_10-11-12-13 | 0.41 | 1.05 |
| Abalone20_vs_8-9-10 | 0.04 | 0.10 |
| Abalone19 | 0.07 | 0.12 |
| Abalone9-18 | 0.41 | 1.78 |
| Winequality-red-3_vs_5 | 0.04 | 46.60 |
| Winequality-white-3-9_vs_5 | 0.06 | 33.60 |
| Yeast-0-5-6-7-9_vs_4 | 0.19 | 54.92 |
| Yeast-1_vs_7 | 0.15 | 70.59 |
| Yeast-1-2-8-9_vs_7 | 0.07 | 0.32 |
| Yeast-1-4-5-8_vs_7 | 0.10 | 68.98 |

the classification performance. This result provides evidence that the DBSCAN clustering algorithm can effectively detect the possibly noisy data in the majority class and provide more accurate information about the majority class data for the SWIM-MD method, enabling this method to generate a better representation of the synthetic data.

Moreover, Table 7 shows the percentage of noise removed from the majority class for each dataset. The results show that the loan dataset, all Abalone datasets (Abalone19_vs_10-11-12-13, Abalone20_vs_8-9-10, Abalone19, and Abalone9-18), and Yeast-1-2-8-9_vs_7

Table 8 Significance value of the Wilcoxon signed-rank test

| | ab1 | ab2 | ab3 | ab4 | wr | ww | y1 | y2 | y3 | y4 | Loan |
|------------|--------------|---------------|--------------|--------------|---------------|--------------|--------------|--------------|--------------|--------------|---------------|
| SMOTE | 0.64 | 0.38 | 0.95 | 0.02* | 0.04* | 0.01* | 0.55 | 0.74 | 0.01* | 0.95 | 0.01* |
| ADASYN | 0.38 | 0.55 | 0.74 | 0.01* | 0.02* | 0.01* | 0.64 | 0.84 | 0.01* | 1.00 | 0.02* |
| BL-SMOTE | 0.01* | 0.08** | 0.01* | 0.01* | 0.05** | 0.01* | 0.46 | 1.00 | 0.02* | 1.00 | 0.02* |
| NearMiss | 0.02* | 0.01* | 0.05* | 0.01* | 0.01* | 0.01* | 0.20 | 0.20 | 0.04* | 0.95 | 0.01* |
| ENN | 0.01* | 0.01* | 0.01* | 0.01* | 0.01* | 0.01* | 0.03* | 0.01* | 0.01* | 0.02* | 0.01* |
| Tomek | 0.01* | 0.01* | 0.01* | 0.01* | 0.01* | 0.01* | 0.02* | 0.02* | 0.01* | 0.03* | 0.01* |
| NCR | 0.01* | 0.01* | 0.01* | 0.01* | 0.01* | 0.01* | 0.02* | 0.01* | 0.01* | 0.02* | 0.01* |
| SMOTEENN | 0.84 | 0.74 | 0.84 | 0.02* | 0.04* | 0.01* | 0.55 | 0.84 | 0.02* | 0.74 | 0.05** |
| SMOTETomek | 0.25 | 0.31 | 0.74 | 0.01* | 0.04* | 0.01* | 1.00 | 0.84 | 0.01* | 1.00 | 0.02* |
| SWIM-MD | 0.01* | 0.01* | 0.01* | 0.04* | 0.04* | 0.02* | 0.95 | 0.84 | 0.01* | 0.95 | 0.15 |
| SWIM-RBF | 0.01* | 0.04* | 0.02* | 0.04* | 0.01* | 0.01* | 0.64 | 0.46 | 0.02* | 1.00 | 0.02* |
| Tomek-SWIM | 0.02* | 0.02* | 0.04* | 0.02* | 0.11 | 0.01* | 0.64 | 0.95 | 0.02* | 0.38 | 0.02* |
| ENN-SWIM | 0.02* | 0.02* | 0.04* | 0.02* | 0.11 | 0.01* | 0.64 | 0.95 | 0.02* | 0.38 | 0.95 |

Numbers in bold with * indicate that the NF-SWIM method is significantly better compared to the method in the same row at $\alpha = 0.05$, while numbers in bold with ** indicate significant improvement at $\alpha = 0.10$
 ab1 = abalone19_vs._10-11-12-13; ab2 = abalone20_vs._8-9-10; ab3 = abalone19; ab4 = abalone9-18; wr = winequality-red-3_vs._5; ww = winequality-white-3-9_vs._5; y1 = yeast-0-5-6-7-9_vs._4; y2 = yeast-1_vs._7; y3 = yeast-1-2-8-9_vs._7; y4 = yeast-1-4-5-8_vs._7

have a relatively small percentage of noise removed from the majority class, i.e., less than 4%. However, the rest of the datasets have a high percentage of noise removed from the majority class. Datasets that have a high *degOver* (Loan data, Abalone19_vs._10-11-12-13, and Abalone9-18) notably have a small percentage of noise removed from the majority class. Based on this evidence, it appears that the noise detected in the majority class does not inherently come from overlapping data between the two classes. NF-SWIM detects and removes noisy data from the majority class that are not aligned with the general pattern observed in the majority class. Eliminating noise from the majority class results in more accurate information about the distribution, consequently yielding more distinctly defined class groups.

