Intelligent Systems with Applications 22 (2024) 200357

Contents lists available at ScienceDirect

Intelligent Systems with Applications

journal homepage: www.journals.elsevier.com/intelligent-systems-with-applications

Cluster-based oversampling with area extraction from representative points for class imbalance learning

Zakarya Farou [a,*], Yizhi Wang [a], Tomáš Horváth [b,c]

[a] ELTE – Eötvös Loránd University, Faculty of Informatics, Institute of Industry-Academia Innovation, Department of Data Science and Engineering, Pázmány Péter sétány 1/C, H-1117, Budapest, Hungary
[b] Edinburgh Napier University, School of Computing, Engineering & Built Environment, 10 Colinton Road, EH10 5DT Edinburgh, Scotland, UK
[c] Pavol Jozef Šafárik University, Faculty of Science, Institute of Computer Science, Jesenná 5, 040 01 Košice, Slovakia

ARTICLE INFO

Dataset link: https://github.com/ghostqriver/AROSS

Keywords:
Class imbalance
Adaptive oversampling
Clustering analysis
Predicting benchmark
k-Nearest neighbor

ABSTRACT

Class imbalance learning is challenging in various domains where training datasets exhibit disproportionate samples in a specific class. Resampling methods have been used to adjust the class distribution, but they often have limitations for small disjunct minority subsets. This paper introduces AROSS, an adaptive cluster-based oversampling approach that addresses these limitations. AROSS utilizes an optimized agglomerative clustering algorithm with the Cophenetic Correlation Coefficient and the Bayesian Information Criterion to identify representative areas of the minority class. Safe and half-safe areas are obtained using an incremental k-Nearest Neighbor strategy, and oversampling is performed with a truncated hyperspherical Gaussian distribution. Experimental evaluations on 70 binary datasets demonstrate the effectiveness of AROSS in improving class imbalance learning performance, making it a promising solution for mitigating class imbalance challenges, especially for small disjunct minority subsets.

1. Introduction

Data is crucial in various domains, powering innovation and decision-making in healthcare (Cios & Moore, 2002), finance (Lusardi & Mitchell, 2014), education (Zoric, 2019), and technology (Merrild et al., 2008). In healthcare, patient records and medical data support precise diagnoses and research. Finance uses data analytics for investment, risk management, and fraud detection. Educational institutions enhance learning with data, while technology companies improve user experiences and refine products. Data's value is essential in optimizing processes, predicting trends, and driving progress across diverse fields. In the dynamic landscape of machine learning (ML), the importance of data cannot be overstated. The success of ML algorithms relies heavily on the richness and balance of the datasets they are trained on. However, the persistent challenges of imbalanced datasets and limited data availability underscore the critical need for innovative solutions in the dynamic landscape of ML, especially that the problem of class imbalance is prevalent in many domains, where the dataset is characterized by a disproportionate number of samples belonging to a specific class. This imbalance can occur naturally, as seen in rare disease detection and credit card fraud scenarios, or unnaturally due to challenges in acquiring minority-class data, such as privacy constraints. Addressing ML challenges under class imbalance, referred to as class imbalance learning (CIL), has been a subject of extensive research. Researchers have proposed various strategies to tackle CIL, which can be broadly classified into algorithm-level and data-level methods. Algorithm-level methods, such as cost-sensitive learning (Thai-Nghe et al., 2010) and ensemble learning (Bi & Zhang, 2018), aim to adjust classifiers to prioritize accurate classification of the minority class. However, these methods often require domain-specific knowledge and rely on specific learning algorithms. As an example, Hazarika and Gupta (2021, 2022, 2023), Hazarika et al. (2023) proposed several algorithmic level solutions based on Support Vector Machines (SVM) to address class imbalance problems. Despite their computational power, the reliance on SVM intricacies may introduce unnecessary complexity for small datasets, where simpler, interpretable classification algorithms are often more suitable.

In contrast, data-level solutions focus on modifying the class distribution to create a balanced training set, enabling their application across multiple classifiers. Moreover, data-level solutions enhance the dataset's diversity, providing the predictive models with richer information and contributing to a more robust and generalized learning

* Corresponding author.
E-mail addresses: zakaryafarou@inf.elte.hu (Z. Farou), srgfxm@inf.elte.hu (Y. Wang), T.Horvath@napier.ac.uk, Tomas.Horvath@upjs.sk (T. Horváth).

https://doi.org/10.1016/j.iswa.2024.200357
Received 7 August 2023; Received in revised form 9 February 2024; Accepted 8 March 2024
Available online 16 March 2024

experience, particularly crucial in scenarios of class imbalance. As a result, these solutions have gained attention for their potential to address CIL challenges effectively. By adjusting the class distribution through techniques such as oversampling (Gosain & Sardana, 2017) or undersampling (Liu & Tsoumakas, 2020), data-level methods aim to enhance the performance of classification algorithms in imbalanced scenarios.

One approach that has attracted significant interest in recent years is cluster analysis, which is well-suited for identifying data distributions for resampling purposes (Jiang, Zhao, et al., 2023, Lu et al., 2022). In these methods, majority and minority samples are assigned to distinct clusters, and either oversampling or undersampling is performed within these clusters to achieve a balanced class distribution. Most existing cluster-based resampling methods utilize unsupervised clustering on labeled data. However, the absence of class information can hinder the clustering process in finding the optimal solution for resampling and subsequent classification. Challenges arise in selecting the appropriate number of clusters or determining the suitable linkage for hierarchical clustering in cluster-based methods. To overcome these limitations, leveraging unsupervised clustering with the support of prior knowledge embedded in labeled data can effectively improve clustering performance. Nonetheless, this potential remains relatively unexplored in CIL, with limited existing solutions (Douzas & Bacao, 2017, Jiang, Zhao, et al., 2023, Wang et al., 2023, Zhang et al., 2017).

This paper introduces an adaptive cluster-based resampling method, named AROSS, that runs an agglomerative clustering on labeled data to discover sub-spaces of the minority class for efficient resampling. While optimization is crucial for AROSS's performance, it is essential to clarify that the optimization of execution time is beyond the scope of this research. Primarily, we propose an optimized agglomerative clustering that incorporates the usage of cophenetic correlation coefficient (CPCC, described in Farris (1969)) as a linkage validation metric and Bayesian information criterion (B_{score}, described in Schwarz (1978)) for cluster optimization. CPCC measures how well a dendrogram describes the pairwise distances between the data points, while B_{score} specifies the optimal number of clusters. The proposed approach aims to oversample sub-spaces of the minority class, pure, i.e., safe clusters containing labeled data from the majority class are not actively utilized during the oversampling process, but instead, they are ignored, while a set of representative points symbolizes the remaining clusters. Then the proposed incremental k-Nearest Neighbor (k-NN) strategy is used to substitute the clusters with so-called *half-safe areas* and *safe areas*. A generator following a truncated hyperspherical Gaussian distribution is utilized to sample data using a Gaussian function to oversample these regions. The proposed oversampling considers the specific distribution of the minority class within the small disjunct areas and allows for more fine-grained control over the generation of synthetic samples. Moreover, the suggested approach can be easily implemented with existing classification algorithms, making it suitable for a wide range of class-imbalanced tasks.

The proposed method is compared with state-of-the-art resampling methods on 66 real-world and 4 synthetic datasets. The results of an extensive experiment demonstrate that the proposed approach has competitive or significant superiority over the compared baseline methods. The contributions of this paper are the following:

- A novel adaptive cluster-based resampling method is presented that introduces an optimized agglomerative clustering algorithm to extract the representative areas within the minority class and efficiently capture the underlying data distribution to facilitate the resampling process.
- An incremental kNN approach is introduced, a strategy to estimate an area from cluster's sub-space starting from a representative point to substitute clusters with so-called half-safe and safe areas.
- An oversampling approach that utilizes a truncated hyperspherical Gaussian distribution to sample data for the minority class is presented. Unlike conventional interpolation-based oversampling methods, our approach ensures a more uniform distribution of generated instances around representative points, which serve as centroids for specific areas. This novel approach enhances the effectiveness of oversampling by promoting a more balanced and representative augmentation of the training set.
- Extensive experiments on 70 datasets are conducted to show the effectiveness of the proposed cluster-based resampling method. The result demonstrates that AROSS can improve the performance of CIL algorithms.

The rest of this paper is organized as follows: Section 2 describes problems related with CIL and proposes a taxonomy for data-level resampling methods. After illustrating the general structure of the proposed solution, Section 3 presents the optimized agglomerative clustering algorithm, the extraction and classification of minority class representative areas, and how these areas are populated. The experimental settings and empirical results of the proposed algorithm against CIL baseline are reported in Section 4 and 5 respectively. Finally, we conclude the paper and describe possible future work in Section 6.

2. Basic concepts and literature review

In this chapter, the main concepts related to the presented work are introduced, together with literature review categorized by different CIL approaches.

Definition 1 *(Learning – classification).* Let $\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid i \in \{1, \dots, n\}\}$ be a set of n labeled training examples, where, without loss of generality, $\mathbf{x}_i \in \mathbb{R}^d$ is the i^{th} input and $y_i \in \mathbb{Z}$ is the corresponding output. A classification algorithm takes $\mathcal{D}$ as input and learns a function $f : \mathbb{R}^d \to \mathbb{R}$ that minimizes a specified loss function $L(y_i, f(\mathbf{x}_i))$ over the training data. The learned function $f(\mathbf{x})$ can then be used to predict the output y for a new, unseen input $\mathbf{x}$.

Definition 2 *(Class imbalance – imbalance ratio).* Let $\mathcal{D} = \mathcal{P} \cup \mathcal{N}$ be a binary labeled dataset, i.e. $\mathcal{P} = \{(\mathbf{x}_i, y_i) \mid i \in \{1, \dots, n\}, y_i = +1\}$ and $\mathcal{N} = \{(\mathbf{x}_i, y_i) \mid i \in \{1, \dots, n\}, y_i = -1\}$. Without loss of generality, $\mathcal{P}$ is called the minority class while $\mathcal{N}$ is called the majority class. If $|\mathcal{P}| \ll |\mathcal{N}|$ then $\mathcal{D}$ is called imbalanced. Class imbalance is measured by the imbalance ratio (Cordón et al., 2018).

$$IR = \frac{|\mathcal{P}|}{|\mathcal{N}|} \tag{1}$$

2.1. Data intrinsic characteristics in CIL

CIL comprises several learning barriers, including small sample size, class overlapping, within-class imbalance, and imbalanced distribution, explained in detail in the following sections.

2.1.1. Small sample size and imbalanced class distribution

When the training set has a fair amount of data, the classification algorithm can learn the underlying patterns and affinities in the data that are essential for accurate predictions. With more additional data, the algorithm can capture a broader range of variations in the data, leading to a more robust model that can better generalize to new examples. Furthermore, adequate training sample size reduces the risk of overfitting, which occurs when the algorithm learns the noise or random variations in the training data instead of the underlying patterns, leading to poor performance on new, unseen data.

Recently, we observed that Deep Neural Networks (DNNs) had achieved great success in solving diverse ML problems (Alshemali & Kalita, 2020), mainly for a unique reason, i.e., the availability of a massive number of samples (Big Data). Therefore, insufficient data and a lack of instances are commonly linked to CIL. Thus, it will be challenging to discover regularities and pattern uniformity, especially in the minority class. For imbalanced datasets, the imbalance class distribution

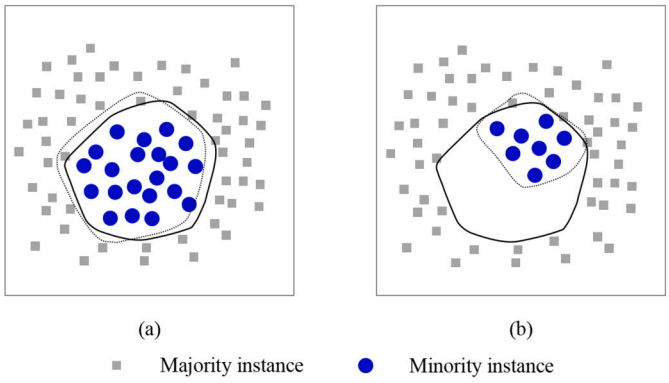

Fig. 1. Small sample size effect in imbalance problem; Real (solid line) and estimated (dashed line) decision boundaries while using appropriate number of samples from the minority class (a) and while using insufficient number of samples from the minority class (b).

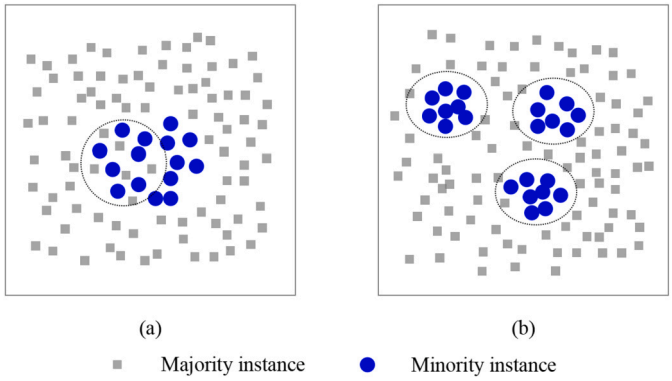

Fig. 2. Imbalance learning problems; (a) Class overlapping (b) Within class imbalance.

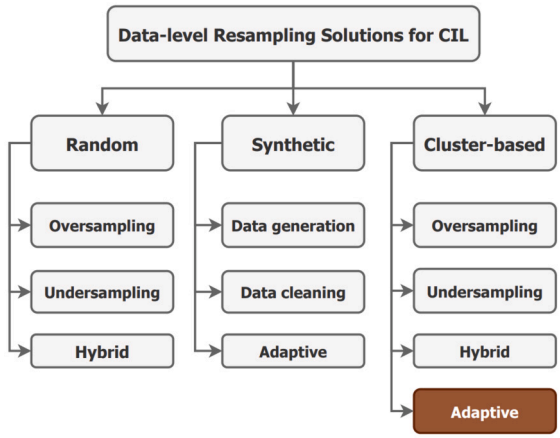

Fig. 3. Proposed taxonomy for Data-level resampling solutions for CIL.

further affects the training process as the minority class is underrepresented, especially in the case of small disjoint minority class subsets. Imbalance class distribution directly connects with the *IR* (Eq. (1)).

Shi et al. (2022) pointed out that training sets with a nearly balanced class distribution, i.e., $IR \in [0.7, 1.5]$, generally give better results. Fig. 1 emphasizes that a small number of samples in $\mathcal{P}$ and, thus, a high *IR* influences the classification performance. Indeed, Fig. 1(a) reveals that using a reasonable number of samples referring to $\mathcal{P}$ will enable a given classifier to estimate decision boundaries (dashed line) that are very close to the expected decision boundaries (solid line). However, Fig. 1(b) reveals that the insufficient number of samples in $\mathcal{P}$ samples have infected the estimated decision boundaries (dashed line). The shortage in training data did not help to learn the natural boundaries between classes, thus, classifiers' performance will deteriorate.

It is often assumed that achieving a balanced class distribution (with *IR* close to 1) would lead to favorable results in CIL. Consequently, we might think that generating additional samples for $\mathcal{P}$ would be a solution for CIL challenges. However, it is important to note that *IR*'s tolerability depends on the specific characteristics of the dataset. While having a larger number of training samples is generally beneficial, simply increasing the sample size or synthesizing new data will not address all the complexities associated with class imbalance. Other issues, such as class overlapping or the presence of small disjunct subsets, can still pose challenges. Therefore, when considering data resampling techniques, it is crucial to consider all the intricacies and problems related to CIL.

2.1.2. Class overlapping and within class imbalance

Class overlapping as described in Fig. 2(a), is a common problem while dealing with imbalanced datasets. Indeed, in real-life applications, we rarely encounter a dataset where class instances are linearly separable. Some classification methods, such as Support Vector Machines (SVMs), apply kernel functions[1] to solve the linearity issue, yet, the lack of the data persists and penalizes classifiers' performances. Other studies, such as Shi et al. (2022), used resampling algorithms based on sample concatenation (Re-SC). Re-SC transforms an imbalanced training dataset in the original sample space into a concatenated dataset in a new sample space. In the transformation process, Re-SC considers both the distribution of the original dataset and that of the majority samples, thereby alleviating the loss of valuable samples and reducing the class overlapping region. However, the non-linearity of the data is not the only problem that ruined the separability among classes. Another issue is known as small disjoint subsets or within class imbalance problem, illustrated in Fig. 2(b), which would appear when a minority class $\mathcal{P}$ is compromised of s dispatched subsets $\mathcal{P}_i$ of different sizes, where $\bigcap_{i=1}^{s} \mathcal{P}_i = \emptyset$, $\bigcup_{i=1}^{s} \mathcal{P}_i = \mathcal{P}$ and $s > 1$.

Hardly separating $\mathcal{P}$ from $\mathcal{N}$ is a critical factor; it complicates the learning process of small classes, which is, as previously mentioned, the target classes. It will likely increase the complexity of the problem, and thus, the performance of the learning method might decrease dramatically. In such scenarios, supervised methods cannot generate discriminating patterns to correctly separate $\mathcal{P}$ from $\mathcal{N}$. As instances from $\mathcal{N}$ are present in huge numbers, standard supervised methods that try to maximize classification's performance will treat $\mathcal{P}$ instances belonging to the overlapping region, as illustrated in Fig. 2(b), as noise (Weiss, 2004). As a result, the overlapping zone as well as the underrepresented disjoint subsets $\mathcal{P}_i \in \mathcal{P}$ will be misclassified due to the bias toward $\mathcal{N}$, which will make the recognition of instances from the target class even harder.

2.2. Data-level resampling methods for CIL

As illustrated in Fig. 3, we categorize data-level solutions into random, synthetic, and cluster-based sampling techniques.

2.2.1. Random sampling

Random sampling approaches are non-heuristic methods that seek to balance class distribution via the arbitrary replication of the samples from $\mathcal{P}$ (random oversampling, i.e., ROS), the random dismissal of samples from $\mathcal{N}$ (random oversampling, i.e., RUS), or by hybridization between ROS and RUS (Wongvorachan et al., 2023). Miscellaneous studies consider that ROS can raise the likelihood of occurring overfitting. Chawla (2010) has pointed out that ROS might lead to a smaller

[1] Projection of data into a higher dimensional space (and distance computation within).

and more specific decision region by simply replicating minority samples, especially on those classifiers based on error-based objective functions. In contrast, the significant drawback of RUS is that it may discard essential data for the induction process (Batista et al., 2005, Santoso et al., 2017). Hybridization of ROS and RUS originate from the mentioned drawbacks. It pursues to trade-off between removing instances from $\mathcal{N}$ and replicating others from $\mathcal{P}$ to reach the best possible performance in any imbalanced dataset.