5.4 Significance test of resampling performance

In this section, we test the significant improvement of NF-SWIM for balancing the class distribution compared to the results of the other sampling-based methods across different classifiers on all datasets. We use the nonparametric Wilcoxon signed-rank test to evaluate the significantly different performance of NF-SWIM compared to the performance of the other sampling-based methods.

The Wilcoxon signed-rank test is used to compare two related samples or to conduct a pair difference test on repeated measurements within a single sample. The aim is to determine if there are statistically significant differences in the population mean ranks of the compared elements [63, 64]. We compare the percentage average *G*-mean values of NF-SWIM with those of another sampling-based method across different classifiers for all datasets.

Table 8 shows the significance values of the Wilcoxon signed-rank test. We test the difference in average *G*-mean values under two levels of significance ($\alpha = 0.05$ and $\alpha = 0.10$).

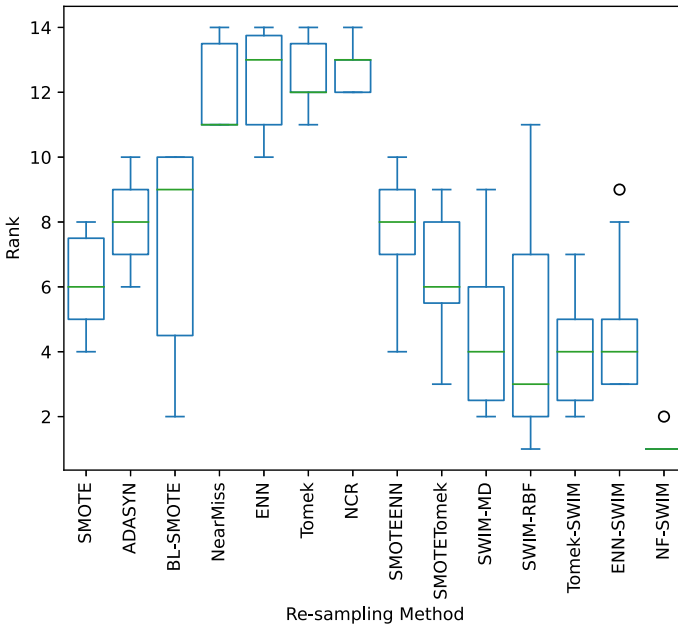


Fig. 6 Boxplots for rank distribution of the *G*-mean in each resampling method across all datasets. The green line in each boxplot depicts the median of the rank, and the size of the box shows the rank variability across all datasets

The bold numbers with * indicate that the proposed NF-SWIM is significantly better than the corresponding method of the row under $\alpha = 0.05$, and the bold numbers with ** indicate that the proposed NF-SWIM is significantly better than the corresponding method of the row method under $\alpha = 0.10$.

There are some pieces of evidence that indicate that the proposed NF-SWIM is significantly better than the other sampling-based methods, e.g., the significance values that are less than the significance level ($\alpha = 0.05$ or $\alpha = 0.10$) and shown in bold numbers. Some examples in Table 8 show that NF-SWIM is significantly better than other sampling-based methods for Abalone9-18, winequality-white-3-9_vs_5, and yeast-1-2-8-9_vs_7 datasets. These evidences are shown by the significance values that are less than the significance level of 0.05.

However, the Wilcoxon signed-rank test results show no statistically significant difference between NF-SWIM and some sampling-based methods for some datasets. One example from the data in Table 8, there is no significant difference between NF-SWIM and some sampling-based methods, namely, SMOTE, ADASYN, SMOTEENN, and SMOTE-Tomek, on the Abalone19_vs_10-11-12-13 dataset. Nevertheless, the proposed NF-SWIM framework yielded competitive results, with the best percentage average *G*-mean across all classifiers for all datasets. To support this statement, we evaluate the rank of the implemented methods for each dataset. Figure 6 shows the boxplot of the ranks for each sampling-based method on all datasets. Specifically, the boxplots demonstrate that NF-SWIM achieves the highest average ranks and exhibits lower variability in comparison with those of the other methods.

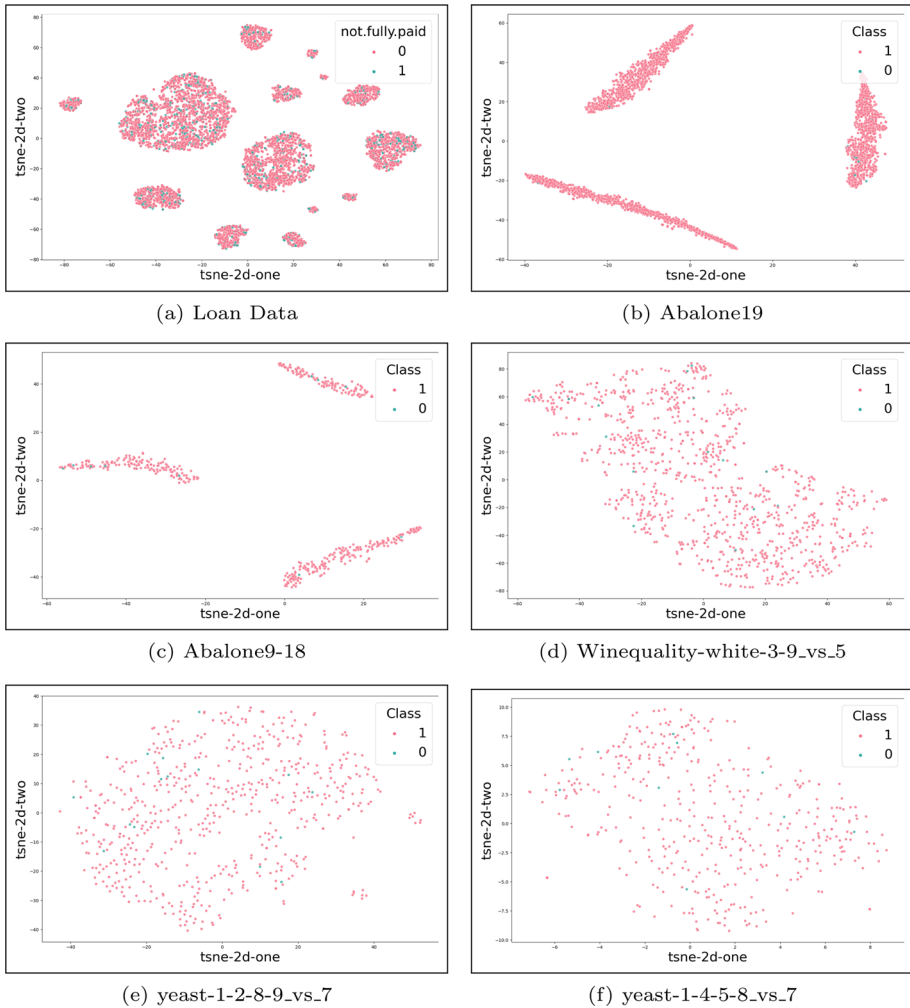


Fig. 7 t-SNE plot of six original imbalanced datasets. Various data distributions are represented

5.5 Effect of the data distribution on NF-SWIM

In this section, we evaluate the effect of the data distribution on the performance of NF-SWIM in balancing the data distribution. Our hypothesis is that the effectiveness of the proposed NF-SWIM lies in its ability to eliminate noisy data from the majority class. By eliminating these noisy data, the majority class will contain more accurate information, which could enhance the capability of the SWIM framework to generate synthetic data. By employing t-SNE analysis, we can determine the specific dataset categories in which NF-SWIM demonstrates the optimal performance.

Figure 7a–f shows the t-SNE plots of six imbalanced datasets that have various data distributions, particularly the loan data, Abalone19, Abalone9-18, winequality-white-3-9_vs_5, yeast-1-2-8-9_vs_7, and yeast-1-4-5-8_vs_7. Throughout our empirical analysis, we find that the NF-SWIM framework has an advantage on datasets in which the majority class has

multiple cluster points that are close together, e.g., see Figs. 7a–c. Additionally, Fig. 7d, e shows that datasets with a single cluster with closely located points are also particularly favorable for DBSCAN to detect noise effectively and improve the NF-SWIM performance. However, NF-SWIM undergeneralizes on the dataset with a single cluster and lower-density area, as shown in Fig. 7f. In this case, the DBSCAN algorithm may detect misleading noisy data, resulting in a substantial loss of majority class information. Although it is feasible to ease the DBSCAN parameter, we expect that such relaxation would frequently yield negative consequences.

5.6 Effect of hyperparameters on NF-SWIM

In this section, we examine the effect of hyperparameter changes on the variation in the performance of NF-SWIM. Our proposed framework relies on two parameter values related to the noise removal component, namely, $minPts$ and ϵ , which need to be appropriately adjusted to suit the given input dataset. As mentioned in Sect. 4.4, the $minPts$ value in this study is empirically set to 5. This $minPts$ value is then used as the k value of the k -distance graph to find the optimum ϵ . To check the impact of changing these parameters on the performance of NF-SWIM, we test the proposed framework under different settings.

Figure 8 shows the average G -mean plots with different values of $minPts$ and ϵ on three imbalanced datasets with various data distributions, particularly Abalone19, Winequality-white-3-9_vs._5, and Yeast-1-2-8-9_vs._7. Figure 8a–e shows the variation in the average G -mean with different values of $minPts$ [5, 10, 15, 20] on these datasets. When we test the effect of the $minPts$ values, we set the ϵ parameter for each dataset according to the optimum ϵ in Table 2. Conversely, we set the $minPts$ parameter as 5 when we test the effect of ϵ on the NF-SWIM performance. Figure 8b, d, f shows the variation in the average G -mean with different values of ϵ [0.1, 1.0]. Moreover, SVC with linear and RBF kernels is used to test the effect of changing the parameter settings on the performance of NF-SWIM.

Figure 8c, e shows a decreasing pattern of the average G -mean when a higher $minPts$ value is used. Meanwhile, Fig. 8b shows an insignificant change in the average G -mean using different $minPts$. Therefore, we recommend using a $minPts$ value of 5 for NF-SWIM. Too large a $minPts$ value may mislead the noise detection process, as the algorithms may mistakenly label important data as noise, which leads to the loss of valuable information from the majority class. Meanwhile, Fig. 8b, d, f shows some significant changes in the average G -mean when the ϵ parameter is changed. However, the change pattern of the average G -mean is inconsistent in this case. Therefore, determining the most suitable ϵ parameter for each dataset is crucial in NF-SWIM. In this study, the parameters $minPts$ and ϵ are manually determined for each dataset a process that is time-consuming and labor-intensive, particularly in optimizing ϵ for noise data detection in the majority class. The automation of these parameters presents a valuable direction for future research, potentially enhancing efficiency and effectiveness.