2.2.2. Synthetic sampling

In synthetic sampling, we create new samples using existing data points to expand $|\mathcal{P}|$ using data generation. There are several techniques for data generation, including synthetic minority oversampling (SMOTE), introduced by Chawla et al. (2002), which is the traditional data resampling method.

In SMOTE, additional minority samples are created along the line segment among the minority samples, although with no indication of any kind to the samples available in the confrontational majority class. SMOTE has been applied in various domains, including finance (Sun et al., 2020), fraud detection (Xia et al., 2023), medical diagnosis (Bokhare et al., 2023, Kamarulzalis et al., 2018) and image classification (Khan & Sheikh, 2023). It has shown promising results in improving the classification accuracy of models in these domains. However, SMOTE has some limitations, mainly when dealing with small disjunct subsets and overlapping regions, where the existing examples may not represent the actual distribution of $\mathcal{P}$. In such cases, SMOTE would probably produce noisy examples, leading to overfitting and decreasing the model's performance. Furthermore, its narrow sample generation range may cause the over-dense of synthetic samples on a line segment, meaning that the overall distribution of $\mathcal{P}$ is not uniform, especially when the *IR* is large, or the number of nearest neighbors within the clusters is too small.

To overcome these limitations, researchers have proposed various modifications to SMOTE to improve the quality of synthetic examples and reduce the risk of overfitting. Borderline-SMOTE1 and Borderline-SMOTE2 methods, introduced by Han et al. (2005), prioritize the class boundary's vicinity for classifier construction. Zhang and Li (2014) proposed a random walk oversampling (RWO) that generates diverse and realistic synthetic samples by employing random walks in the feature space. In contrast, Sandhan and Choi (2014) introduced partially guided oversampling (GS) that extracts linear and nonlinear patterns from the minority class to guide the random imputation process and generate synthetic samples in each feature dimension. Additionally, Rivera (2017) introduced noise reduction apriori synthetic oversampling (NRAS). NRAS incorporates propensity scores as additional features to improve the selection of nearest neighbors and reduce noise. These techniques enhance the quality and diversity of synthetic samples, leading to improved handling of class imbalance. In addition to these techniques, adaptative solutions such as adaptive synthetic sampling (AS, He et al. (2008)), deterministic SMOTE (SD, Torres et al. (2016)), adaptive neighbor synthetic minority oversampling (ANS, Siriseriwan and Sinapiromsaran (2017)) and synthetic minority based on the probabilistic distribution (SMPD, Kunakorntum et al. (2020)) improve SMOTE according to specific criteria and only generate synthetic samples with adaptive weights in particular regions that are considered helpful for the learning algorithm. However, similar to SMOTE, adaptative synthetic sampling solutions cannot efficiently target the within-class imbalance issue. Data cleaning solutions such as SMOTE-Tomeklink (STL, Swana et al. (2022)) and SMOTE with Wilson's edited nearest neighbor rule (SENN, Parthasarathy et al. (2023)) improve the quality of augmented data with a post-processing mechanism that would remove noisy, ambiguous or wrongly located samples. It is important to note that STL and SENN are extensions of the Tomeklink (TL, Tomek (1976)) and the Wilson's edited nearest neighbor rule (ENN, Wilson (1972)) undersampling approaches. However, TL and ENN may not be effective when $\mathcal{P}$ and $\mathcal{N}$ overlap, as TL can remove correctly classified examples but close to the decision boundary. In contrast, ENN may remove correctly classified instances but have a high-density overlap with $\mathcal{N}$. Therefore, synthetic sampling with data cleaning has limited applicability as it may not be effective for all class imbalance problems, particularly those with small disjunct minority class subsets.

2.2.3. Cluster-based sampling

Cluster-based sampling methods are commonly used to uncover the underlying data structure in resampling techniques (Jiang, Zhao, et al., 2023) and are more effective than random or synthetic sampling for identifying $P_i \in \mathcal{P}$. Several approaches have been proposed to address CIL, for instance, Jo and Japkowicz (2004) applied k-means clustering beforehand, assuming clusters $\mathcal{N}$ and $\mathcal{P}$ as disjunct subsets. Then, replicated samples in clusters to make the data size of subsets within each class and between classes consistent. Similarly, Cieslak et al. (2006) adopted cluster-SMOTE to resample $\mathcal{P}$. Furthermore, Douzas et al. (2018) proposed an adaptive approach based on k-means and SMOTE, namely kmeans-SMOTE (KS). KS clusters training samples and measures the sparsity for clusters dominated by $\mathcal{P}$. SMOTE is applied in those clusters, and the number of synthetic samples generated in each cluster depends on the minority samples' sparsity so that minority distribution can be compensated more in sparse clusters. Compared to conventional approaches, the experiments prove the effectiveness of the method, especially for within-class imbalance. However, k-means clustering is sensitive to k, which the authors did not give a feasible optimization solution in their proposed approach. Another issue with k-means is that it is good at capturing equally sized-spherical clusters. Thus it may inhibit the detection capability of disjoints with various shapes and sizes. Unlike previous methods which use k-means for clustering, Bunkhumpornpat et al. (2012) proposed DB-SMOTE (DBS) to cluster $\mathcal{P}$ and exclude noises by DBSCAN (Ester et al., 1996) and perform oversampling along the shortest density-reachable path from each minority instance to the most central instance of the cluster. DBS assumes well-separated minority instances and may generate synthetic samples in non-representative regions when there is class overlap. Another adaptive approach, the self-organizing map oversampling (SOMO), introduced by Douzas and Bacao (2017), transforms the training data into a two-dimensional space using self-organizing map (SOM). Each map of SOM is considered a cluster. Then for clusters dominated by $\mathcal{P}$, SMOTE is applied within and between neighboring clusters by the weight, where the weight is inversely proportional to $\mathcal{P}$'s density, such that synthetic samples will be generated by a greater weight in sparser areas. While SOMO has shown promising results, it may not be a good fit when $\mathcal{P}$ has a poor representation which is the case in small disjunct problems, or the dataset is non-linearly separable. As SOMO assumes a linear relationship between examples, it may not be able to capture non-linear decision boundaries. Adaptive semi-unsupervised weighted oversampling (ASWO, Nekooeimehr and Lai-Yuen (2016)) combines oversampling with semi-unsupervised learning but may not effectively balance small disjunct minority classes. The reason is that ASWO relies on density distributions to specify the weights for oversampling. However, in small disjunct minority class problems, the density distribution may be very different from that of larger, more overlapping minority classes, which can make it difficult for ASWO to identify and accurately generate examples for $\mathcal{P}$. In their study, Yang and Cha (2021) proposed GMOTE to address SMOTE's overfitting issue. GMOTE uses distribution-based generators instead of linear interpolation and employs a Gaussian Mixture Model (GMM) to estimate the minority class distribution. However, experimental results showed that GMOTE's oversampling has limited effectiveness for small disjunct minority class subsets and could lead to the misclassification of nearby majority instances. Last but not least, Ma and Fan (2017) introduced CURE-SMOTE (CS) that integrates the CURE algorithm (Guha et al., 1998) to generate representative clusters and then applies SMOTE to oversample $\mathcal{P}$ within each cluster. Using the CURE algorithm seeks to create more representative clusters than traditional clustering methods, which would help capture the underlying

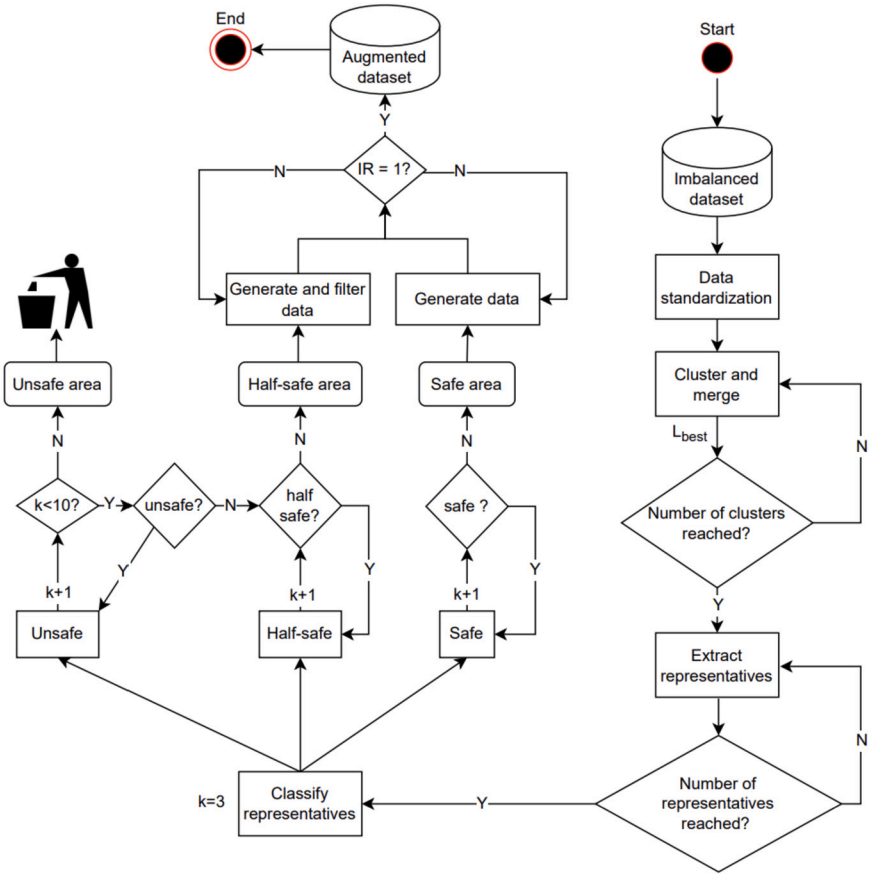

Fig. 4. The AROSS training process.

structure of $\mathcal{P}$ more accurately. However, the authors did not assume linkage validation and used single linkage (S-Link) to extract clusters. The main side-effect of S-Link is the potential of building long-chained data points, also known as chaining (Seifoddini, 1989). Chaining occurs when we join clusters using S-Link based on the similarity between their closest points, which may result in long, narrow chains of points, which would introduce new and none existing patterns and relationships in the original data. Also, they set static values for the number of clusters and representative points. These values can lead to sub-optimal performance for some datasets, leading to a poor exploration of $\mathcal{P}$ sub-spaces, especially when we have a within-class problem. As a result, CS would overpopulate some regions in $\mathcal{P}$ and leave others underrepresented.

Considering CS advantages and disadvantages, designing a new cluster-based sampling approach called AROSS is proposed in this article, which incorporates linkage determination, optimized number of clusters, and representative points selection to enhance clustering accuracy and address CIL challenges.

3. Area-based representative points oversampling with shifting (AROSS)

Let D be a binary labeled, imbalanced dataset, with its minority and majority classes $\mathcal{P}$ and $\mathcal{N}$, respectively, as introduced in Definition 2.

As represented in Fig. 4, AROSS starts by standardizing the dataset using z-score normalization. After that, D is clustered into c clusters $C_1, C_2, \ldots, C_c$, i.e. $D = C_1 \cup C_2 \cup \cdots \cup C_c$, using the given optimized agglomerative clustering. Then the representative points $\mathcal{R}_i$ from each cluster C_i are extracted, forming the set $\mathcal{R}$ of all representative points, i.e. $\mathcal{R} = \bigcup_{i=1}^{c} \mathcal{R}_i$. Then, clusters are substituted with *safe* and *half-safe* areas given as output from the incremental kNN algorithm. After that, AROSS generates samples inside these areas using the Gaussian generator. The augmented dataset comprises the original and the synthetic data generated by AROSS.

The modules and implementation steps of AROSS are described in details in the following sub-sections.

3.1. Clustering using optimized agglomerative clustering

As formerly mentioned in Sect. 2.1.2, the small disjuncts imbalance problem emerges when there are one or more sub-classes with very few examples dissimilar from those in other sub-classes but all of which characterize the minority class. To address this case, one could use clustering, a method that tends to group similar instances into clusters, to locate these small disjuncts. One of the clustering strategies, belonging to hierarchical clustering techniques and producing interpretable results, is agglomerative clustering (AC, Lukasová (1979)). AC begins with each data object[2] $(\mathbf{x}_i, y_i) \in D$ as a standalone cluster $C_i = \{(\mathbf{x}_i, y_i)\}$ and recursively merges the two closest clusters C_j and C_l, $1 \leq i, j, l \leq n$, based on a specific distance measure (in this work, the most generic one, the Euclidean distance) and a so-called linkage method until a stopping criterion is met. The result is a dendrogram, a tree-like diagram used to represent the hierarchical clustering of data, that can be visually examined and interpreted. As we consider only low-dimensional binary datasets in this study, AC is a suitable choice due to its intuitive results, ease of implementation, preservation of spatial proximity, flexibility in linkage criteria, and scalability for low-dimensional datasets.

[2] When performing clustering, the labels y_i are not utilized, only the objects $\mathbf{x}_i$ are used.

Table 1
Different linkages for merging clusters C_j and C_l: $\mathbf{c}_j$ and $\mathbf{c}_l$ denote the centroids of C_j and C_l, respectively; $d(\mathbf{u},\mathbf{v})$ ($d(\mathbf{c}_j,\mathbf{c}_l)$) denotes the (in our case, Euclidean) distance, i.e. the dissimilarity measure, between objects $\mathbf{u}$ and $\mathbf{v}$ ($\mathbf{c}_j$ and $\mathbf{c}_l$); $wcss(C_j)$ and $wcss(C_l)$ denote the within-cluster sum of squared distances between objects in clusters C_j and C_l, respectively.

Linkage	Distance between clusters C_j and C_l ($1 \le j,l \le n$)	Paper
S-Link (L_S)	$d_S(C_j,C_l) = \min\{d(\mathbf{u},\mathbf{v}) \mid \mathbf{u} \in C_j, \mathbf{v} \in C_l\}$	Sneath (1957)
A-Link (L_A)	$d_A(C_j,C_l) = \frac{1}{\lvert C_j\rvert\lvert C_l\rvert}\sum_{\mathbf{u}\in C_j}\sum_{\mathbf{v}\in C_l} d(\mathbf{u},\mathbf{v})$	Sokal (1958)
C-Link (L_C)	$d_C(C_j,C_l) = d(\mathbf{c}_j,\mathbf{c}_l)$	Sokal (1958)
M-Link (L_M)	$d_M(C_j,C_l) = \max\{d(\mathbf{u},\mathbf{v}) \mid \mathbf{u} \in C_j, \mathbf{v} \in C_l\}$	McQuitty (1960)
W-Link (L_W)	$d_W(C_j,C_l) = \sqrt{\frac{(\lvert C_j\rvert + \lvert C_l\rvert).wcss(C_j).wcss(C_l)}{(\lvert C_j\rvert + \lvert C_l\rvert)^2}} d(\mathbf{c}_j,\mathbf{c}_l)^2$	Ward (1963)

3.1.1. Merging clusters and linkage selection

During the AC process, we combine each pair of clusters C_j and C_l ($1 \le j,l \le n$) by using a *linkage* (Murtagh & Contreras, 2012), determining the distance between clusters of data points. For our work we considered five different linkages, such that single (S-Link), complete (M-Link), average (A-Link), centroid (C-Link) and ward (W-Link) linkage. Different linkages may produce different dendrograms and each linkage has its way of operating. For instance, in S-link, also known as the min linkage, the distance between two clusters would be the minimum distance between any two points in C_j and C_l. As a result, we would produce long, elongated clusters (chaining problem). However, if we consider M-Link, also known as the max linkage, then the distance between two clusters would be the maximum distance between any two points in the two clusters. That would produce compact, spherical clusters. Further, in A-Link, a compromise between S-Link and M-Link, we compute the distance between two clusters as the average distance between all pairs of points in the two clusters. In C-Link, the distance between two clusters is the distance between their centroids. This linkage is computationally efficient and can produce well-separated clusters. Last but not least, in W-Link, the distance between C_j and C_l is expressed as the increase in the sum of squared distances within the resulting merged cluster compared to the sum of squared distances within the separate clusters before merging. This linkage yields compact, spherical clusters but is sensitive to cluster size and shape. Table 1 summarizes how each linkage is computed.

Choosing the wrong linkage would result in poor clustering performance, forging none well-separated clusters, or clusters that do not correspond to the underlying structure of the data. Therefore, it is essential to consider the choice of linkage method carefully. On the other hand, the choice of the distance measure has not as great influence as the choice of the linkage, thus, a generic Euclidean distance is, usually, a good choice.

Several clustering validation metrics can be used to optimize the linkage selection in AC. One strategy is to employ the silhouette score (Rousseeuw, 1987), which estimates the clustering quality by quantifying the extent of separation between clusters, or the Davies-Bouldin index (Davies & Bouldin, 1979), which measures the likeness between clusters by assuming the ratio of the within-cluster scatter and the between-cluster distance. Another good technique is the cophenetic correlation coefficient (CPCC, Farris (1969)), which measures how well a dendrogram describes the pairwise distances between the data points.

Definition 3 *(Cophenetic correlation coefficient).* Let $\mathbf{x}_i$ and $\mathbf{x}_j$ ($1 \le i,j \le n$) be two objects in the dendrogram, resulting from AC. Let $d_E(\mathbf{x}_i,\mathbf{x}_j)$ denote the pairwise (Euclidean) distance between $\mathbf{x}_i$ and $\mathbf{x}_j$, and $d_C(\mathbf{x}_i,\mathbf{x}_j)$ denote the cophenetic (Euclidean) distance between $\mathbf{x}_i$ and $\mathbf{x}_j$, i.e. the distance between the largest two clusters that contain $\mathbf{x}_i$ and $\mathbf{x}_j$ individually before they are merged into a single cluster that contains both of these objects. The Cophenetic Correlation Coefficient is computed as:

$$CPCC = \frac{2}{n(n-1)}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left(d_E(\mathbf{x}_i,\mathbf{x}_j).d_C(\mathbf{x}_i,\mathbf{x}_j)\right) \tag{2}$$

CPCC can be considered better than the silhouette coefficient and Davies-Bouldin index mainly for two reasons. First, CPCC assumes the whole dendrogram, whereas the others only consider the pairwise distances between data points. In other words, CPCC can capture the overall structure of the dendrogram, including the branching pattern and the distances between the clusters. Second, unlike the silhouette coefficient and Davies-Bouldin index, CPCC is insensitive to the number of clusters chosen, making it useful when comparing clustering results with different numbers of clusters. For the mentioned advantages, we only considered CPCC as a clustering validation metric in this work to determine which linkage method, denoted as L_{best}, fits each dataset well.