5.7 Impact of selective training partition

In this section, we investigate the effects of applying our framework to subsets of the data. Previously, we employed fivefold cross-validation to mitigate bias and overfitting during the training process, allocating 80% of the dataset for training and 20% for testing in each fold. The methodology was applied to the entire training set, constituting 80% of the total data.

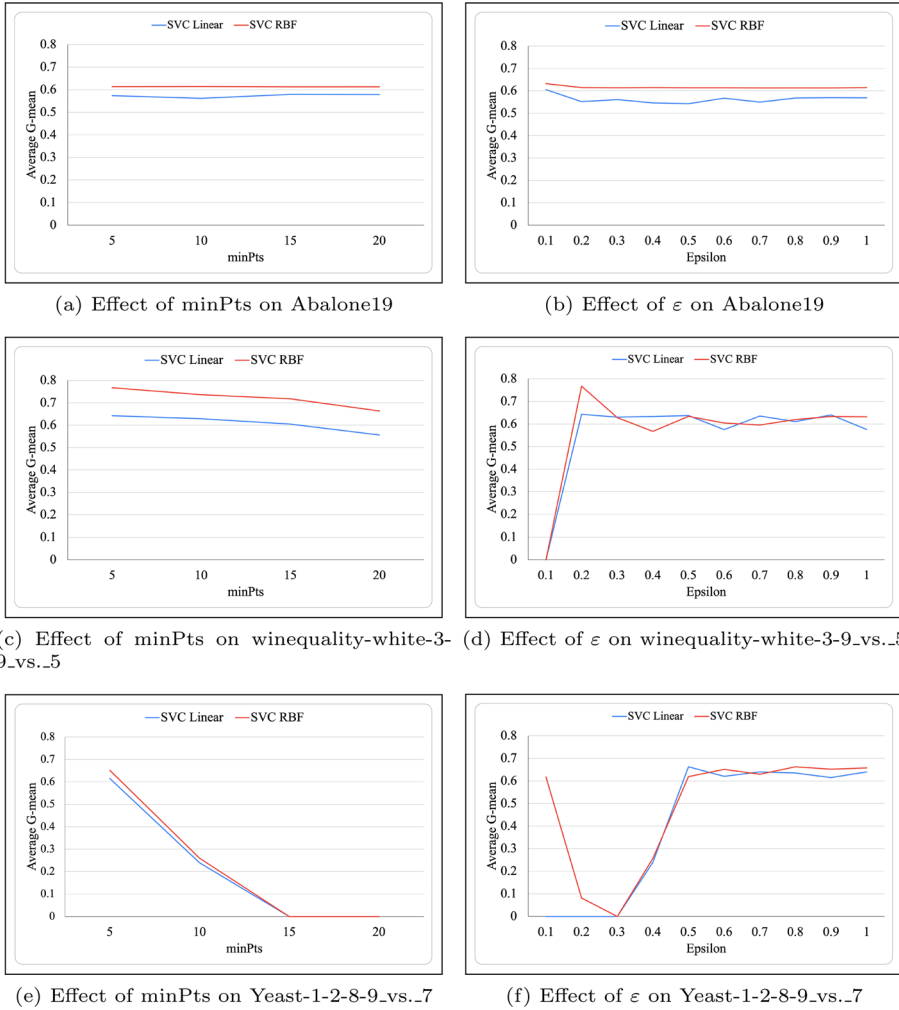


Fig. 8 Performance plots showing the relationship between the average G -mean and the hyperparameter values (minPts and ϵ) of NF-SWIM

However, we now evaluate the progression of the G -mean as our framework is applied to incremental portions of the training data, rather than its entirety.

We further explore the implications of NF-SWIM to understand the impact, potential advantages, or disadvantages of this selective application, especially for model training and validation. We divide the original training set into four equal segments. Subsequently, we train each of the eight classification methods examined in this study on these segments, comparing the predictive G -mean performance when utilizing NF-SWIM on 25%, 50%, 75%, and 100% of the training data. This evaluation is conducted on both the Abalone9-18 and Abalone19 datasets.

Figure 9 illustrates the predictive performance of each training partition for the eight classification models. The plots show that for most classifiers, the G -mean increases as the percentage of training data used grows from 25 to 100%. In Fig. 9a, for most classifiers,

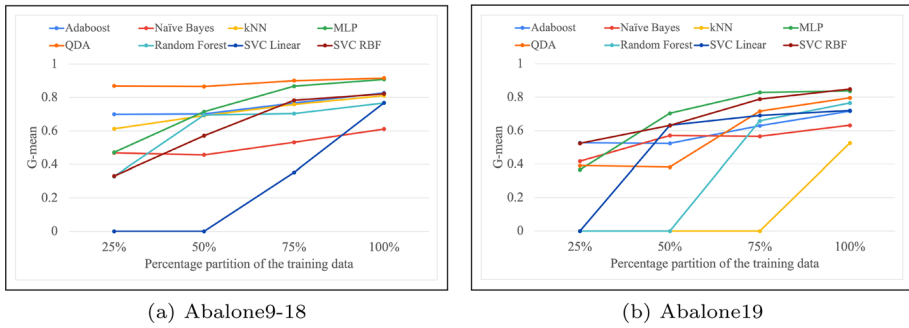


Fig. 9 Comparative predictive performance of NF-SWIM on eight classifiers across varying training data sizes

there is a marked improvement in performance as the training partition size increases. This is significant because it suggests that the models are able to utilize the additional balanced data to refine their decision boundaries and improve classification accuracy. From this plot, we can see that the degree of improvement varies among different classifiers. For instance, random forest and AdaBoost show a steady and significant increase in G-mean, indicating that these models benefit substantially from more data.

Figure 9b shows that the rate of improvement in G-mean varies between classifiers. Some, such as the random forest and AdaBoost, show a steep increase in G-mean as the percentage of training partition utilized using NF-SWIM increases from 25% to 50%, and continues to improve steadily up to 100%. While the G-mean generally improves with more balanced data, the rate of improvement is not constant. Additionally, this could indicate that models have varying degrees of sensitivity to the amount of training data. Overall, the plot underscores the critical role of sufficient balanced training data in developing effective machine learning models and suggests that the benefits of NF-SWIM utilized in the higher volume of training sets are significant, particularly when moving from smaller to larger partition training sets. As we increase the percentage of training data utilized by NF-SWIM, it leads to an enhanced G-mean, indicating better performance with more comprehensive balanced training data.

6 Conclusion

The problem of class imbalance has been acknowledged as a significant factor, leading to the degradation of the classification model performance. This problem occurs in various crucial domains where the minority class often holds greater significance, such as fraud detection, enterprise credit evaluation, disease diagnosis, image recognition, and failure detection. Another notable concern that often accompanies imbalanced data classification is the presence of noise, which can significantly reduce classifier performance. The presence of noise becomes a notable issue when the features of the majority data overlap with those of minority data, causing the shift of decision boundaries. To address this, we proposed a noise-free sampling with majority (NF-SWIM) by removing the noisy data from the majority class and then generating synthetic data for the minority class. Removing the noisy data from the majority class will yield more precise information on the majority class distribution. Consequently, the SWIM framework is able to generate more valuable and representative synthetic minority data. We implemented the DBSCAN clustering algorithm to detect and remove possibly

noisy data from the majority class and SWIM-MD to generate synthetic data for the minority class to obtain a balanced class distribution.

We compared and evaluated the proposed NF-SWIM with several existing sampling-based methods on a loan dataset and ten KEEL benchmark datasets. The experimental results indicated that our framework has the ability to enhance the predictive accuracy of eight different classifiers, with improvements ranging from 7.78% to 67.45% across the eleven datasets evaluated. The results also showed that the DBSCAN clustering algorithm can effectively identify potentially noisy data in the majority class, providing more precise information about the majority class. This enhanced the information of the majority class for the SWIM-MD, and led to a better representation of the synthetic data generated. Based on the percentage average G -mean values, the results showed that classifiers trained on balanced datasets using NF-SWIM outperformed other sampling-based methods. In general, there were statistically significant differences between the proposed NF-SWIM and the other sampling-based methods across all classifiers on most datasets.

Although our proposed framework is able to improve the classification performance, there are some limitations that require further investigation. First, the determination of the parameters $minPts$ and ϵ is done manually, which makes it time-consuming, as it requires substantial manual effort to find the best parameters. This is especially true for finding the best ϵ that detects noise in the majority class. An automated way of finding these parameters would significantly enhance the efficiency and accuracy of the framework, as well as reduce the workload for the user. Another point of improvement relates to the need to investigate noisy data in the minority class after synthetic data generation. A thorough examination, followed by the elimination of noise from both classes has the potential to improve the overall classification performance. Lastly, our framework could be expanded to function reliably in more diverse conditions, such as environments with higher noise levels and greater degrees of class overlap, a remaining key area for future research. Overcoming these limitations would not only increase the robustness of our method but also extend its applicability to a broader range of scenarios.

Declarations

Conflict of interest The authors declare that there is no conflict of interest.

References

1. Spelmen VS, Porkodi R (2018) A review on handling imbalanced data. In: 2018 International conference on current trends towards converging technologies (ICCTCT), pp. 1–11. <https://doi.org/10.1109/ICCTCT.2018.8551020>
2. Rekha G, Tyagi AK, Krishna Reddy V (2020) A novel approach to solve class imbalance problem using noise filter method. In: Abraham A, Cherukuri AK, Melin P, Gandhi N (eds) Intelligent systems design and applications. Springer, Cham, pp 486–496. https://doi.org/10.1007/978-3-030-16657-1_45
3. Li J, Zhu Q, Wu Q, Fan Z (2021) A novel oversampling technique for class-imbalanced learning based on SMOTE and natural neighbors. Inf Sci 565:438–455. <https://doi.org/10.1016/j.ins.2021.03.041>
4. Guzmán-Ponce A, Sánchez JS, Valdovinos RM, Marcial-Romero JR (2021) DBIG-US: A two-stage under-sampling algorithm to face the class imbalance problem. Expert Syst Appl 168:114301. <https://doi.org/10.1016/j.eswa.2020.114301>
5. Rezvani S, Wang X (2023) A broad review on class imbalance learning techniques. Appl Soft Comput 143:110415. <https://doi.org/10.1016/j.asoc.2023.110415>
6. Koziański M, Krawczyk B, Woźniak M (2019) Radial-based oversampling for noisy imbalanced data classification. Neurocomputing 343:19–33. <https://doi.org/10.1016/j.neucom.2018.04.089>