In the context of linkage selection, CPCC evaluates the quality of a particular linkage method by comparing the pairwise distances between the data points with the distances between the corresponding nodes in the dendrogram. A high CPCC indicates that the dendrogram accurately represents the pairwise distances between the data points, and therefore the linkage method used to construct the dendrogram is a good choice for clustering the data. Thus, the linkage with the highest CPCC value is selected as the best method for agglomerative clustering. The algorithm for linkage selection based on CPCC can be summarized as the following:

- **Step 1:** Calculate the pairwise distances between the data points using the Euclidean distance.
- **Step 2:** Construct a dendrogram using a particular linkage method.
- **Step 3:** Compute CPCC score for the dendrogram using Eq. (2).
- **Step 4:** Repeat steps 2 and 3 for different linkage methods.
- **Step 5:** Select the linkage method that produces the dendrogram with the highest CPCC.

3.1.2. Cluster optimization

Cluster optimization is the process of determining the optimal number c of clusters in a clustering algorithm, a challenging task, as it depends on the dataset and the clustering algorithm used. Selecting inappropriate c can lead to inaccurate or irrelevant results. If there are too few clusters, the data may be oversimplified, and subgroups may be missed. On the other hand, if there are too many clusters, then the data may be over-complicated, and the results may not be interpretable or useful. Several methods have been developed for optimizing c, including the elbow method (Thorndike, 1953), silhouette analysis (Rousseeuw, 1987), the Davies-Bouldin index (Davies & Bouldin,

1979), and the Bayesian Information Criterion (B_{score}, Schwarz (1978)) computed as:

$$B_{score} = -2\ln\mathcal{L} + z\ln n \tag{3}$$

with:

$$z = (c-1) + (d\,.\,c) + 1$$

where $\mathcal{L}$ is the maximum likelihood of the data given the model, z is the number of parameters in the model, n is the sample size and d is the dimensionality of the data (see Eq. (1)).

B_{score} penalizes models with more parameters, which helps to avoid overfitting and ensures that the algorithm does not create excessive clusters, preventing the overshadowing effect of large clusters. For optimizing c in AC, we calculate B_{score} for different numbers of clusters and select c that minimizes B_{score}.

Schubert (2022) compared several methods for selecting c for the k-means clustering algorithm and reported that the Elbow method could be inconsistent (delivering sub-optimal clustering results), while the silhouette analysis and Davies-Bouldin index often produce more trustworthy and informative results than the elbow criterion, especially for well-separated and overlapping clusters. Their experimental results reveal that the B_{score} always performed well[3] despite using different heuristics on synthetic datasets, such as well-separation, overlapping, number of clusters, and uniform or normal data distribution.

3.2. Extraction and classification of representative points

A typical application of cluster-based sampling for imbalance learning is in disjunct subsets problems, where several disjoint sub-clusters characterize $\mathcal{P}$. In such cases, traditional oversampling methods may be ineffective, as they tend to oversample the target class uniformly across the feature space rather than focusing on the specific sub-clusters where the minority class is most prevalent. However, class boundaries, which play a critical role in most classifications (Han et al., 2005), are not often considered. Thus, instead of synthesizing data directly, we extract clusters' representative points in this work. Representative points, elements of the set $\mathcal{R}$ introduced before, are often more interpretable than the original data points, making it easier to gain insight into the underlying patterns and relationships in the data, explore cluster's sub-spaces, and consider the distribution of both classes, especially the instances nearby the borderlines. By contrast, the proposed approach is tailored to each sub-cluster individually, allowing for more effective oversampling of $\mathcal{P}$ in each region of the feature space. Which would probably improve the classification performance, mainly when dealing with complex and highly imbalanced datasets.

Thus, at this level, we would extract the representative points $\mathcal{R}_i$ for each cluster C_i, but before that, we first should specify the number of representative points $|\mathcal{R}_i|$ per cluster ($1 \le i \le c$), as defined in the Eq. (4)

$$|\mathcal{R}_i| = \begin{cases} 0, & \text{if } p = 0 \wedge |C_i| \ge k' \\ 1, & \text{if } (p = 0 \wedge |C_i| < k') \vee (p = 1) \\ \frac{|C_i|\,.\,|S|}{|S| + (|C_i|-1)}, & \text{otherwise} \end{cases} \tag{4}$$

where

$$|S| = \frac{Z_\alpha^2\,.\,p\,.\,(1-p)}{(\frac{\log|C_i|}{|C_i|} + \epsilon)^2} \tag{5}$$

and

$$p = \frac{\sum_{\mathbf{x}\in C_i} \phi(\mathbf{x} \in \mathcal{P})}{|C_i|} \tag{6}$$

S is inspired by the sample size formula in statistics (Cochran, 1977), in which ϵ is the acceptable tolerance error that can be adjusted as required, Z_α is the critical value of the Z test at the significance level α (Shi et al., 2022), ϕ is an indicator function,[4] and p is the variance of a proportion denoting the percentage of a sample having a particular characteristic,[5] and k' is a hyper-parameter related to the size of clusters. Here, p denotes the proportion of objects in the given cluster C_i belonging to the minority class $\mathcal{P}$.

In other words, $|\mathcal{R}_i|$ will increase as the within-class impurity grows i.e., $p \cdot (1-p)$ increases. Z_α is set to the value of 1.645, referring to $\alpha = 90\%$, and the margin of error ϵ is set to 0.05. The function $\frac{\log(|C_i|)}{|C_i|}$ scales down the cardinality of the cluster C_i into the interval $(0, \frac{1}{e})$, and is added to a fixed ϵ, the square of which is inversely proportional to S. This would help us to limit $|\mathcal{R}_i|$ for large clusters.

Moreover, $|\mathcal{R}_i| = 0$ solely if all the instances within a given cluster C_i belong to $\mathcal{N}$ while the size of the cluster is greater than k'. If k' is too small (e.g., $k' = 1$), it will be difficult to distinguish the features of different samples; if it is too large, it cannot capture the local property of the representative point. Thus, referring to Napierala and Stefanowski (2016), Shi et al. (2022), we use $k' = 5$ to capture the local property of each representative point. If $|C_i| \ge k'$, it would mean that the neighboring area of the representative point is populated only by instances from $\mathcal{N}$. As we would use the incremental kNN strategy to extract safe and half-safe areas (see Sect. 3.3) and only consider oversampling minority class regions, representative of size $|\mathcal{R}_i| = 0$ can neither be safe nor half-safe. Thus, we will not extract any representative point in such areas. In contrast, $|\mathcal{R}_i| = 1$ reflects that the cluster contains only instances from $\mathcal{P}$ or less than five instances from $\mathcal{N}$. Otherwise, $|\mathcal{R}_i|$ is computed by the third line in Eq. (4). The closer a cluster is to the borderline, the more representative points will be extracted. It is important to note that as we mainly deal with neighboring searches at current and the following stages, KDTree (K-Dimensional Tree, Bentley (1990)) is used to reduce the number of computational resources. The algorithm can quickly find a point's nearest neighbor and reduce the number of pairwise distances that need to be computed. By using KDTree, the time complexity of finding nearest neighbors can be reduced from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$, where n is the number of data points. Overall, KDTree offers fast nearest neighbor search and efficient range queries in low to moderate dimensional spaces.

Once we set the size $|\mathcal{R}_i|$, we start extracting the representative points $\mathcal{R}_i$ for each cluster C_i according to the well-scattered points extraction strategy described by Algorithm 1, adopted from CURE (Guha et al., 1998).

Algorithm 1 Extraction of representative points.

Input: Cluster C_i, shifting rate $\delta \in [0, 1]$
Output: A set of representative points $\mathcal{R}_i$

$\mathcal{R} \leftarrow \emptyset$
determine $|\mathcal{R}_i|$ using Eq. (4)
$\mathbf{c}_i \leftarrow \frac{1}{|C_i|}\sum_{\mathbf{x}\in C_i}\mathbf{x}$ ▷ Get the cluster's centroid
$\mathbf{r}_1 \leftarrow \arg\max_{\mathbf{x}\in C_i} d(\mathbf{x}, \mathbf{c}_i)$ ▷ Select the first representative point
$\mathcal{R} \leftarrow \mathcal{R} \cup \{\mathbf{r}_1\}$
for $j = 2, \ldots, |\mathcal{R}_i|$ **do**
  $\mathbf{r}_j \leftarrow \arg\max_{\mathbf{x}\in C_i \setminus \mathcal{R}} \min_{\mathbf{r}\in\mathcal{R}} d(\mathbf{x}, \mathbf{r})$ ▷ Find the farthest point from $\mathcal{R}$
  $\mathcal{R} \leftarrow \mathcal{R} \cup \{\mathbf{r}_j\}$
end for
$\mathcal{R}_i \leftarrow \{\mathbf{r} + \delta(\mathbf{c}_j - \mathbf{r}) \mid \mathbf{r} \in \mathcal{R}\}$ ▷ Shift points toward the centroid
return $\mathcal{R}_i$

[3] Justified by an other experiment reported in https://towardsdatascience.com/are-you-still-using-the-elbow-method-5d271b3063bd (last access: 26/07/2023).

[4] $\phi(\cdot) = 1$ if the expression $\cdot$ in its parameter is true, otherwise $\phi(\cdot) = 0$.

[5] E.g., Taherdoost (2017) used $p = 0.4$ to denote that 40% of the population are female.

For a cluster C_i with $|\mathcal{R}_i| = 1$, its centroid $\mathbf{c}_i$ will be used as a representative point, while clusters with $|\mathcal{R}_i| = 0$ are discarded (ignored in the subsequent steps). These clusters are populated only by instances from $\mathcal{N}$ and are not considered during the generation process as they are far away from the decision boundaries between $\mathcal{N}$ and $\mathcal{P}$. For each remaining cluster C_i, we take the farthest point $\mathbf{r}_1 \in C_i$ from its centroid $\mathbf{c}_i$ as the first representative point. Then, in each subsequent iteration $j \in \{2, \ldots, |\mathcal{R}_i|\}$, we pick up an instance $\mathbf{r}_j \in C_i$, which is farthest from the set of representative points $\mathcal{R}_i$, considering the single linkage (S-Link) distance between the two sets $\{\mathbf{r}_j\}$ and $\mathcal{R}$ (see Table 1). After that, we shift $\mathbf{r}_1, \ldots, \mathbf{r}_{|\mathcal{R}_i|} \in \mathcal{R}_i$ toward the centroid $\mathbf{c}_i \in C_i$ by δ. The hyper-parameter δ, represented by α in the original algorithm, as suggested by Guha et al. (1998) and Cai and Liang (2018), should ideally fall within the range of $[0.2, 0.7]$. This range helps mitigate the impact of noise and outliers during cluster formation. A higher δ value leads to representatives being positioned closer to the centroid, while in our research, δ denotes the shifting rate of representative points. These representatives act as centroids around which synthetic samples are generated in subsequent steps, affecting the spreads between synthetic data points and each cluster's centroid. A larger δ value would result in synthesizing samples closer to the centroid, while a smaller δ value would allow for more dispersion within the cluster. Considering the diverse characteristics of each dataset, we retain the need to optimize δ to achieve optimal performance.

Once all representative points $\mathbf{r}_j \in \mathcal{R}_i$ $(1 \le j \le |\mathcal{R}_i|)$ are extracted, we classify them into *safe*, *half-safe* or *unsafe* classes using Eq. (8). For that, the k-nearest neighbors $\mathbf{n}_1^j, \ldots, \mathbf{n}_k^j$ of each representative point $\mathbf{r}_j$ are identified, and the proportion of the minority samples among these nearest neighbors is adopted as the weight $w^k(\mathbf{r}_j) \in [0, 1]$ of $\mathbf{r}_j$, computed as:

$$w^k(\mathbf{r}_j) = \frac{1}{k} \sum_{l=1}^{k} \phi(\mathbf{n}_l^j \in \mathcal{P}) \tag{7}$$

where ϕ is an indicator function, defined before, and k is a hyper-parameter. Based on $w^k(\mathbf{r}_j)$, $\mathbf{r}_j$ is classified as safe, half-safe or unsafe as follows.

$$\mathbf{r}_j = \begin{cases} \text{safe}, & \text{if } w^k(\mathbf{r}_j) = 1 \\ \text{half-safe}, & \text{if } 0.5 < w^k(\mathbf{r}_j) < 1 \\ \text{unsafe}, & \text{otherwise} \end{cases} \tag{8}$$

This indicates that a representative point $\mathbf{r}_j \in \mathcal{R}_i$ is classified as safe only if all its k-nearest neighbors belong to $\mathcal{P}$. In contrast, $\mathbf{r}_j$ is half-safe if most of its k-nearest neighbors belong to $\mathcal{P}$. If $\mathbf{r}_j$ does not satisfy one of the mentioned conditions, we categorize it as unsafe as most of its k-nearest neighbors belong to $\mathcal{N}$ and not $\mathcal{P}$.

3.3. Areas estimation using incremental kNN

In order to extract minority class sub-spaces, we investigate instances surrounding a representative point. The objective is to expand representative point areas using incremental kNN as long as some prerequisites are satisfied. The details are presented in Algorithm 2. The resulting areas are used as input for a given data generator to synthesize artificial minority class samples with new characteristics unavailable in the original data.

Considering the classification condition of the point (object) being half-safe, the parameter of kNN should be an odd number and greater than one. We choose $k = 3$ to ensure that the disjunct subset with at least three minority instances can be identified as a safe area, as larger k values may result in bypassing some tiny but safe areas (Fig. 5). Furthermore, the resulting representative points from the previous step are initially classified by 3-NN.

Algorithm 2 treats each $\mathbf{r}_j \in \mathcal{R}_i$ $(1 \le j \le |\mathcal{R}_i|)$ differently, relying on Eq. (8). As described in Fig. 6(a), the algorithm keeps expanding the safe area incrementally, by $k + 1$. The $k + 1^{th}$ neighbor of $\mathbf{r}^j$ is accepted only when it is a minority instance; if not, the kth nearest neighbor $\mathbf{n}_k^j$ of $\mathbf{r}_j$ would be the farthest point that keeps $\mathbf{r}_j$'s area safe. For a half-safe point, the algorithm keeps expanding its area as long as the area remains half-safe (Fig. 6(b)).

For unsafe points, two possibilities may occur (see Fig. 4). An unsafe point may remain unsafe or revert to half-safe. Thus, to avoid losing

Algorithm 2 Incremental kNN.

Input: The set of representative points $\mathcal{R}_i$ of C_i
Output: Safe $\mathcal{R}_s^i$ and half-safe $\mathcal{R}_h^i$ areas of $\mathcal{R}_i$

$\mathcal{R}_s^i, \mathcal{R}_h^i \leftarrow \emptyset$
for $j = 1, 2, \ldots, |\mathcal{R}_i|$ **do**
  Calculate $w^k(\mathbf{r}_j)$ by Eq. (7) ▷ $\mathbf{r}_j \in \mathcal{R}_i$, $k = 3$
  if $w^k(\mathbf{r}_j) = 1$ **then** ▷ Safe
    $k = \min_{l \ge 3} w^{l+1}(\mathbf{r}_j) \ne 1$
    $\mathcal{R}_s^i \leftarrow \mathcal{R}_s^i \cup \langle \mathbf{r}_j, \mathbf{n}_k^j \rangle$ ▷ Store the safe area (from $\mathbf{r}_j$ to its k-th nearest neighbor $\mathbf{n}_k^j$)
  else if $w^k(\mathbf{r}_j) \le 0.5$ **then** ▷ Unsafe
    $k' = \min_{l \in [3,10]} w^l(\mathbf{r}_j) > 0.5$
    $k = \min_{k' \le 10, l \ge k'} \left(w^{l+1}(\mathbf{r}_j) \le 0.5 \wedge \phi(\mathbf{n}_{l+1}^j \in \mathcal{P}) = 0 \right)$
    $\mathcal{R}_h^i \leftarrow \mathcal{R}_h^i \cup \langle \mathbf{r}_j, \mathbf{n}_k^j \rangle$ ▷ Store the area that become half-safe after expansion
  else ▷ Half-safe
    $k = \min_{l \ge 3} \left(w^{l+1}(\mathbf{r}_j) \le 0.5 \wedge \phi(\mathbf{n}_{l+1}^j \in \mathcal{P}) = 0 \right)$
    $\mathcal{R}_h^i \leftarrow \mathcal{R}_h^i \cup \langle \mathbf{r}_j, \mathbf{n}_k^j \rangle$ ▷ Store the half-safe area
  end if
end for
return $\mathcal{R}_s^i$, $\mathcal{R}_h^i$

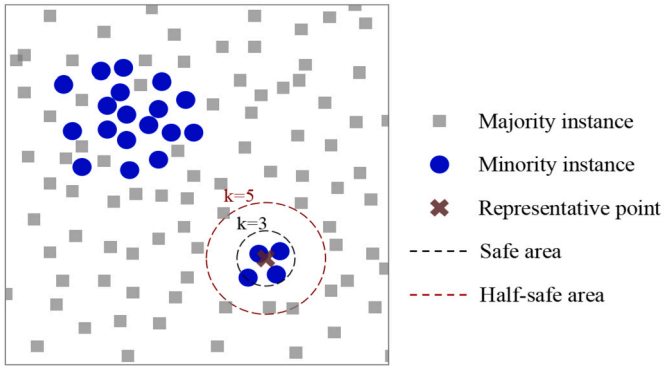

Fig. 5. A large k may result in omitting some small safe minority disjunct subsets.