7. Liu J (2021) A minority oversampling approach for fault detection with heterogeneous imbalanced data. *Expert Syst Appl* 184:115492. <https://doi.org/10.1016/j.eswa.2021.115492>
8. Isangediok M, Gajamannage K (2022) Fraud detection using optimized machine learning tools under imbalance classes. <https://doi.org/10.48550/arXiv.2209.01642>
9. Sun J, Li J, Fujita H (2022) Multi-class imbalanced enterprise credit evaluation based on asymmetric bagging combined with light gradient boosting machine. *Appl Soft Comput* 130:109637. <https://doi.org/10.1016/j.asoc.2022.109637>
10. Teh K, Armitage P, Tesfaye S, Selvarajah D, Wilkinson ID (2020) Imbalanced learning: Improving classification of diabetic neuropathy from magnetic resonance imaging. *PLoS ONE* 15(12):1–15. <https://doi.org/10.1371/journal.pone.0243907>
11. Kumar V, Lalotra GS, Sasikala P, Rajput DS, Kaluri R, Lakshmana K, Shorfuzzaman M, Alsufyani A, Uddin M (2022) Addressing binary classification over class imbalanced clinical datasets using computationally intelligent techniques. *Healthcare*. <https://doi.org/10.3390/healthcare10071293>
12. Matsuoka D (2021) Classification of imbalanced cloud image data using deep neural networks: performance improvement. *Prog Earth Planet Sci* 8:68. <https://doi.org/10.1186/s40645-021-00459-y>
13. Xu Y, Li Y-L, Li J, Lu C (2022) Constructing balance from imbalance for long-tailed image recognition. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (eds.) *Computer Vision—ECCV 2022*, pp. 38–56. Springer, Cham. https://doi.org/10.1007/978-3-031-20044-1_3
14. Ahmed J, Green RC II (2022) Predicting severely imbalanced data disk drive failures with machine learning models. *Mach Learn Appl* 9:100361. <https://doi.org/10.1016/j.mlwa.2022.100361>
15. Pandey S, Kumar K (2023) Software fault prediction for imbalanced data: a survey on recent developments. *Proc Comput Sci* 218:1815–1824. <https://doi.org/10.1016/j.procs.2023.01.159>
16. Moniz N, Cerqueira V (2021) Automated imbalanced classification via meta-learning. *Expert Syst Appl* 178:115011. <https://doi.org/10.1016/j.eswa.2021.115011>
17. Saripuddin M, Suliman A, Syarmila Sameon S, Jorgensen BN (2022) Random undersampling on imbalance time series data for anomaly detection. In: *Proceedings of the 2021 4th international conference on machine learning and machine intelligence. MLMI '21*, pp. 151–156. Association for computing machinery, New York, NY, USA. <https://doi.org/10.1145/3490725.3490748>
18. García V, Sánchez JS, Marqués AI, Florencia R, Rivera G (2020) Understanding the apparent superiority of over-sampling through an analysis of local information for class-imbalanced data. *Expert Syst Appl* 158:113026. <https://doi.org/10.1016/j.eswa.2019.113026>
19. Fujiwara K, Huang Y, Hori K, Nishioji K, Kobayashi M, Kamaguchi M, Kano M (2020) Over- and undersampling approach for extremely imbalanced and small minority data problem in health record analysis. *Front Public Health*. <https://doi.org/10.3389/fpubh.2020.00178>
20. Santoso B, Wijayanto H, Notodiputro KA, Sartono B (2017) Synthetic over sampling methods for handling class imbalanced problems: a review. *IOP Conf Ser Earth Environ Sci* 58(1):012031. <https://doi.org/10.1088/1755-1315/58/1/012031>
21. Mohammed R, Rawashdeh J, Abdullah M (2020) Machine learning with oversampling and undersampling techniques: overview study and experimental results. In: *2020 11th International conference on information and communication systems (ICICS)*, pp. 243–248. <https://doi.org/10.1109/ICICS49469.2020.239556>
22. Wongvorachan T, He S, Bulut O (2023) A comparison of undersampling, oversampling, and SMOTE methods for dealing with imbalanced classification in educational data mining. *Information*. <https://doi.org/10.3390/info14010054>
23. Shamsudin H, Yusof UK, Jayalakshmi A, Akmal Khalid MN (2020) Combining oversampling and undersampling techniques for imbalanced classification: a comparative study using credit card fraudulent transaction dataset. In: *2020 IEEE 16th international conference on control & automation (ICCA)*, pp. 803–808. <https://doi.org/10.1109/ICCA51439.2020.9264517>
24. Park S, Park H (2021) Combined oversampling and undersampling method based on slow-start algorithm for imbalanced network traffic. *Computing* 103:401–424. <https://doi.org/10.1007/s00607-020-00854-1>
25. Bellinger C, Sharma S, Japkowicz N, Zaiane OR (2020) Framework for extreme imbalance classification: SWIM—sampling with the majority class. *Knowl Inf Syst* 62(3):841–866. <https://doi.org/10.1007/s10115-019-01380-z>
26. Firdausanti NA, Fatyanosa TN, Data M, Mendonça I, Aritsugi M (2022) Two-stage sampling: a framework for imbalanced classification with overlapped classes. In: *2022 IEEE international conference on big data (Big Data)* pp. 271–280. <https://doi.org/10.1109/BigData55660.2022.10020788>
27. Asniar Maulidevi NU, Surendro K (2022) SMOTE-LOF for noise identification in imbalanced data classification. *J King Saud Univ Comput Inf Sci* 34(6):3413–3423. <https://doi.org/10.1016/j.jksuci.2021.01.014>

28. Li J, Zhu Q, Wu Q, Zhang Z, Gong Y, He Z, Zhu F (2021) Smote-nan-de: addressing the noisy and borderline examples problem in imbalanced classification by natural neighbors and differential evolution. *Knowl-Based Syst* 223:107056. <https://doi.org/10.1016/j.knosys.2021.107056>
29. Hao S, Zhou X, Song H (2015) A new method for noise data detection based on DBSCAN and SVDD. In: 2015 IEEE International conference on cyber technology in automation, control, and intelligent systems (CYBER), pp. 784–789. <https://doi.org/10.1109/CYBER.2015.7288042>
30. Saeedi Emadi H, Mazinani SM (2018) A novel anomaly detection algorithm using DBSCAN and SVM in wireless sensor networks. *Wireless Pers Commun* 98:2025–2035. <https://doi.org/10.1007/s11277-017-4961-1>
31. Chen H, Yu G, Liu F, Cai Z, Liu A, Chen S, Huang H, Cheang CF (2020) Unsupervised anomaly detection via DBSCAN for KPIs jitters in network managements. *Comput Mater Cont* 62(2):917–927. <https://doi.org/10.32604/cmc.2020.05981>
32. Sheridan K, Puranik TG, Mangortey E, Pinon-Fischer OJ, Kirby M, Mavris DN. An application of DBSCAN clustering for flight anomaly detection during the approach phase. <https://doi.org/10.2514/6.2020-1851>
33. Wibisono S, Anwar MT, Supriyanto A, Amin IHA (2021) Multivariate weather anomaly detection using dbscan clustering algorithm. *J Phys Conf Ser* 1869(1):012077. <https://doi.org/10.1088/1742-6596/1869/1/012077>
34. Chandralekha HM C, PS N, PS SP, Ghosh MK (2022) Anomaly detection in recorded CAN log using DBSCAN and LSTM autoencoder. In: 2022 IEEE 3rd global conference for advancement in technology (GCAT), pp. 1–7. <https://doi.org/10.1109/GCAT55367.2022.9971885>
35. 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>
36. Han H, Wang W-Y, Mao B-H (2005) Borderline-smote: a new over-sampling method in imbalanced data sets learning. In: Huang D-S, Zhang X-P, Huang G-B (eds) *Advances in intelligent computing*. Springer, Berlin, pp 878–887
37. Tomek I (1976) Two modifications of CNN. *IEEE Trans Syst Man Cybern* 6(11):769–772. <https://doi.org/10.1109/TSMC.1976.4309452>
38. Wilson DL (1972) Asymptotic properties of nearest neighbor rules using edited data. *IEEE Trans Syst Man Cybern* 2(3):408–421. <https://doi.org/10.1109/TSMC.1972.4309137>
39. Batista GEAPA, Prati RC, Monard MC (2004) A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor Newsl* 6(1):20–29. <https://doi.org/10.1145/1007730.1007735>
40. Sasada T, Liu Z, Baba T, Hatano K, Kimura Y (2020) A resampling method for imbalanced datasets considering noise and overlap. *Proc Comput Sci* 176:420–429. <https://doi.org/10.1016/j.procs.2020.08.043>
41. Miranda ALB, Garcia LPF, Carvalho ACPLF, Lorena AC (2009) Use of classification algorithms in noise detection and elimination. In: Corchado E, Wu X, Oja E, Herrero Á, Baroque B (eds) *Hybrid artificial intelligence systems*. Springer, Berlin, Heidelberg, pp 417–424. https://doi.org/10.1007/978-3-642-02319-4_50
42. Puri A, Kumar Gupta M (2021) Knowledge discovery from noisy imbalanced and incomplete binary class data. *Expert Syst Appl* 181:115179. <https://doi.org/10.1016/j.eswa.2021.115179>
43. Fang X, Chong CF, Yang X, Wang Y (2022) Clustering algorithms based noise identification from air pollution monitoring data. In: 2022 IEEE Asia-pacific conference on computer science and data engineering (CSDE), pp. 1–6. <https://doi.org/10.1109/CSDE56538.2022.10089276>
44. Kotary DK, Nanda SJ (2021) A distributed neighbourhood DBSCAN algorithm for effective data clustering in wireless sensor networks. *Wireless Pers Commun* 121(4):2545–2568. <https://doi.org/10.1007/s11277-021-08836-y>
45. Schubert E, Sander J, Ester M, Kriegel HP, Xu X (2017) DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *ACM Trans Database Syst*. <https://doi.org/10.1145/3068335>
46. 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-Val Logic Soft Comput* 17(2–3):255–287
47. Santos MS, Abreu PH, Japkowicz N, Fernández A, Soares C, Wilk S, Santos J (2022) On the joint-effect of class imbalance and overlap: a critical review. *Artif Intell Rev* 55:6207–6275. <https://doi.org/10.1007/s10462-022-10150-3>
48. Patro SGK, Sahu KK (2015) Normalization: a preprocessing stage. <https://doi.org/10.48550/arXiv.1503.06462>
49. Swana EF, Doorsamy W, Bokoro P (2022) Tomek link and smote approaches for machine fault classification with an imbalanced dataset. *Sensors*. <https://doi.org/10.3390/s22093246>