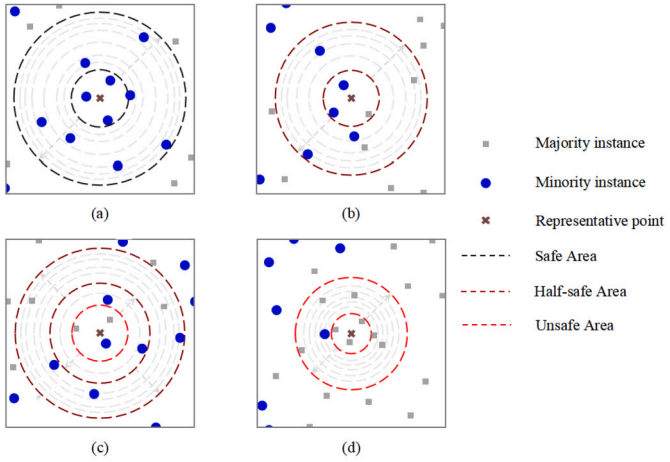

Fig. 6. Examples of areas extraction using the incremental kNN strategy: (a) a safe area populated from a safe representative point; (b) a half-safe area populated from a half-safe representative point; (c) a half-safe area populated from an unsafe representative point; (d) a discarded unsafe area populated from an unsafe point where *max-iter* is reached.

the local property of a given unsafe point, we set a *max-iter* parameter for the incremental kNN. If max-iteration is reached and the point stays unsafe, that area will be discarded (Fig. 6(d)) as it indicates an overpopulated area by majority-class instances. However, if an unsafe point changes to half-safe before reaching *max-iter*, we extend its area as shown in Fig. 6(c) according to the strategy utilized for half-safe points.

The value for *max-iter* for identified unsafe areas in our incremental kNN algorithm was empirically set to 10 so that it would provide a good, overall default setting for the datasets used in our experiments. This choice aims to balance meaningful area expansion and preventing excessive growth. The goal is to ensure effective expansion without compromising computational efficiency, addressing the need for meaningful exploration while avoiding unnecessary computational burden. Opting for a higher value of *max-iter* could result in unnecessary computational costs, whereas a lower value might limit the algorithm's capacity to identify potential minority class half-safe areas. By restricting expansion, we also mitigate potential overlap issues with other areas, preserving the algorithm's ability to distinguish distinct minority class sub-spaces. This constraint on iteration optimizes resource utilization, allowing for the efficient capture of relevant sub-spaces. Additionally, the finite nature of *max-iter* aids in identifying overpopulated majority-class areas, preventing the consideration of regions dominated by the majority class. However, we do realize that this hyper-parameter might need fine-tuning in case of new datasets with different characteristics than those used in our experiments.

3.4. Weighting safe and half-safe areas

Given the expected imbalance ratio, denoted as IR_e, to specify the desired balance level after the synthetic data generation process, and the weights w_s and w_h of safe and half safe areas, respectively, we determine the number of synthetic data examples γ that we would generate using Algorithm 3 for $\mathcal{P}$ by Eq. (9), and which then we split between safe areas and half-safe areas according to Eqs. (10) and (11), respectively.

$$\gamma = \frac{|\mathcal{N}| - |\mathcal{P}|}{IR_e} \tag{9}$$

$$|D_s| = \gamma \cdot \frac{w_s \,.\, |\mathcal{R}_s|}{w_s \,.\, |\mathcal{R}_s| + w_h \,.\, |\mathcal{R}_h|} \tag{10}$$

$$|D_h| = \gamma - |D_s| \tag{11}$$

$$\eta_a = \begin{cases} \frac{\mu_a \,.\, |D_s|}{\mu_s} & \text{if } a \in \mathcal{R}_s \\ \frac{\mu_a \,.\, |D_h|}{\mu_h} & \text{if } a \in \mathcal{R}_h \end{cases} \tag{12}$$

In practice, we set the value of w_s and w_h equal to 1 by default, which could be customized if needed. Then, the exact number $\eta_a \in \mathbb{N}$ of points to generate for each area a is determined based on μ_a, the number of minority instances that belongs to the area a. In Eqs. (10) and (11), μ_s and μ_h are the sum of the number of minority instances that belongs to all $a \in \mathcal{R}_s$ and $a \in \mathcal{R}_h$, respectively. As the number of synthetic samples $|D_s|$ that would be generated in safe areas, and the number of synthetic samples $|D_h|$ in half-safe areas, as well as in a given area η_a are rounded, the remaining synthetic samples that were not counted are added systematically into random safe areas to satisfy γ.

3.5. Synthetic data generation using Gaussian generator

When instances from the minority class are located in small, isolated areas of the feature space, standard methods such as SMOTE find it hard to generate synthetic samples useful for the classification task.

First, SMOTE can create dense areas in the feature space as it generates samples by interpolating between existing minority class samples. This can create dense areas of synthetic samples that are not representative of the true underlying distribution (Fig. 7(a)). These dense areas can have several negative consequences: they may lead to overfitting the machine learning model, as the model may learn to rely too heavily on the dense clusters of synthetic samples rather than the true underlying distribution. Thus, using a Gaussian generator, instead SMOTE, can be seen as a more nuanced and context-sensitive approach. The adopted Gaussian generator recognizes the non-normal distribution of imbalanced datasets. Rather than assuming a global normal distribution, it targets specific sub-spaces within the minority class for oversampling, allowing for more fine-grained control over the generation of synthetic samples (Fig. 7(b)) and ensuring a more accurate representation of the minority class without making assumptions about the overall dataset's normality. In contrast, SMOTE is a more general approach that does not account for the specifics of the distribution of $\mathcal{P}$, which can lead to the generation of synthetic samples that are not representative of the true underlying distribution of $\mathcal{P}$. Overall, a Gaussian generator can improve performance and a more accurate representation of $\mathcal{P}$ in small disjunct areas. Its implementation is described in Algorithm 3.

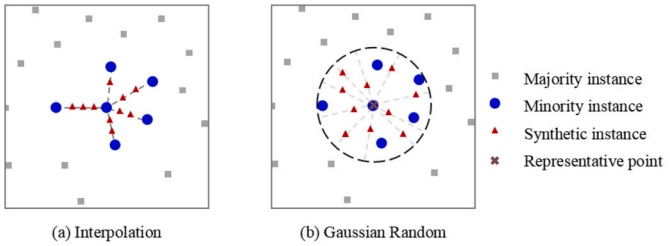

Fig. 7. Gaussian random generator (right) compare to the interpolation method (left).

Algorithm 3 Sampling from Truncated Hyper-spherical Gaussian Distribution.

Input: Representative area $\langle \mathbf{r}_j, \mathbf{n}_k^j \rangle$, standard deviation σ
Output: Synthetic sample $\mathbf{x}$

1: $r = d_E(\mathbf{r}_j, \mathbf{n}_k^j)$ ▷ (Euclidean) Radius of the hyper-sphere centered at $\mathbf{r}_j$
2: **do** ▷ check-point
3: $\mathbf{z} \sim N(0, \sigma . I_d)$ ▷ I_d is a d-dimensional identity matrix
4: $\mathbf{x} = \mathbf{r}_j + r \,.\, \mathbf{z}$ ▷ generate a sample
5: **while** $d_E(\mathbf{r}_j, \mathbf{x}) > r$
6: **if** $\langle \mathbf{r}_j, \mathbf{n}_k^j \rangle \in \mathcal{R}_h$ **then** ▷ The given area is half-safe
7: **if** $w^k(\mathbf{x}) \neq 1$ **then** ▷ Eq. (7)
8: Go back to step 2
9: **end if**
10: **end if**
11: **return x**

Initially, Algorithm 3 computes the radius r, which represents the Euclidean distance between $\mathbf{r}_j$, the center of the hyper-spherical region and $\mathbf{n}_k^j$, the k-th nearest neighbor of $\mathbf{r}_j$ estimated by Algorithm 2. This process involves iteratively sampling a point $\mathbf{z}$ from a Gaussian distribution with a mean of 0 and a standard deviation of σ, denoted as $N(0, \sigma \cdot I_d)$. Subsequently, a candidate sample $\mathbf{x}$ is generated by scaling $\mathbf{z}$ with the radius r and translating it to the center $\mathbf{r}_j$. Following this, the algorithm verifies whether the distance between $\mathbf{r}_j$ and $\mathbf{x}$ exceeds r, iterating until a valid sample within the hyper-sphere is obtained. In case the representative area $\langle \mathbf{r}_j, \mathbf{n}_k^j \rangle \in \mathcal{R}_h$, an additional check is conducted. Specifically, the algorithm examines the local property of $\mathbf{x}$ by evaluating whether its weight, denoted as $w^k(\mathbf{x})$, is not equal to 1. If this condition is not satisfied, the algorithm returns to the sampling step. Upon fulfilling all conditions, the algorithm outputs the synthetic sample $\mathbf{x}$.

As mentioned earlier, the algorithm involves iteratively sampling a point $\mathbf{z}$ from a Gaussian distribution with a mean of 0 and a standard deviation of σ, where $\sigma \in [0, 1]$. The value of σ reflects the degree of convergence of the generated samples from the center $\mathbf{r}_j$ of the given area.

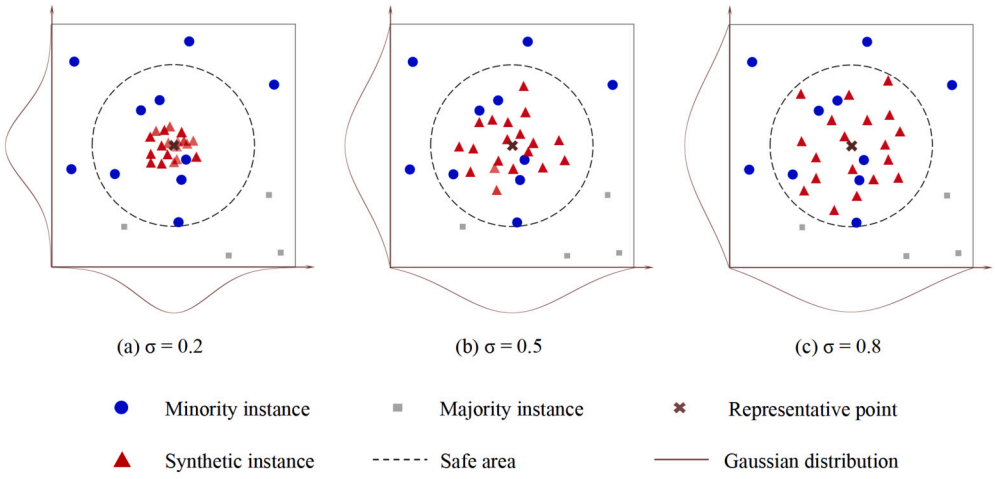

Fig. 8. Effect of σ on synthetic instances.

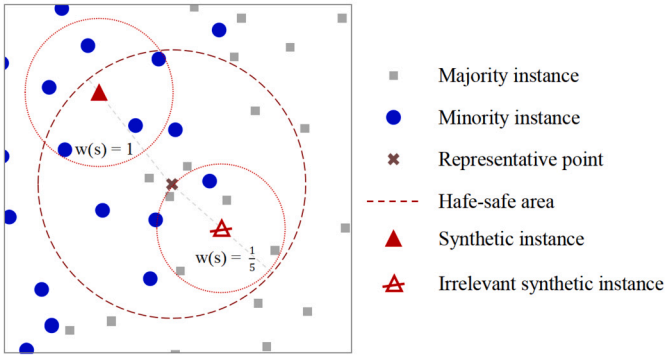

Fig. 9. Illustration of the filter in a half safe area.

As described in Fig. 8(a) and Fig. 8(b), picking a small σ would yield a dense area of generated instance near $\mathbf{r}_j$ and do not fully cover the extracted area's sub-spaces, whereas a high σ (Fig. 8(c)) would cover more sub-space but might generate instances out of the area's borders which might cause a problem if two extracted areas or more are overlapping. Therefore, we check whether a generated instance is within a hyper-sphere (line 5 of Algorithm 3), and fixed σ to 0.8. To justify our choice of $\sigma = 0.8$ in our study, we note that the Gaussian sampling results were highly comparable for values of σ greater than 0.8. Thus, by selecting $\sigma = 0.8$, we can ensure a satisfactory level of diversity in the generated data. Moreover, we found that $\sigma = 0.8$ accelerates our algorithm's convergence as it reduces the number of rejected instances due to being outside the designated radius r (line 1 of Algorithm 3). Therefore, we determined that $\sigma = 0.8$ was suitable for our study based on these considerations.

Given that our Algorithm 3 populates both safe and half-safe areas, if we apply it directly in the half-safe area, we would probably induce instances encircled by majority class instances. Such instances influence classes' separability and further worsen the precision of majority class (Kulkarni et al., 2020). Thus, we involve a filter (line 7 of Algorithm 3) based on the nearest neighbors for each synthetic sample when oversampling in half-safe areas. The filter can be seen as a post-step restraining the random vector from generating artificial instances only in the minority class's direction. We compute the weight $w^k(\mathbf{x})$ of each sample $\mathbf{x}$ using Eq. (7). The number of nearest neighbors k is adaptative to the half-safe area. When $\mu_a < 5$ for a given area a, k would be equal to μ_a. Otherwise, the filter fixes $k = 5$. As illustrated in Fig. 9, $\mathbf{x}$ is preserved if $w^k(\mathbf{x}) = 1$ meaning that all its neighbors belong to $\mathcal{P}$.

4. Experiments

Using the taxonomy proposed in Fig. 3, we classify AROSS as a cluster-based adaptive oversampling method due to its agglomerative clustering module to extract data features and its adaptive generation of synthetic samples in safe and half-safe regions using different weights. The parameters L_{best} (the best linkage), c (the number of clusters), and $|\mathcal{R}_i|$ (the number of representative points per cluster) are automatically fine-tuned for each dataset based on the CPCC, B_{score}, and the sample size formula, respectively, while δ (the shifting rate, see Algorithm 1) is a customizable hyper-parameter.

To assess the performance of the proposed approach, we evaluate it with two variants. The first one, without the use of shifting, denoted as AROS (area-based representative points oversampling), uses $\delta = 0$. The second one, with shifting, designated as AROSS (area-based representative points oversampling with shifting), optimizes δ based on the recall measure. Luque et al. (2019) highlighted that recall is the sole bias-free metric in imbalance learning that specifically emphasizes the prediction of minority samples, which are frequently the target class. While optimizing, various δ values from the interval $[0, 1]$, using a step length of 0.1 are considered. Selecting the optimal δ that maximizes recall ensures that representative points are shifted within clusters to populate ideal areas, capturing most of the minority data distribution and generating effective synthetic samples.

To evaluate the performance of the proposed method, we present the experimental comparison with state-of-the-art data-level resampling methods highlighted in Table 2, and their ranges of hyper-parameters in Table 3. Considering the scale of our experimental design, providing a consolidated overview through the specification of hyper-parameter ranges emerges as a pragmatic and informative strategy for presenting the breadth of our experimentation.

4.1. Datasets

To assess the performance of AROS and AROSS, we utilized 70 imbalanced datasets for benchmarking. Among these, 66 were real datasets sourced from the UCI (Asuncion & Newman, 2007) and KEEL (Derrac et al., 2015) repositories. Additionally, we created four artificial datasets with sample prefix. As we focus on binary classification problems, datasets with more than two classes were converted into two-class datasets. Detailed information about the 70 datasets can be found in Table 5. We categorized the datasets into three groups to analyze the imbalance severity based on their IR values. Specifically, 22 datasets were slightly imbalanced ($IR \leq 5$), 25 were moderately imbalanced ($5 < IR \leq 10$) while the remaining datasets were severely imbalanced ($IR > 10$).

Table 2
Data-level CIL approaches involved in the experiments.

Category	Sub-category	CIL approach	Acronym
Random sampling	Oversampling	Random oversampling	ROS (Chawla, 2010)
Synthetic sampling	Data generation	Synthetic minority oversampling	S (Chawla et al., 2002)
		Noise reduction a priori synthetic oversampling	NRAS (Rivera, 2017)
		Random walk oversampling	RWO (Zhang & Li, 2014)
		Partially guided SMOTE oversampling	GS (Sandhan & Choi, 2014)
	Adaptive	Adaptive synthetic sampling	AS (He et al., 2008)
		Deterministic SMOTE	DS (Torres et al., 2016)
		Adaptive neighbor synthetic minority oversampling	ANS (Siriseriwan & Sinapiromsaran, 2017)
		Synthetic minority based on probabilistic distribution	SMPD (Kunakorntum et al., 2020)
	Data cleaning	SMOTE with Edited nearest neighbors	SENN (Batista et al., 2004)
		SMOTE with Tomek-link	STL (Batista et al., 2004)
Cluster-based	Oversampling	DB-SMOTE	DBS (Bunkhumpornpat et al., 2012)
		Cure-SMOTE	CS (Ma & Fan, 2017)
	Adaptive	Kmeans-SMOTE	KS (Douzas et al., 2018)
		Self-organizing map oversampling	SOMO (Douzas & Bacao, 2017)
		Area-based representative points oversampling	AROS (Farou et al., 2024)
		Area-based representative points oversampling with shifting	AROSS (Farou et al., 2024)

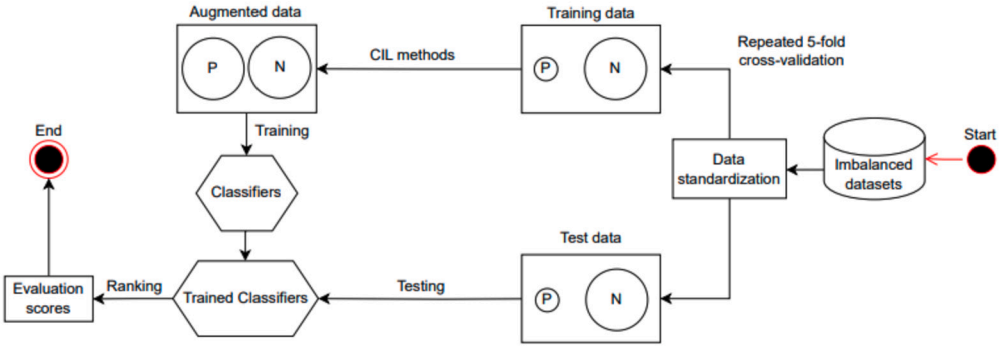

Fig. 10. The benchmark process.

4.2. Evaluation measures

To evaluate the performance of the different models, particularly their ability to effectively classify samples from $\mathcal{P}$, a set of commonly used metrics computed from the confusion matrix (Table 4) for classification tasks is employed, including the area under the curve (*AUC*), F_1 score (F_1), geometric mean (*GM*), and recall (*Rec*). These metrics comprehensively assess the models' performance and ability to distinguish $\mathcal{P}$ from $\mathcal{N}$.

- *AUC* is a metric that estimates the classifier's ability to distinguish between classes by calculating the area under the Receiver Operating Characteristic (ROC) curve. A high *AUC* signifies a stronger ability to accurately classify samples from different classes.
- *Rec* (also called sensitivity) quantifies the proportion of correctly classified $\mathcal{P}$ out of all the actual $\mathcal{P}$. A high *Rec* signifies a higher rate of correctly recognizing $\mathcal{P}$.
- F_1 is the harmonic mean of Precision (*Pre*) and *Rec*. It can be expressed as $F_1 = 2(Rec \times Pre)/(Rec + Pre)$, where $Rec = TP/(TP + FN)$ and $Pre = TP/(TP + FP)$. A high F_1 indicate better model performance.
- *GM* considers the balance of a classifier's performance for both $\mathcal{P}$ and $\mathcal{N}$. It is calculated as the square root of the product of sensitivity (*Rec*) and specificity (*Spec*), where $Spec = TN/(TN + FP)$. Similarly to F_1, a higher *GM* value indicates more balanced and accurate performance.

GM and *AUC* are identified as the most appropriate evaluation metrics for CIL problems by Luque et al. (2019), suggesting that these metrics have a null bias and consider the performance of both $\mathcal{P}$ and $\mathcal{N}$, providing a comprehensive two-dimensional assessment. Additionally, *Rec* is recognized as the optimal bias-free metric for CIL (Luque et al., 2019). It is a one-dimensional metric that focuses explicitly on predicting minority samples. While there is a significant bias associated with F_1-score when used for CIL, it is essential to note that F_1 have a high frequency of usage in CIL problems and statistical analysis and is considered a conventional metric for evaluating the performance of binary classification models (Jiang, Lu, et al., 2023, Wang et al., 2023).

4.3. The experimental process

Fig. 10 illustrates the whole process of the benchmark analysis the details of which are listed below.

Step 1: partitioning the train and test sets. To preserve class distributions and considering that some datasets have less than ten minority instances, we use the repeated stratified five fold cross-validation approach (Prusty et al., 2022). This method is highly dependable and resilient when dealing with limited or imbalanced data, as it helps mitigate bias, improve generalization, and provides valuable insights into the model's performance and variability. We perform stratified five fold cross-validation for each dataset and repeat this procedure ten times, resulting in 50 distinct split configurations.

Step 2: addressing the CIL problem. The baseline methods described in Table 2 were implemented from `imblearn` (Lemaître et al., 2017) and `smote-variants` (Kovács, 2019a) Python libraries. In addition to the baselines, the experimental results on the original datasets were also shown. These baselines are applied to the training sets generated in the previous step. In this process, we employ the default balancing strategy for each dataset, ensuring that the expected imbalance ratio IR_e

Table 3
Ranges of all hyper-parameters of the reported Data-level CIL methods.

	Random	Synthetic sampling									Cluster based					
hyper-parameters	ROS	S	AS	DS	SMPD	SENN	STL	NARS	GS	RWO	ANS	DBS	CS	KS	SOMO	AROS	AROSS
IR_e	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
k_neighbors	-	{3,5,7,9}	{3,5,7,9}	{3,5,7,9}	{3,5,7,9}	{3,5,7,9}	{3,5,7,9}	{3,5,7,9}	{3,5,7,9}	-	-	-	-	{3,5,7,9}	-	-	-
n_clusters	-	-	-	-	-	-	-	-	-	-	-	-	{5,10,15}	{2,5,10,20,50}	-	-	-
m_neighbors	-	-	-	-	{3,5,7,9}	-	-	-	-	-	-	-	-	-	-	-	-
learning rate	-	-	-	-	-	-	-	-	-	-	-	-	-	-	{0.3,0.5}	-	-
n_grid	-	-	-	-	-	-	-	-	-	-	-	-	-	-	{5,9,13}	-	-
min_samples	-	-	-	-	-	-	-	-	-	-	-	{1,3,5,7,9}	-	-	-	-	-
eps	-	-	-	-	-	-	-	-	-	-	-	{0.5,0.8,1.2}	-	-	-	-	-
threshold (t)	-	-	-	-	-	-	-	{0.3,0.5,0.8}	-	-	-	{0.5,0.8,1.2}	-	-	-	-	-
shifting rate (δ)	-	-	-	-	-	-	-	{0.3,0.5,0.8}	-	-	-	{0.5,0.8,1.2}	-	-	-	0	[0,0.1,⋯,1]

Table 4
Confusion matrix for binary problems, TP denotes the count of accurately predicted (true) positives, TN denotes the count of accurately predicted (true) negatives, FP denotes the count of falsely predicted positives, and FN denotes the count of falsely predicted negatives.

		Actual	
		Positive	Negative
Pred	Positive	TP	FP
	Negative	FN	TN

is set to 1. The augmented training sets $\mathcal{D}'$ are obtained as the outcome of this step. Mandating an IR_e of 1 enhances the efficacy of classification models as it ensures equitable representation, particularly favoring the identification of minority classes. According to Fotouhi et al. (2019), classification algorithms are affected by high *IR*s, especially when classes are non-linearly separated, underlining the importance of mitigating imbalances for optimal model performance. Therefore, solving the dominance of majority classes by resampling techniques would enable a potential model generalization and facilitate the impartial evaluation of diverse data-level methods. Furthermore, Shi et al. (2022) pointed out that resampling approaches aim to decrease the *IR* to a balanced or nearly balanced state, which could reduce or eliminate the bias toward the majority class. Also, Kovács (2019b) highlighted in his empirical comparison and evaluation of minority oversampling techniques that models trained on datasets with IR_e of 1 provide a standardized benchmark for comparing the effectiveness of various resampling techniques, enabling well-informed decisions regarding the most suitable methods. As a result, using an IR_e of 1 would help address dataset imbalances, advocate for fairness, and yield robust and dependable results.

Step 3: training classification models. we trained four classifiers—k-nearest neighbor (kNN),[6] decision tree (DT),[7] support vector classifier (SVC),[8] and random forest (RF)[9]—using the rebalanced training sets denoted as $\mathcal{D}'$. The implementation of these classification algorithms utilized the `scikit-learn` library (Pedregosa et al., 2011) Python API with their default hyper-parameter settings, displayed in Table 6.

The decision to maintain default parameters was strategic, as our primary focus was investigating the impact of data-level solutions on classification results. By adhering to default parameters, we ensured that any observed variations in performance could be unequivocally attributed to the resampling techniques, enabling a concentrated analysis of their effectiveness. Additionally, adopting default parameters enhances our experimental setup's simplicity and reproducibility and aligns with considerations of computational efficiency. Moreover, Mantovani et al. (2015) concluded that, for highly imbalanced data sets, tuning does not obtain much higher performance. Also, experiments conducted by Horváth et al. (2023) showed that using default values worked considerably well for classification algorithms. Therefore, given the study's scope, retaining default parameters allows us to efficiently explore the influence of data-level solutions without introducing unnecessary computational complexity. However, we do realize that hyper-parameter tuning, despite its high computational cost, might lead to better results.

[6] kNN implementation. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[7] DT implementation. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[8] SVC implementation. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[9] RF implementation. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Table 5
Description of datasets used for benchmark experiment.

#	Dataset	Att	Samp	$\mathcal{P}_{\%}$	IR	#	Dataset	Att	Samp	$\mathcal{P}_{\%}$	IR	#	Dataset	Att	Samp	$\mathcal{P}_{\%}$	IR
Skewness		IR ≤ 5						5 < IR ≤ 10						IR > 10			
D1	breast	30	569	37.26	1.68	D23	new-thyroid1	5	215	16.28	5.14	D48	glass016vs2	9	192	8.85	10.29
D2	wpbc	31	569	37.26	1.68	D24	ecoli2	7	336	14.48	5.46	D49	ecoli0147vs2356	7	336	8.63	10.59
D3	pimaindians-diabetes	8	768	34.89	1.87	D25	abalone510	8	749	15.35	5.51	D50	led7digit-02456789vs1	7	443	8.35	10.97
D4	seeds	7	210	33.33	2	D26	winequality-red3456vs78	11	1599	13.57	6.37	D51	ecoli01vs5	6	240	8.33	11
D5	wheat1	7	210	33.33	2	D27	glass6	9	214	13.55	6.38	D52	glass06vs5	9	108	8.33	11
D6	glass0	9	214	32.71	2.06	D28	yeast3	8	1484	10.98	8.1	D53	glass0146vs2	9	205	8.29	11.06
D7	glass1	9	214	32.71	2.06	D29	ecoli3	7	336	10.42	8.6	D54	glass2	9	214	7.94	11.59
D8	vertebral	6	310	32.26	2.1	D30	page-blocks1vs2345	10	5473	10.23	8.77	D55	ecoli0147vs56	6	332	7.53	12.28
D9	eligibility-loan	11	614	31.27	2.2	D31	ecoli034vs5	7	200	10	9	D56	cleveland0vs4	13	173	7.51	12.31
D10	sampledata_3	2	1100	30.0	2.33	D32	yeast2vs4	8	514	9.92	9.08	D57	ecoli0146vs5	6	280	7.14	13
D11	yeast1	8	1484	28.91	2.46	D33	ecoli067vs35	7	222	9.91	9.09	D58	yeast1vs7	7	459	6.54	14.3
D12	maternal-risk-lmvsh	6	1014	26.82	2.73	D34	ecoli0234vs5	7	202	9.90	9.1	D59	glass4	9	214	6.07	15.46
D13	haberman	3	306	26.47	2.78	D35	glass015vs2	9	172	9.88	9.12	D60	ecoli4	7	336	5.95	15.8
D14	parkinsons	22	195	24.62	3.06	D36	yeast0359vs78	8	506	9.88	9.12	D61	page-blocks13vs4	10	472	5.93	15.86
D15	glass0123vs456	9	214	23.83	3.2	D37	yeast0256vs3789	8	1004	9.86	9.14	D62	abalone918	8	731	5.75	16.4
D16	ecoli1	7	336	22.92	3.36	D38	ecoli046vs5	6	203	9.85	9.15	D63	glass016vs5	9	184	4.89	19.44
D17	page-blocks2vs4	10	417	21.1	3.74	D39	ecoli01vs235	7	244	9.84	9.17	D64	glass5	9	214	4.21	22.78
D18	leaf	15	340	20.59	3.86	D40	ecoli0267vs35	7	224	9.82	9.18	D65	yeast2vs8	8	482	4.15	23.1
D19	sampledata_2	2	1000	20.0	4	D41	glass04vs5	9	92	9.78	9.22	D66	yeast4	8	1484	3.44	28.1
D20	page-blocks3vs5	10	143	19.58	4.11	D42	ecoli0346vs5	7	205	9.76	9.25	D67	yeast1289vs7	8	947	3.17	30.57
D21	sampledata	2	600	16.67	5	D43	ecoli0347vs56	7	257	9.73	9.28	D68	yeast5	8	1484	2.96	32.73
D22	sampledata_1	2	900	16.67	5	D44	yeast05679vs4	8	528	9.66	9.35	D69	ecoli0137vs26	7	281	2.49	39.14
						D45	abalone48	8	625	9.12	9.96	D70	wilt	5	4339	1.71	57.64
						D46	vowel0	13	988	9.11	9.98						
						D47	ecoli067vs5	6	220	9.09	10						

Table 6
Key hyperparameters of reported classification algorithms. Here d refers to the number of features in the dataset, and Var(X) is the variance of the features.

Classifier	Hyperparameters
kNN	Number of neighbors (k): 5, Weighting strategy: 'uniform'
SVC	Regularization (C): 1.0, Kernel: 'rbf', Gamma: $\frac{1}{d \times Var(X)}$ Probability estimates: Enabled
DT	Splitting criterion: 'gini'
RF	Number of trees: 100, Splitting criterion: 'gini'

Step 4: evaluation and ranking. The performance measures that were previously discussed are utilized to evaluate the models. Scores are computed based on these measures and subsequently ranked in ascending order to determine the results.

5. Empirical results

The empirical findings include how CIL algorithms performed across the 70 datasets using the 4 performance measures. In the following subsections, we begin by summarizing the experimental results and comparing the results achieved by our proposed approach to the other approaches. Subsequently, we delve deeper into the observed outcomes and investigate their underlying reasons.

5.1. Benchmark analysis

Tables 7 and 8 present the average metric values and the average ranks, respectively, for *Rec*, F_1, *GM*, and *AUC* achieved by various classifiers using CIL approaches on a total of 70 datasets, where the best CIL approach is indicated by boldface. Supplementary findings for each dataset are accessible for further exploration.[10] The higher average metric values and smaller the ranking or score of a particular CIL method under a specific metric, the better its effect, which means that the best approach receives a ranking of 1, and the worst one gets an 18 while average metric values are between $[0, 1]$. In Table 8, the penultimate row, AvgR, referring to the mean average of ranks, highlights the performance of different CIL approaches (classifier combinations). The last row, Final score, is the ranking score of AvgR for each CIL method. It can be observed from Table 7 that AROSS is superior to the compared cluster-based approaches in terms of average *Rec*, F_1, and *GM* scores across all classification algorithms, and that AROS (AROSS with $\delta = 0$) is superior to or competitive with the remaining cluster-based techniques. Furthermore, the average rank of AROSS, as indicated in Table 8, is significantly lower than the average rank scores of the compared methods. AROSS achieves particularly low ranks in terms of *Rec* (2.04), *GM* (2.93), and *AUC* (2.57) with DT, as well as in terms of F_1 score with kNN (4.91). The rankings in AvgR results (Table 8) further highlight that AROSS consistently ranks at the top, followed by AROS. These results emphasize the importance of tuning the shifting rate δ in AROSS, as each dataset has unique characteristics and class distributions. By optimizing the resampling process and generating more representative synthetic instances, AROSS effectively captures the aspects of $\mathcal{P}$ within each cluster. Tuning δ based on recall is a reasonable approach when the primary goal is to improve the performance of minority class classification. However, improving the correct prediction rate for $\mathcal{P}$ may result in the loss of prediction rate for $\mathcal{N}$. As suggested by Chawla (2010), the primary purpose in CIL is to improve recall while avoiding affecting precision. The test results of classifiers' F_1 scores and ranks indicate that although AROSS is optimized for recall, it does not adversely affect predictions for the majority $\mathcal{N}$. In fact, AROSS exhibits the highest average F_1 scores and ranks compared to both cluster-based approaches and other baselines.

When considering the average values and average F_1 ranks, AROSS consistently exhibited higher average values and lower average ranks compared to all random sampling and synthetic baseline methods. However, in terms of *Rec* and *GM*, AROSS displayed lower average scores and higher ranks compared to SENN and RWO, except when applied with the DT classifier. In the case of DT classifier, AROSS surpassed all baselines in terms of average *Rec* and *GM* values. Thus, SENN and RWO are superior than AROSS except when applied with DT classifier where it is superior to all baselines regarding the average *Rec*, and *GM* values. These results can be interpreted based on the characteristics of the ENN and AROSS approaches. As a data-cleaning approach ENN tends to remove some relevant instances from the majority class. This leads to a lower precision (*Pre*) and, consequently, a lower F_1 score than AROSS. On the other hand, AROSS employs an adaptive oversampling technique that focuses on generating synthetic instances that are more representative and relevant to the minority class. This results in a higher *Pre* and F_1 score for AROSS. Additionally, the ranking results of RWO from Table 8 suggest that the SVC classifier, which is highly influenced by the distribution of data points and the presence of CIL problems, was influenced by RWO. This could provide a favorable environment for SVC to identify positive and negative instances correctly. In the case of AROSS, it assists SVC in effectively classifying positive samples, potentially reducing the false positive (FP) rate and improving the precision-recall trade-off.

The results indicate that CIL approaches are competitive against each other and no significant improvement is noticeable in *AUC* for RF, SVC and kNN. Such high *AUC* scores across CIL methods indicate that they have similar abilities to discriminate between positive and negative samples. This also indicates that, in general, CIL approaches enhance the predictive performances of classification models. The results from Tables 7 and 8 suggest that AROSS is a particularly effective resampling approach when combined with DT, as it consistently outperforms all baseline methods across multiple performance metrics. It highlights the ability of AROSS to enhance the classification model's accuracy, discriminatory power and capturing positive instances, making it a promising choice for addressing CIL in decision tree-based classification tasks. Overall, the AROSS algorithm outperforms cluster-based baseline methods and most synthetic sampling approaches. It remains competitive with SENN and RWO regarding *Rec* and *AUC* scores. Taking into account the rankings in the AvgR results (Table 8), which summarize all performed results, AROSS achieves the lowest score (5.72), indicating its overall superiority. However, our proposed method is based on the optimization of cluster analysis, and better classification results come at the cost of more time complexity.

5.2. Statistical analysis using Welch's t-test

Based on the previous analysis, our assertion that AROSS is superior to or competitive with the compared CIL methods is primarily drawn from the average values and ranks of four metrics. However, it is important to note that each method exhibits varying standard deviations across different datasets. To thoroughly assess the statistical significance of AROSS's performance, we employed Welch's t-test (W_{test}, Derrick et al. (2016)). Several formal arguments substantiate the selection of W_{test} for benchmarking and comparing the performance of data-level resampling approaches (Zhang et al., 2008, Ellis et al., 2022, Shi et al., 2022, Darville et al., 2023). Unlike non-parametric tests such as Friedman or Wilcoxon, W_{test} is well-suited for handling imbalanced datasets, accommodating variations in variances among different resampling techniques. Its parametric nature allows it to consider both metrics average values and variances, providing a robust statistical evaluation that is particularly advantageous when dealing with real-world datasets exhibiting inherent imbalances (Shi et al., 2022). W_{test}, computed by the Eq. (13), serves as a reliable measure to validate the

[10] Detailed results. https://github.com/ghostqriver/AROSS/blob/main/Detailed%20experimental%20results

Table 7
Average of metric values for CIL approaches across 70 datasets.