50. Tonini M, Abellan A (2014) Rockfall detection from terrestrial LiDAR point clouds: a clustering approach using R. *J Spat Inf Sci* 8:95–110. <https://doi.org/10.5311/JOSIS.2014.8.123>
51. Starczewski A, Goetzen P, Er MJ (2020) A new method for automatic determining of the dbscan parameters. *J Artif Intell Soft Comput Res* 10(3):209–221. <https://doi.org/10.2478/jaiscr-2020-0014>
52. Bessrouer M, Elouedi Z, Lefèvre E (2020) E-DBSCAN: An evidential version of the DBSCAN method. In: 2020 IEEE Symposium series on computational intelligence (SSCI), pp. 3073–3080. <https://doi.org/10.1109/SSCI47803.2020.9308578>
53. McKinney: data structures for statistical computing in python. In: Walt, Millman (eds.) Proceedings of the 9th python in science conference, pp. 56–61 (2010). <https://doi.org/10.25080/Majora-92bf1922-00a>
54. Harris CR, Millman KJ, Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R, Picus M, Hoyer S, Kerkwijk MH, Brett M, Haldane A, Río JF, Wiebe M, Peterson P, Gérard-Marchant P, Sheppard K, Reddy T, Weckesser W, Abbasi H, Gohlke C, Oliphant TE (2020) Array programming with NumPy. *Nature* 585(7825):357–362. <https://doi.org/10.1038/s41586-020-2649-2>
55. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey CJ, Polat İ, Feng Y, Moore EW, VanderPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, van Mulbregt P, SciPy 1.0 Contributors (2020) SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat Methods* 17:261–272. <https://doi.org/10.1038/s41592-019-0686-2>
56. Hunter JD (2007) Matplotlib: A 2d graphics environment. *Comput Sci Eng* 9(3):90–95. <https://doi.org/10.1109/MCSE.2007.55>
57. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in python. *J Mach Learn Res* 12:2825–2830
58. 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
59. He H, Bai Y, Garcia EA, Li S (2008) ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In: 2008 IEEE International joint conference on neural networks (ieee world congress on computational intelligence), pp. 1322–1328. <https://doi.org/10.1109/IJCNN.2008.4633969>
60. Han H, Wang W-Y, Mao B-H (2005) Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In: Huang D-S, Zhang X-P, Huang G-B (eds) *Adv Intell Comput*. Springer, Berlin, Heidelberg, pp 878–887
61. Zhang J, Mani I (2003) kNN approach to unbalanced data distributions: a case study involving information extraction. In: Proceedings of the ICML'2003 workshop on learning from imbalanced datasets
62. Laurikkala J (2001) Improving identification of difficult small classes by balancing class distribution. *Artif Intell Med* 21(1):63–66. https://doi.org/10.1007/3-540-48229-6_9
63. Scheff SW (2016) Chapter 8 - nonparametric statistics. In: Scheff, S.W. (ed.) *Fundamental Statistical Principles for the Neurobiologist*, pp. 157–182. <https://doi.org/10.1016/B978-0-12-804753-8.00008-7>. <https://www.sciencedirect.com/science/article/pii/B9780128047538000087>
64. Xia Y (2020) Chapter eleven - correlation and association analyses in microbiome study integrating multiomics in health and disease. In: Sun, J. (ed.) *The Microbiome in health and disease*. Progress in Molecular Biology and Translational Science, vol. 171, pp. 309–491. <https://doi.org/10.1016/bs.pmbts.2020.04.003>. <https://www.sciencedirect.com/science/article/pii/S1877117320300478>

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

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.



Neni Alya Firdausanti is a dedicated and emerging academic in the field of statistics and computer science. Currently serving as a junior lecturer in the Statistics Department at Sepuluh Nopember Institute of Technology (ITS) in Indonesia, Neni is also pursuing a PhD degree in Computer Science and Communication Engineering at Kumamoto University, Japan. Throughout her academic career, Neni has developed a keen interest in the intersection of machine learning and data mining, a passion that has led to her co-authoring 10 articles published in international conferences and journals.



Israel Mendonça received the BS degree in computer science from State University of Rio de Janeiro, Brazil, and the MS degree in system's engineering from Federal University of Rio de Janeiro, Brazil, in 2011 and 2014, respectively. He received his PhD degree from Kyushu University, Japan, in 2019. From 2019 to 2020, he worked as an AI researcher for Team AIBOD, Japan. Since the end of 2020, he has been an assistant professor at Kumamoto University. His research interests include machine-learning methods and applications, image processing, and combinatorial optimization.



Masayoshi Aritsugi received the BE and DE degrees in computer science and communication engineering from Kyushu University, Japan, in 1991 and 1996, respectively. From 1996 to 2007, he was with the Department of Computer Science, Gunma University, Japan. Since 2007, he has been a Professor with Kumamoto University, Japan. His research interests include database systems and parallel/distributed data processing. Prof. Aritsugi is a Senior Member of IPSJ and IEICE and a member of ACM, IEEE, and DBSJ. He was a recipient of the COMP-SAC 2015 Best Paper Award, the Best Paper Award in Image Processing and Understanding in 13th IEEE International Conference on Signal Processing (ICSP2016), and the Best Paper Award in the 2019 IEEE International Cyber Science and Technology Congress (Cyber-SciTech).