		Random	Synthetic sampling										Cluster based					
	ORIG	ROS	S	AS	DS	SMPD	SENN	STL	NARS	GS	RWO	ANS	DBS	CS	KS	SOMO	AROS	AROSS
DT classifier																		
Rec	0.64810	0.64081	0.70624	0.70197	0.69886	0.65478	0.74953	0.70466	0.60253	0.68700	0.70290	0.66943	0.63311	0.66113	0.67142	0.66926	0.71142	**0.81337**
F_1	0.60827	0.61410	0.62541	0.61782	0.61816	0.60849	0.61362	0.62447	0.59433	0.61978	0.62396	0.61812	0.60297	0.60918	0.61937	0.62066	0.60466	**0.65809**
GM	0.73726	0.73146	0.77433	0.76779	0.76644	0.73831	0.78166	0.77437	0.70893	0.76083	0.77112	0.74919	0.72378	0.74464	0.75155	0.75290	0.76083	**0.82742**
AUC	0.78265	0.78376	0.79824	0.77653	0.79973	0.78309	0.80323	0.80117	0.76858	0.79654	0.80327	0.79076	0.77713	0.78588	0.79236	0.79190	0.80074	**0.84667**
RF classifier																		
Rec	0.61933	0.67131	0.72291	0.71540	0.71030	0.62008	**0.77322**	0.72299	0.60500	0.69938	0.72785	0.67111	0.62028	0.64935	0.64224	0.64805	0.70211	0.75461
F_1	0.63835	0.66432	0.67317	0.65796	0.67193	0.63760	0.66484	0.67264	0.61855	0.66460	0.68020	0.66461	0.63663	0.65021	0.65016	0.64971	0.65013	**0.68575**
GM	0.71348	0.75630	0.78941	0.77673	0.78302	0.71347	**0.80704**	0.78988	0.69882	0.77004	0.79006	0.75614	0.71296	0.73545	0.73018	0.73032	0.75892	0.80071
AUC	0.91557	0.92132	0.92034	0.91664	0.92009	0.91585	0.91131	0.91982	0.89756	0.91867	**0.92148**	0.91632	0.91665	0.91565	0.91476	0.91663	0.91255	0.91316
SVC classifier																		
Rec	0.53775	0.78625	0.77506	0.79397	0.75876	0.56797	**0.80635**	0.77403	0.63293	0.76769	0.79904	0.69821	0.68558	0.70118	0.59044	0.64377	0.71981	0.78245
F_1	0.57177	0.65305	0.65531	0.63921	0.64629	0.59595	0.64543	0.65410	0.61439	0.65687	0.65021	0.66538	0.64423	0.65946	0.59278	0.62359	0.63315	**0.66774**
GM	0.71348	0.75630	0.78941	0.77673	0.78302	0.71347	**0.80704**	0.78988	0.69882	0.77004	0.79006	0.75614	0.71296	0.73545	0.73018	0.73032	0.75892	0.80071
AUC	0.90386	0.90592	0.90701	0.89906	0.90492	0.90223	0.90148	0.90628	0.89853	0.90483	**0.90759**	0.90232	0.89385	0.90114	0.90333	0.89871	0.89499	0.89826
kNN classifier																		
Rec	0.58932	0.78564	0.80778	0.82697	0.78606	0.60671	**0.83231**	0.80860	0.67328	0.79695	0.80355	0.72095	0.66877	0.72064	0.64556	0.66723	0.67730	0.74269
F_1	0.60565	0.63775	0.63528	0.62809	0.63755	0.61456	0.62255	0.63544	0.62043	0.64104	0.62982	0.65016	0.62630	0.64483	0.61657	0.62689	0.63584	**0.67415**
GM	0.68458	0.81135	0.82237	0.82521	0.81512	0.69860	**0.83154**	0.82264	0.73767	0.82049	0.81937	0.78511	0.74488	0.78578	0.71365	0.73012	0.74580	0.79678
AUC	0.86675	0.86051	0.87277	0.86177	0.87097	0.86678	0.86462	0.87177	0.84763	0.87018	0.87432	0.86734	0.86062	0.86898	0.86397	0.86555	0.86852	**0.87480**

Table 8
Average rank scores for CIL approaches.

		Random	Synthetic sampling										Cluster based					
	ORIG	ROS	S	AS	DS	SMPD	SENN	STL	NARS	GS	RWO	ANS	DBS	CS	KS	SOMO	AROS	AROSS
DT classifier																		
Rec	11.80	12.87	6.87	6.87	7.69	11.43	4.96	7.34	15.20	8.56	7.41	9.49	13.14	11.16	10.20	10.86	6.51	**2.04**
F_1	10.20	9.66	8.74	8.99	9.43	10.21	8.81	8.56	10.77	9.16	9.01	8.76	10.79	10.54	8.93	9.76	10.47	**5.01**
GM	11.19	12.03	7.47	7.70	8.27	11.50	6.46	7.40	14.26	8.70	7.79	9.30	12.25	11.07	10.01	10.84	8.64	**2.93**
AUC	10.73	11.11	8.94	10.91	8.43	11.00	6.80	7.93	12.43	8.37	8.27	8.66	11.24	10.84	9.33	10.57	7.96	**2.57**
RF classifier																		
Rec	13.50	9.98	5.24	5.21	7.26	14.08	3.56	5.44	14.41	7.61	5.56	10.16	13.31	12.51	12.13	11.23	7.51	**3.53**
F_1	10.89	8.57	7.54	9.55	8.84	11.46	9.37	7.38	11.33	8.60	7.96	8.61	11.14	10.94	9.90	9.97	10.74	**6.57**
GM	12.86	9.79	6.33	7.09	8.10	13.46	6.13	6.01	13.36	8.11	6.41	9.97	12.74	12.44	11.69	11.05	9.35	**4.54**
AUC	9.32	7.94	8.64	10.48	9.27	8.42	11.21	9.00	12.45	8.50	**6.75**	9.12	9.34	8.80	8.47	9.18	10.44	10.05
SVC classifier																		
Rec	15.92	4.60	6.28	4.00	6.77	15.62	4.13	6.60	12.54	6.68	**3.47**	11.32	11.55	11.94	13.51	11.21	8.01	5.00
F_1	12.00	8.65	8.05	11.17	9.44	11.11	10.10	8.07	9.21	7.99	9.60	7.51	10.00	8.22	10.87	9.48	10.22	**7.50**
GM	15.02	6.97	6.88	7.70	7.92	14.64	6.98	6.82	11.27	7.22	**6.48**	9.68	11.41	10.80	12.74	10.74	9.30	6.61
AUC	8.60	8.51	8.16	11.22	9.00	8.80	10.41	8.41	10.84	8.85	**7.77**	8.98	10.65	9.42	9.64	9.41	10.61	9.98
kNN classifier																		
Rec	16.40	5.74	4.38	**2.43**	5.68	15.72	3.41	4.08	11.20	5.41	4.41	9.98	12.52	11.34	12.48	10.31	11.67	7.82
F_1	10.84	10.30	9.35	11.27	9.48	9.57	11.80	9.52	9.98	8.78	10.28	7.08	10.50	7.58	9.52	9.04	9.15	**4.91**
GM	14.75	8.44	6.58	7.42	7.50	14.05	7.77	**6.54**	10.68	6.88	7.20	9.01	12.15	9.91	11.58	10.48	11.28	6.74
AUC	7.78	12.82	8.67	12.42	9.34	7.54	11.50	9.07	13.21	9.54	7.88	7.81	11.32	7.97	9.22	9.68	7.55	**5.78**
AvgR	11.99	9.25	7.38	8.40	8.27	11.79	7.71	7.39	12.07	8.06	7.27	9.09	11.51	10.35	10.64	10.24	9.34	**5.72**
Final score	17	10	3	8	7	16	5	4	18	6	2	9	15	13	14	12	11	**1**

previous results, as it considers the distribution characteristics beyond just the mean values.

$$W_{test} = \frac{\overline{m_2} - \overline{m_1}}{\sqrt{s_2^2/\kappa + s_1^2/\kappa}} \tag{13}$$

The mean values of the metrics, such as *Rec*, F_1, *GM*, or *AUC* scores, for two CIL methods, are denoted as $\overline{m}_1$ and $\overline{m}_2$. Their standard deviations are represented as s_1 and s_2, and the total number of evaluation iterations is denoted as κ. Since we employ a stratified five-fold cross-validation approach repeated ten times, we have $\kappa = 50$. W_{test} follows a t-distribution, and its degree of freedom ν is approximately calculated as:

$$\nu \approx \frac{\kappa^2(\kappa-1)(s_2^2/\kappa + s_1^2/\kappa)^2}{s_2^4 + s_1^4} \tag{14}$$

The outcomes of the W_{test}, conducted between AROSS and the compared CIL methods, with a significance level of 0.05, are presented in Table 9, providing information on the number of datasets where AROSS outperforms, performs equally to, or underperforms the other methods, categorized as win-tie-lose, for each classifier.

For example, if we consider two sets of performance metrics, denoted as $method_A$ and $method_B$. For $method_A$, the mean ($\overline{m}_1$), variance (s_1^2), and sample size (n_1) are 0.85, 0.0005, and 50, respectively. Correspondingly, for $method_B$, the mean ($\overline{m}_2$), variance (s_2^2), and sample size (n_2) are 0.82, 0.0003, and 50. Substituting the given values in Eq. (13) and Eq. (14):

$$W_{test} = \frac{0.85 - 0.82}{\sqrt{0.0005/50 + 0.0003/50}} = 7.5$$

$$\nu \approx \frac{50^2(50-1)((0.0003/50) + (0.0005/50))^2}{(0.0003)^4 + (0.0005)^4} \approx 18679.36$$

The next step involves consulting a t-distribution table to obtain the critical values (t_{critical}) for a two-tailed test with $\nu \approx 18679.36$ degrees of freedom at a significance level of 0.05. In the given example, the critical value is calculated using the *scipy.stats*[11] module in Python. For a two-tailed test with $\nu = 18679.36$ degrees of freedom and a significance level of 0.05, the calculated t_{critical} is approximately 1.96. The decision-making process based on the t-statistic W_{test} is formalized by Eq. (15).

$$\text{Decision} = \begin{cases} \text{win}, & \text{if } W_{test} > t_{\text{critical}} \\ \text{loss}, & \text{if } W_{test} < -t_{\text{critical}} \\ \text{tie}, & \text{if } -t_{\text{critical}} \le W_{test} \le t_{\text{critical}} \end{cases} \tag{15}$$

In Eq. (15) formulation, a *win* is given if the calculated t-statistic is greater than the critical value, *loss* is given if the t-statistic is less than the negative of the critical value, and *tie* is given if the t-statistic falls within the range defined by the negative and positive critical values. Since $W_{test} > t_{\text{critical}}$ (in our example, $W_{test} > 1.96$), we would reject the null hypothesis, suggesting that $method_A$ statistically outperforms $method_B$. Therefore, in this case, $method_A$ is considered to have a *win* over $method_B$.

Few implications can be derived from the W_{test}:

- AROSS versus ORIG: The proposed approach consistently outperforms classification algorithms performance on the original datasets that are not augmented, with only a few instances where the original dataset performs better than AROSS.
- AROSS versus cluster-based methods: Compared to the five cluster-based methods, AROSS shows superior performance in *Rec*, F_1, *GM*, and *AUC* scores. Specifically, when using DT, AROSS achieves a significant number of wins (64) compared to CS, with only a few ties (6) and no losses (0). However, when using kNN, AROSS has a smaller number of wins (25) compared to CS, with a higher number of ties (34) and a moderate number of losses (11). Overall, AROSS achieves the highest number of wins (64) and the lowest number of losses (0) compared to the cluster-based baselines. Furthermore, AROSS demonstrates superiority in *GM*, with the highest number of wins (51) and no losses (0). The advantage of AROSS over cluster-based methods is also evident in F_1 and *AUC* scores presented in Table 9.
- AROSS versus data-level methods: Compared to data-level methods, namely random and synthetic sampling, AROSS consistently outperforms them in terms of *Rec*, F_1 score and *GM*, except for some cases where AROSS has fewer wins compared to SENN with RF, SVC, and kNN, with 16, 20, and 6 wins, respectively. Additionally, AROSS experiences losses in *Rec* against RWO with SVC and kNN. The analysis of *AUC* scores across different classification algorithms reveals a considerable number of ties, supporting the conclusion drawn from Tables 7 and 8, such that CIL approaches are competitive with each other and show no statistically significant differences in terms of *AUC*.

Overall, the results presented in Table 9 demonstrate that AROSS outperforms the compared cluster-based sampling methods statistically for DT, RF, and SVC classifiers. Furthermore, AROSS proves to be superior or competitive with kNN across the 70 datasets in a statistically significant manner. Compared to the random and synthetic sampling methods, AROSS outperforms them when paired with the DT classifier and is statistically significantly superior or competitive with the remaining classifiers.

5.3. Runtimes and system configurations

Runtime is an important aspect of data-level resampling techniques, as some applications may require retraining, thus rapid resampling procedures. Due to the number of resampling techniques involved in the experiments and parameters influencing their time complexities, presenting and analyzing time complexities in big-O notation is beyond the scope of this study. Nevertheless, we expect that the average runtimes of resampling techniques, as outlined in Table 10, can still offer meaningful insights into their time efficiency. It is important to note that runtimes are inherently contingent on the specific implementations employed. The simulations were executed in Jupyter Lab using Python 3.11 on a server featuring an Intel Core i7 processor. The processor has eight cores and 16 threads, with a max turbo frequency of 3.50 GHz and a processor base frequency of 2.50 GHz. Additionally, it is equipped with 11 MB of cache. In terms of memory, the machine is equipped with 64 GB of DDR4-2400 memory, and the maximum memory speed is 2400 MHz. Table 10 details the average runtimes in seconds for various data-level class imbalance approaches. ROS demonstrates a 3.658×10^{-2} seconds runtime within the random category, while synthetic approaches have runtimes ranging from 2.174×10^{-1} seconds for SMOTE to 7.231×10^{-1} seconds for SMPD. The cluster-based approaches have runtimes spanning from 4.317×10^{-2} seconds to 5.596 seconds. As expected, runtime disparities highlight the computational efficiency variations, with the cluster-based approaches exhibiting higher runtimes than random and synthetic methods. AROSS, which demonstrated superior classification performance, is recognized as the slowest approach in terms of runtime, attributed to incorporating various optimization steps. Specifically, the parameters L_{best}, c, and $|\mathcal{R}_i|$ undergo automatic fine-tuning for each dataset, while the shifting rate δ (refer to Algorithm 1) is a customizable hyper-parameter, contributing to the algorithm's adaptability. While these optimization steps contribute to the extended runtime, they are essential for tailoring AROSS to the specific characteristics of each dataset, which is particu-

[11] *scipy.stats* documentation. https://docs.scipy.org/doc/scipy/reference/stats.html

Table 9
Summary of Welch's t-test results for AUC, Rec, GM, and F_1 between CIL approaches and AROSS at a significance level of 0.05 (win-tie-lose).

		Random	Synthetic sampling										Cluster based				
	ORIG	ROS	S	AS	DS	SMPD	SENN	STL	NARS	GS	RWO	ANS	DBS	CS	KS	SOMO	AROS
DT classifier																	
Rec	62-8-0	61-8-1	54-14-2	53-15-2	54-16-0	59-11-0	40-23-7	54-15-1	68-2-0	57-13-0	54-16-0	55-15-0	61-9-0	64-6-0	58-12-0	59-11-0	53-17-0
F_1	35-32-3	33-30-7	31-32-7	31-29-10	32-33-5	36-30-4	23-40-7	29-35-6	35-33-2	32-34-4	28-37-5	27-39-4	33-31-6	31-35-4	32-35-3	29-35-6	33-36-1
GM	53-16-1	52-16-2	45-22-3	43-24-3	42-27-1	51-18-1	33-26-11	43-25-2	56-14-0	48-20-2	42-26-2	42-28-0	51-18-1	50-20-0	48-21-1	49-20-1	48-22-0
AUC	54-15-1	50-18-2	47-22-1	47-17-6	45-24-1	52-17-1	35-29-6	42-27-1	56-14-0	47-22-1	42-26-2	44-26-0	54-15-1	55-15-0	48-21-1	49-20-1	46-24-0
RF classifier																	
Rec	59-10-1	40-25-5	21-39-10	23-36-11	30-35-5	60-9-1	16-30-24	21-39-10	64-6-0	26-40-4	22-44-4	39-27-4	52-17-1	52-17-1	50-19-1	48-20-2	30-39-1
F_1	25-38-7	21-42-7	18-43-9	22-40-8	19-44-7	25-40-5	22-38-10	18-42-10	31-37-2	19-44-7	16-48-6	21-42-7	27-37-6	23-41-6	21-43-6	25-39-6	15-54-1
GM	49-20-1	31-34-5	20-40-10	25-34-11	23-41-6	50-19-1	19-33-18	19-40-11	52-18-0	22-43-5	18-48-4	26-40-4	43-25-2	37-32-1	39-30-1	38-30-2	28-42-0
AUC	7-51-12	5-54-11	7-47-16	12-40-18	5-51-14	7-49-14	15-40-15	8-48-14	26-35-9	4-47-19	1-60-9	5-55-10	7-50-13	5-52-13	5-56-9	5-57-8	1-69-0
SVC classifier																	
Rec	65-3-2	20-33-17	25-28-17	21-23-26	22-36-12	63-5-2	20-26-24	25-29-16	51-17-2	27-25-18	17-34-19	46-18-6	47-17-6	47-19-4	56-13-1	50-17-3	30-40-0
F_1	36-29-5	18-41-11	17-40-13	24-35-11	24-33-13	32-32-6	21-39-10	18-39-13	27-34-9	18-39-13	17-44-9	19-36-15	23-34-13	17-40-13	31-34-5	23-42-5	15-55-0
GM	53-17-0	18-36-16	21-35-14	25-29-16	22-36-12	52-18-0	16-39-15	19-37-14	39-28-3	19-38-13	14-41-15	30-33-7	31-32-7	32-32-6	46-24-0	39-28-3	24-46-0
AUC	14-41-15	13-41-16	14-41-15	19-33-18	13-44-13	13-43-14	18-40-12	14-41-15	19-41-10	13-43-14	11-44-15	10-47-13	18-38-14	10-48-12	16-46-8	11-52-7	5-64-1
kNN classifier																	
Rec	58-12-0	7-38-25	12-62-9	15-1-50	6-35-29	57-12-1	6-24-40	6-31-33	35-30-5	7-31-32	4-35-31	23-33-14	36-30-4	25-34-11	37-31-2	30-36-4	32-38-0
F_1	32-36-2	29-34-7	29-32-9	37-22-11	29-33-8	31-36-3	33-28-9	30-31-9	27-42-1	26-37-7	27-36-7	21-44-5	26-42-2	22-44-4	29-41-0	25-44-1	23-47-0
GM	48-22-0	15-43-12	10-39-21	16-32-22	10-48-12	48-22-0	11-39-20	12-37-21	31-38-1	11-43-16	11-42-17	17-46-7	35-32-3	24-41-5	36-33-1	29-39-2	25-45-0
AUC	12-57-1	29-39-2	16-47-7	27-32-11	14-49-7	14-55-1	26-38-6	18-45-7	32-37-1	17-48-5	10-54-6	9-55-6	21-48-1	11-56-3	10-60-0	13-55-2	5-65-0

Table 10
Average runtime in seconds of Data-level CIL approaches involved in the experiments.

Category	Sub-category	CIL approach (acronym)	Runtime (seconds)
Random	Oversampling	ROS	3.658×10^{-2}
Synthetic	Data generation	SMOTE	2.174×10^{-1}
		NRAS	3.668×10^{-1}
		RWO	7.264×10^{-2}
		GS	3.180×10^{-2}
	Adaptive	AS	2.806×10^{-2}
		DS	4.446×10^{-2}
		ANS	3.834×10^{-2}
		SMPD	7.231×10^{-1}
	Data cleaning	SENN	6.096×10^{-2}
		STL	5.296×10^{-2}
Cluster-based	Oversampling	DBS	7.332×10^{-1}
		CS	4.317×10^{-2}
	Adaptive	KS	2.202
		SOMO	1.468×10^{-1}
		AROS	2.678
		AROSS	5.596

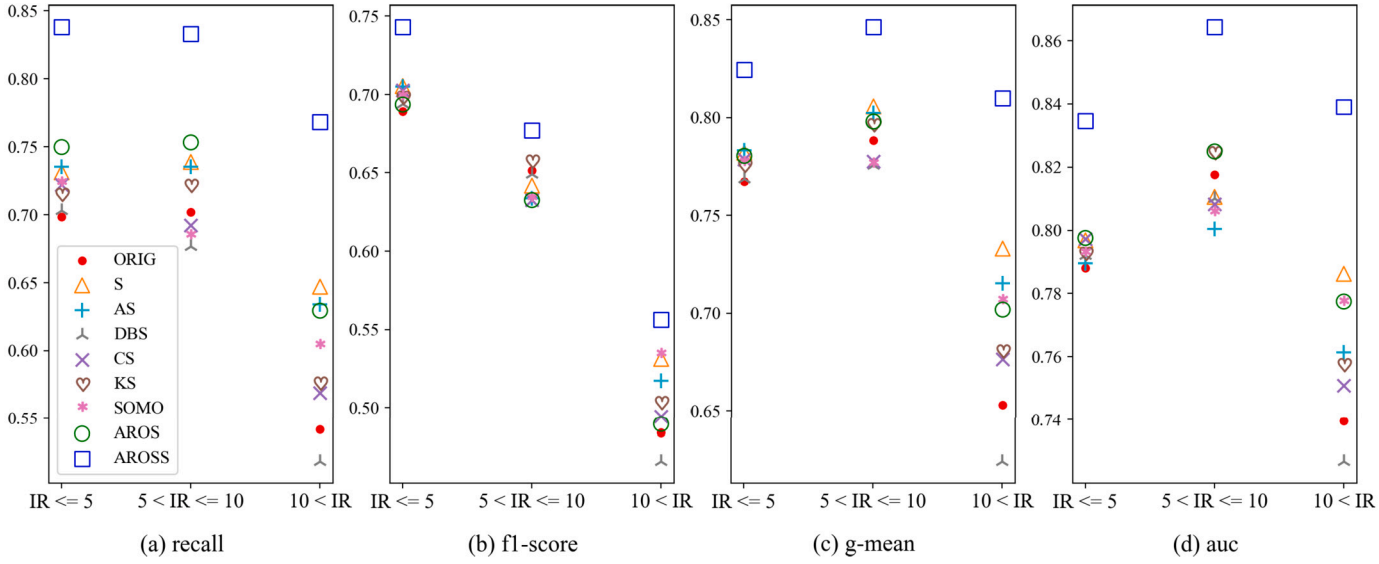

Fig. 11. Test results of different imbalance degrees when employing DT classifier.

larly beneficial for scenarios where we do not have enough data to train models. Therefore, despite its runtime, AROSS is still swifter than the alternative of collecting new data, making it an appealing option in resource-constrained situations.

For faster solutions, we recommend using RWO and SMOTE. According to the experimental results highlighted in Table 8, these algorithms were ranked 2nd and 3rd behind AROSS, concurrently sustaining competitive performance levels. Additionally, they demonstrate lower runtimes than AROSS. AROSS is a resilient solution characterized by a reasonable trade-off between runtime and performance. At the same time, RWO and SMOTE present swifter alternatives without substantial compromises in performance.

5.4. Extended analysis over CIL problems

This chapter overviews the learning barriers primarily affecting the performance of classification related to imbalance learning.

5.4.1. Small sample size and imbalanced class distribution

In the tables presented before, our proposed algorithm exhibits strong efficacy in alleviating the problem of lacking minority instances, further enhancing the predictive performance of classification on $\mathcal{P}$. Since we have 70 datasets with various class distributions, to present the performance of the proposed method under different distributions, we illustrate each test separately according to the dataset skewness.

Figs. 11-14 present the average evaluation results for different skewness levels on the datasets. Each figure consists of subplots representing the *Rec*, F_1, *GM* and *AUC* scores, respectively. The x-axis represents the skewness level, while the y-axis indicates the average performance metric value. For clarity, the figures incorporate the original dataset alongside representative baseline oversamplers, such as SMOTE (S) and ADASYN (AS). Additionally, cluster-based oversamplers, including DBSMOTE (DBS), CURE-SMOTE (CS), Kmeans-SMOTE (KS) and SOMO, are also included in these figures.

Overall, the performance on non-resampled dataset (the red dot) worsens on each metric while increasing imbalance, except for *AUC*, validating the previous hypothesis which states that a large IR influences the classification performance. From Fig. 11 and Fig. 12, if we exclude AROSS, AROS (the green circle) achieved the best recall with DT on slightly and medium imbalanced datasets while presenting competitive and positive results with severely imbalanced datasets on DT and all datasets on RF. Despite the imbalance degree, AROSS (the blue square) outperforms other oversamplers on DT and RF. How-

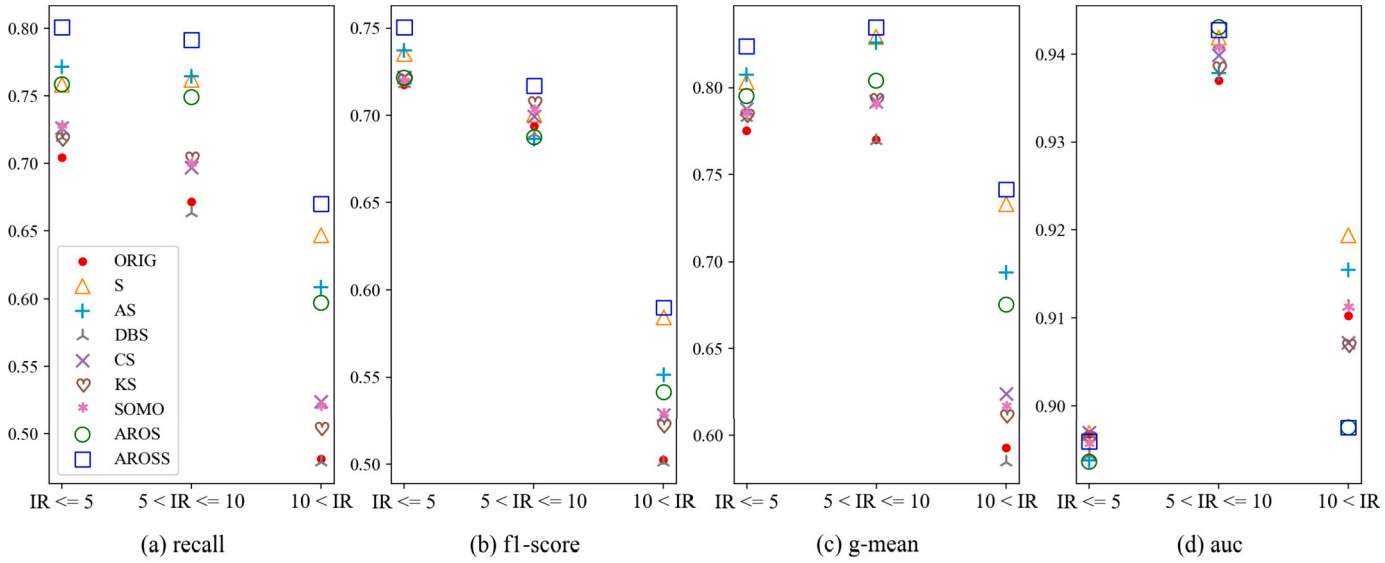

Fig. 12. Test results of different imbalance degrees when employing RF classifier.

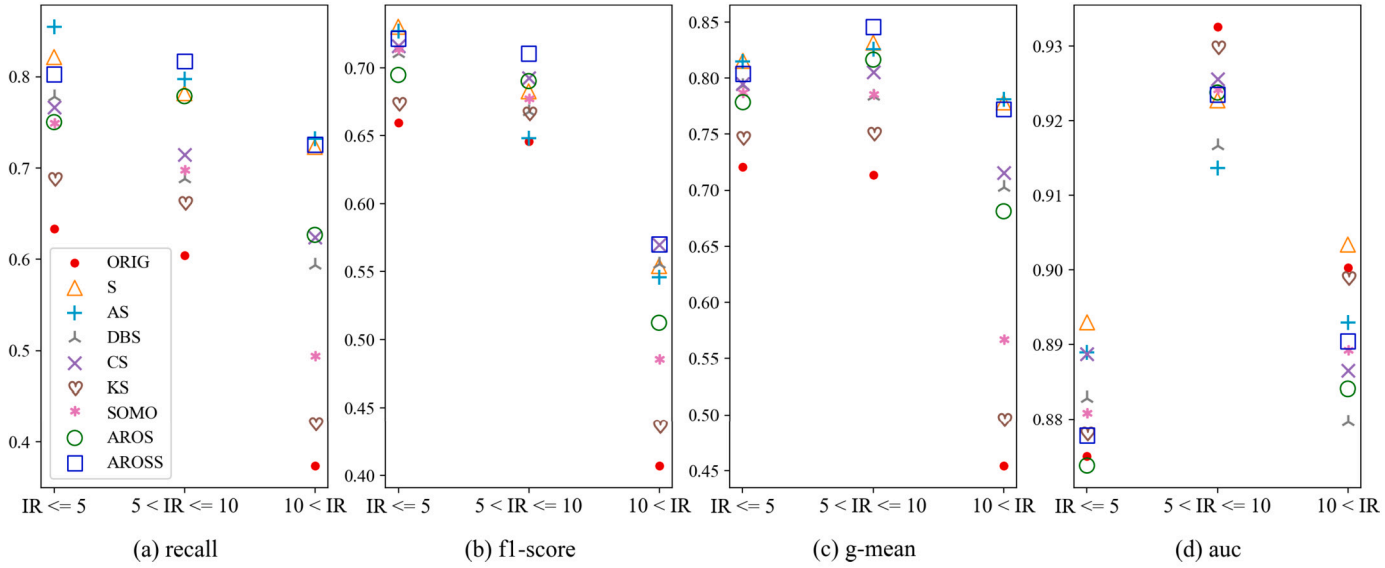

Fig. 13. Test results of different imbalance degrees when employing SVC classifier.

ever, it presents inferior *AUC* scores when employing RF, especially on data sets with $IR > 10$. AROSS may surpass the oversamplers other than SMOTE by a wider margin. As discussed, the proposed method exhibits weaker capability than SMOTE and some variants when combined with SVC and kNN. However, if concentrating on medium and severely imbalanced datasets, as shown in all metrics except *AUC* of SVC (Fig. 13) and F_1 on kNN (Fig. 14), AROSS shows superior results, nevertheless.

The performance discrepancy of the proposed approach can be attributed to the interplay between the algorithm and the chosen classifiers. Decision trees excel in handling imbalanced class distributions, while SVC and kNN classifiers have distinct sensitivities to class imbalance. SVC aims for optimal hyperplane placement but can be biased by imbalanced data, while kNN is influenced by data density and can overshadow minority instances. The varying performance of AROS and AROSS compared to other CIL approaches with different classifiers emphasizes the need to select appropriate resampling strategies based on the task's characteristics and algorithm sensitivities to class imbalance.

5.4.2. Class overlapping and small disjoint subsets

Overlapping between classes and small disjoint subsets of $\mathcal{P}$ might impose complexity on the classification task, which are the main problems we want to solve with the proposed method. The results displayed in Table 8 indicate that cluster-based sampling techniques may not be as effective as anticipated, primarily due to the characteristics and diversity of the datasets utilized in the study. Thus, a two-dimensional toy dataset has been created to describe how well cluster-based sampling approaches overcome such issues. In this dataset, $\mathcal{P}$ consists of several disjoint subsets and contains 200 data points. We then compared the results on this toy dataset on the *sampledata_2* dataset used in the benchmarking process, which contains subsets with a higher density of $\mathcal{P}$ points.

Combining the observations from Fig. 15 and Fig. 16, several insights can be drawn regarding the performance of CIL approaches. In Fig. 15, S and AS generate instances between nearest neighbors without concern about the distribution of $\mathcal{N}$. Thus, although they capture instances in small disjuncts and enhance sub-regions of $\mathcal{P}$, severe class

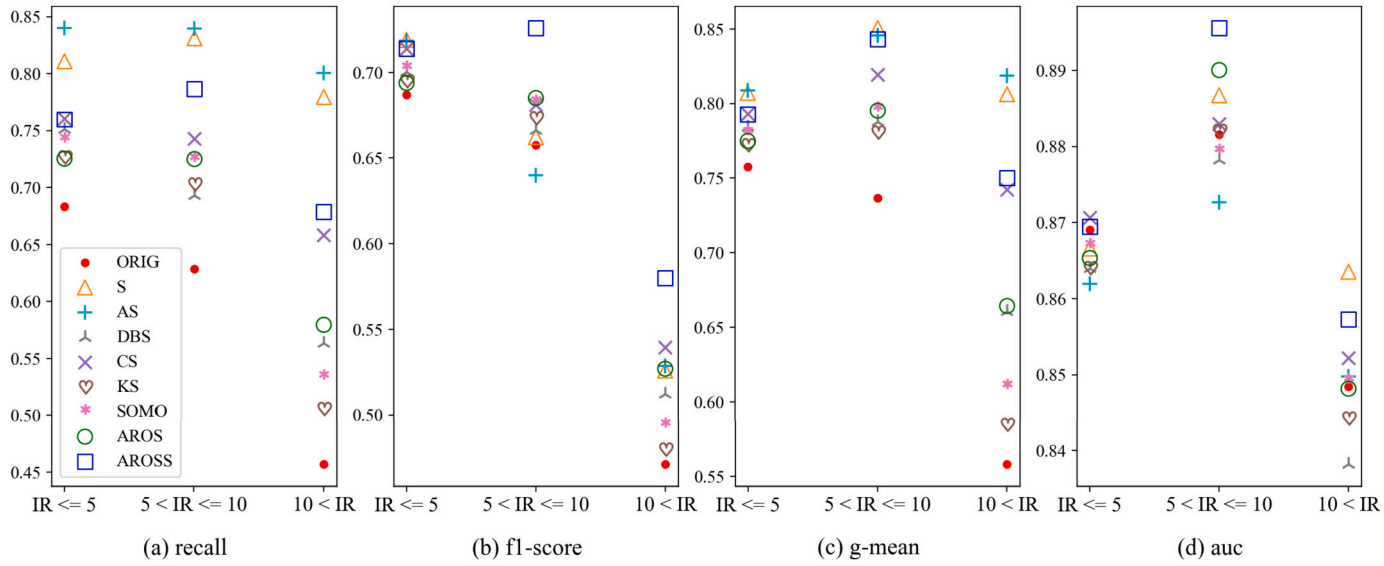

Fig. 14. Test results of different imbalance degrees when employing kNN classifier.

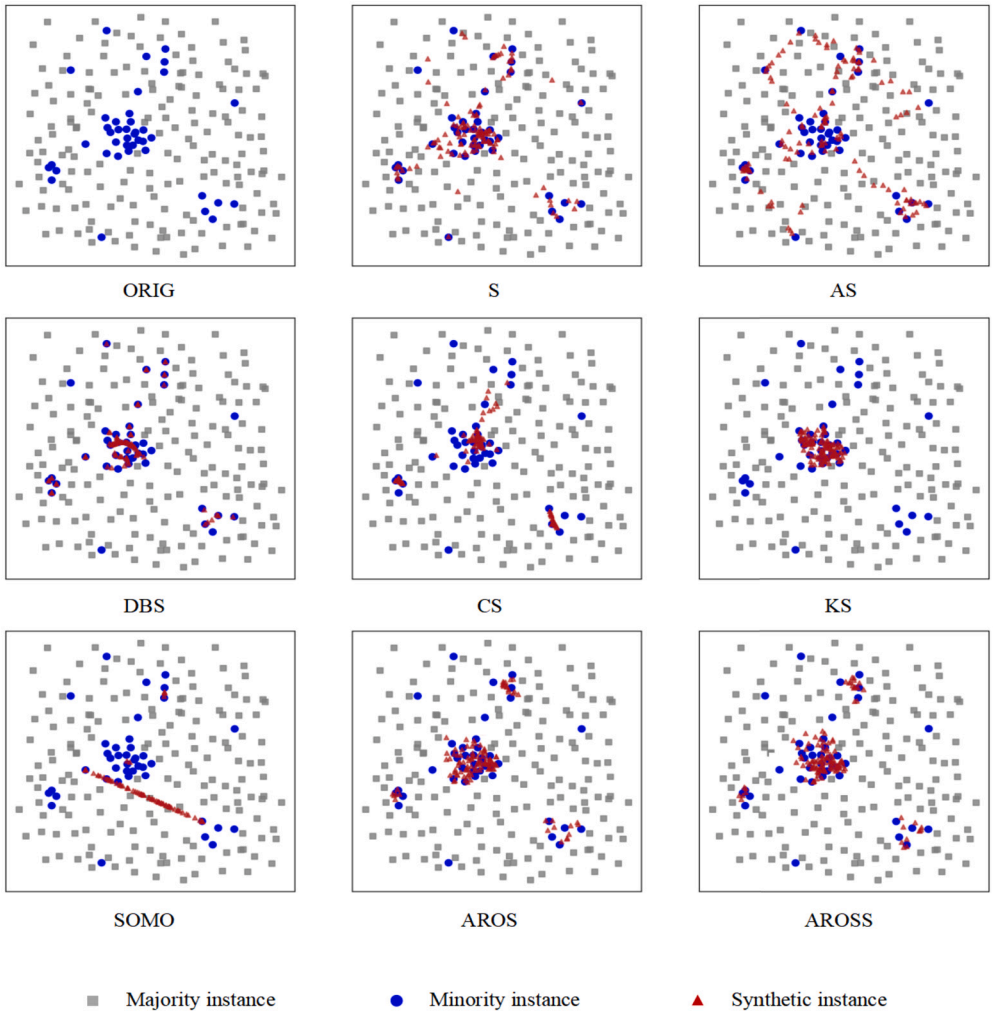

Fig. 15. Artificial samples generated by different CIL methods on a toy dataset with disjoint minority subsets.

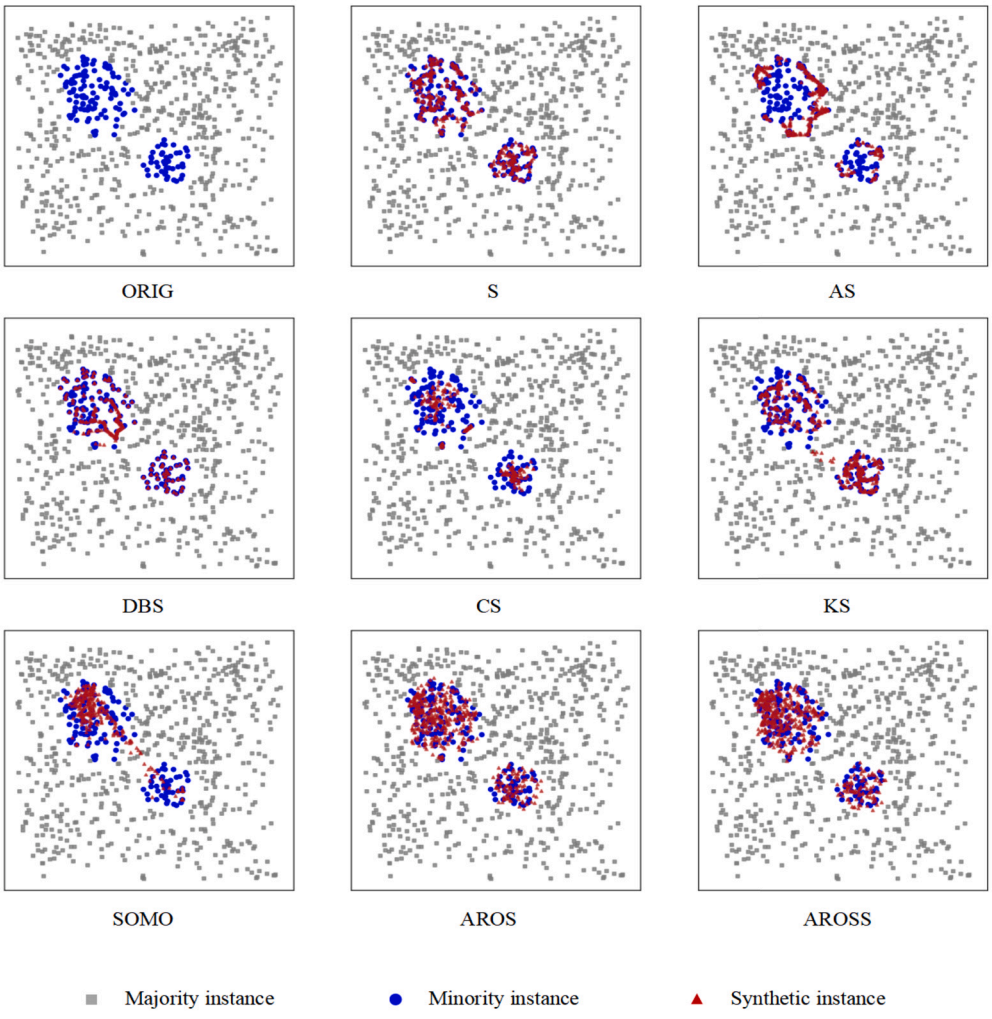

Fig. 16. Artificial samples generated by different CIL methods on *sampledata_2* dataset.

overlapping is caused due to the generation of many irrelevant synthetic samples; setting an appropriate value for the parameter k for SMOTE and AS could improve the oversampling efficiency. However, it is hard to determine k visually on a high-dimensional dataset with an implicit within-class imbalance issue, and the impact caused by noise is still unsolved. Therefore, Fig. 15 validates that cluster-based oversampling approaches effectively identify disjoint subsets. While DBS and CS also ignore the distribution of $\mathcal{N}$, they are dedicated to generating samples in the core regions of $\mathcal{P}$, where DBS generates along the skeleton path, and CS generates between the representative instances within clusters. Indeed, this type of approach is secure enough but does not help much to improve classification decisions, as points around the class border are the ones most relevant to building discriminative boundaries for classification models (Mullick et al., 2019). The invalidity of SOMO is attributed to its SOM clustering method which struggles to capture the complex distribution patterns and boundaries present in $\mathcal{P}$, particularly on 2-dimensional datasets. KS tends to eliminate clusters dominated by majority instances, potentially leading to the loss of many important minority class features. Thus, disjunct subsets may not be identified properly. Whereas AROSS consider both classes' distribution and generate instances in areas centered from representative points extracted from clusters instead of generating in clusters directly, where the representative points densely extracted from $\mathcal{P}$ and borderline could fragment the data space into sub-spaces and capture more refined sample distribution characteristics. Hereby, AROSS could identify almost every small disjunct subset of $\mathcal{P}$ and generate synthetic instances in the absence of overlapping. In addition, limited by the threshold k and half-safe condition in the incremental kNN strategy, singleton noises among majority instances will be ruled out of areas, thus, no samples will be generated nearby them.

Furthermore, in Fig. 16, DBS and KS apply SMOTE within clusters, resulting in similar effects as SMOTE. CS and SOMO focus on oversampling within the core regions of $\mathcal{P}$. However, KS and SOMO generate irrelevant instances that overlap across the two subsets of $\mathcal{P}$. AS oversamples along the borderline regions of minority subsets, but fails to detect some boundary instances and neglects critical features. While it enhances minority instances along the borderline, the increased density of synthetic instances can negatively impact the predictive correctness of nearby majority instances, leading to undesirable F_1 scores, particularly in kNN classification.

In sampledata_2 dataset, SMOTE yields more favorable results. Still, from both Fig. 15 and Fig. 16, we observed that oversampling effects of SMOTE and its variants present that synthetic samples are distributed along polygonal edges caused by the linear interpolation between minority class neighbors that SMOTE uses. Previously we mentioned that during the experimental phase, we set IR_e to 1 by default, which indicates that the same number of samples will be generated for each method. However, the figures show that samples generated by the SMOTE-based method are mainly aggregated in line segments. By applying the Gaussian generator, AROSS could synthesize samples to populate the sub-regions that minority instances may exist in, i.e., sub-spaces among minority instances without majority class invasion.

In addition, it can be found that the synthetic samples are widely distributed among the boundaries of $\mathcal{P}$ without overlapping with the sub-spaces of $\mathcal{N}$.

These observations underscore the distinct characteristics and limitations of different CIL approaches, highlighting the effectiveness of AROS and AROSS in generating synthetic samples that preserve the intrinsic boundaries of $\mathcal{P}$ without compromising the predictive performance on $\mathcal{N}$.

6. Conclusion

In this article, we tackled the challenges associated with class imbalance learning (CIL) by introducing a novel adaptive oversampling method called AROSS, which relies on cluster-based techniques. Our approach utilized an agglomerative clustering algorithm enhanced by integrating the cophenetic correlation coefficient (CPCC) and Bayesian information criterion (B_{score}), enabling us to efficiently determine the appropriate linkage and number of clusters, thereby pinpointing representative zones within the minority class. To better capture and enhance hard-to-learn minority instances more effectively, we applied a statistical sample size formula to identify the number of representatives in each identified zone. Subsequently, these zones were refined through an incremental kNN strategy to outline safe and semi-safe regions more effectively.

To address limitations observed in traditional SMOTE-based oversampling methods, which often cause subspace over-densing, we introduced a truncated hypercube Gaussian Generator. This innovation facilitated the even drawing of samples from safe areas following a Gaussian distribution, resulting in more precise and representative synthetic sample generation.

Our extensive experimental evaluation on 66 real-world and 4 synthetic datasets demonstrated the efficacy of AROSS in improving CIL performance, particularly with tree-based classifiers such as decision trees and random forests. AROSS consistently outperformed existing cluster-based, random, and synthetic sampling methods across various metrics, including Recall, F_1 score, Geometric mean, and AUC scores. The method exhibited significant advantages and minimal drawbacks, highlighting its robustness and versatility. Additionally, oversampling results on a toy dataset and the sampledata_2 dataset validated the effectiveness of our cluster-based oversampling approach for addressing small disjoint subsets imbalance problems. AROSS performed exceptionally well by precisely detecting small disjuncts without exacerbating class overlapping, making it particularly suitable for handling within-class imbalance issues.

However, it's crucial to acknowledge the limitation of our current implementation regarding runtime performance optimization. While our focus has been on the core resampling algorithm's effectiveness, we recognize the potential for further improvement in runtime efficiency.

Future work could explore the application of AROSS in other classification tasks, investigate its performance with different clustering algorithms to mitigate time complexity concerns, and consider approximating the number of clusters based on dataset characteristics as a substitute for B_{score}. Additionally, integrating domain-specific knowledge and analyzing the impact of various hyper-parameters could further enhance the adaptability and effectiveness of AROSS in diverse real-world scenarios.

CRediT authorship contribution statement

Zakarya Farou: Conceptualization, Data curation, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Yizhi Wang:** Data curation, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft. **Tomáš Horváth:** Methodology, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data and code availability

The source code and related datasets are available at https://github.com/ghostqriver/AROSS.

Acknowledgements

This research is supported by the ÚNKP-22-3 New National Excellence Program of the Ministry for Innovation and Technology from the source of the National Research, Development, and Innovation Fund, grant number ÚNKP-22-3-II-ELTE-393.

References

Alshemali, B., & Kalita, J. (2020). Improving the reliability of deep neural networks in nlp: A review. *Knowledge-Based Systems*, *191*, Article 105210.

Asuncion, A., & Newman, D. (2007). *Uci machine learning repository*.

Batista, G. E., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, *6*, 20–29.

Batista, G. E., Prati, R. C., & Monard, M. C. (2005). Balancing strategies and class overlapping. In *Proceedings 6. Advances in intelligent data analysis VI: 6th international symposium on intelligent data analysis, IDA 2005, proceedings 6* (pp. 24–35). Springer.

Bentley, J. L. (1990). K-d trees for semidynamic point sets. In *Proceedings of the sixth annual symposium on computational geometry SCG '90*. New York, NY, USA: Association for Computing Machinery (pp. 187–197).

Bi, J., & Zhang, C. (2018). An empirical comparison on state-of-the-art multi-class imbalance learning algorithms and a new diversified ensemble learning scheme. *Knowledge-Based Systems*, *158*, 81–93.

Bokhare, A., Bhagat, A., & Bhalodia, R. (2023). Multi-layer perceptron for heart failure detection using smote technique. *SN Computer Science*, *4*, 182.

Bunkhumpornpat, C., Sinapiromsaran, K., & Lursinsap, C. (2012). Dbsmote: Density-based synthetic minority over-sampling technique. *Applied Intelligence*, *36*, 664–684.

Cai, M., & Liang, Y. (2018). An improved cure algorithm. In *Intelligence science II: Third IFIP TC 12 international conference, ICIS 2018, proceedings 2* (pp. 102–111). Springer.

Chawla, N. V. (2010). Data mining for imbalanced datasets: An overview. In *Data mining and knowledge discovery handbook* (pp. 875–886).

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357.

Cieslak, D. A., Chawla, N. V., & Striegel, A. (2006). Combating imbalance in network intrusion datasets. In *GrC* (pp. 732–737).

Cios, K. J., & Moore, G. W. (2002). Uniqueness of medical data mining. *Artificial Intelligence in Medicine*, *26*, 1–24.

Cochran, W. G. (1977). *Sampling techniques*. John Wiley & Sons.

Cordón, I., García, S., Fernández, A., & Herrera, F. (2018). Imbalance: Oversampling algorithms for imbalanced classification in R. *Knowledge-Based Systems*, *161*, 329–341.

Darville, J., Yavuz, A., Runsewe, T., & Celik, N. (2023). Effective sampling for drift mitigation in machine learning using scenario selection: A microgrid case study. *Applied Energy*, *341*, Article 121048.

Davies, D. L., & Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-1*, 224–227. https://doi.org/10.1109/TPAMI.1979.4766909.

Derrac, J., Garcia, S., Sanchez, L., & Herrera, F. (2015). Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, *17*.

Derrick, B., Toher, D., & White, P. (2016). Why Welch's test is type I error robust. *The Quantitative Methods for Psychology*, *12*, 30–38.

Douzas, G., & Bacao, F. (2017). Self-organizing map oversampling (somo) for imbalanced data set learning. *Expert Systems with Applications*, *82*, 40–52.

Douzas, G., Bacao, F., & Last, F. (2018). Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. *Information Sciences*, *465*, 1–20.

Ellis, R. J., Sander, R. M., & Limon, A. (2022). *Twelve key challenges in medical machine learning and solutions*.

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the second international conference on knowledge discovery and data mining KDD'96* (pp. 226–231). AAAI Press.

Farris, J. S. (1969). On the cophenetic correlation coefficient. *Systematic Zoology*, *18*, 279–285.

Fotouhi, S., Asadi, S., & Kattan, M. W. (2019). A comprehensive data level analysis for cancer diagnosis on imbalanced data. *Journal of Biomedical Informatics*, *90*, Article 103089. https://doi.org/10.1016/j.jbi.2018.12.003. https://www.sciencedirect.com/science/article/pii/S1532046418302302.

Gosain, A., & Sardana, S. (2017). Handling class imbalance problem using oversampling techniques: A review. In *2017 international conference on advances in computing, communications and informatics (ICACCI)* (pp. 79–85). IEEE.

Guha, S., Rastogi, R., & Shim, K. (1998). Cure: An efficient clustering algorithm for large databases. *ACM Sigmod Record*, *27*, 73–84.

Han, H., Wang, W.-Y., & Mao, B.-H. (2005). Borderline-smote: A new over-sampling method in imbalanced data sets learning. In *Advances in intelligent computing: International conference on intelligent computing, ICIC 2005, proceedings, part I 1* (pp. 878–887). Springer.

Hazarika, B. B., & Gupta, D. (2021). Density-weighted support vector machines for binary class imbalance learning. *Neural Computing & Applications*, *33*, 4243–4261.

Hazarika, B. B., & Gupta, D. (2022). Density weighted twin support vector machines for binary class imbalance learning. *Neural Processing Letters*, *54*, 1091–1130.

Hazarika, B. B., & Gupta, D. (2023). Affinity based fuzzy kernel ridge regression classifier for binary class imbalance learning. *Engineering Applications of Artificial Intelligence*, *117*, Article 105544.

Hazarika, B. B., Gupta, D., & Borah, P. (2023). Fuzzy twin support vector machine based on affinity and class probability for class imbalance learning. *Knowledge and Information Systems*, 1–30.

He, H., Bai, Y., Garcia, E. A., & 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). IEEE.

Horváth, T., Mantovani, R. G., & de Carvalho, A. C. (2023). Hyper-parameter initialization of classification algorithms using dynamic time warping: A perspective on pca meta-features. *Applied Soft Computing*, *134*, Article 109969.

Jiang, C., Lu, W., Wang, Z., & Ding, Y. (2023). Benchmarking state-of-the-art imbalanced data learning approaches for credit scoring. *Expert Systems with Applications*, *213*, Article 118878. https://doi.org/10.1016/j.eswa.2022.118878. https://www.sciencedirect.com/science/article/pii/S0957417422018966.

Jiang, Z., Zhao, L., Lu, Y., Zhan, Y., & Mao, Q. (2023). A semi-supervised resampling method for class-imbalanced learning. *Expert Systems with Applications*, *221*, Article 119733.

Jo, T., & Japkowicz, N. (2004). Class imbalances versus small disjuncts. *ACM SIGKDD Explorations Newsletter*, *6*, 40–49.

Kamarulzalis, A. H., Mohd Razali, M. H., & Moktar, B. (2018). Data pre-processing using smote technique for gender classification with imbalance hu's moments features. In *Proceedings of the second international conference on the future of ASEAN (ICoFA) 2017–volume 2: Science and technology* (pp. 373–379). Springer.

Khan, M. T., & Sheikh, U. U. (2023). A hybrid convolutional neural network with fusion of handcrafted and deep features for fhss signals classification. *Expert Systems with Applications*, Article 120153.

Kovács, G. (2019a). Smote-variants: A python implementation of 85 minority oversampling techniques. *Neurocomputing*, *366*, 352–354.

Kovács, G. (2019b). An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets. *Applied Soft Computing*, *83*, Article 105662. https://doi.org/10.1016/j.asoc.2019.105662. https://www.sciencedirect.com/science/article/pii/S1568494619304429.

Kulkarni, A., Chong, D., & Batarseh, F. A. (2020). Foundations of data imbalance and solutions for a data democracy. In *Data democracy* (pp. 83–106). Elsevier.

Kunakorntum, I., Hinthong, W., & Phunchongharn, P. (2020). A synthetic minority based on probabilistic distribution (symprod) oversampling for imbalanced datasets. *IEEE Access*, *8*, 114692–114704.

Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, *18*, 559–563.

Liu, B., & Tsoumakas, G. (2020). Dealing with class imbalance in classifier chains via random undersampling. *Knowledge-Based Systems*, *192*, Article 105292.

Lu, H., Chen, C., Wei, H., Ma, Z., Jiang, K., & Wang, Y. (2022). Improved deep convolutional embedded clustering with re-selectable sample training. *Pattern Recognition*, *127*, Article 108611.

Lukasová, A. (1979). Hierarchical agglomerative clustering procedure. *Pattern Recognition*, *11*, 365–381.

Luque, A., Carrasco, A., Martín, A., & de las Heras, A. (2019). The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition*, *91*, 216–231. https://doi.org/10.1016/j.patcog.2019.02.023. https://www.sciencedirect.com/science/article/pii/S0031320319300950.

Lusardi, A., & Mitchell, O. S. (2014). The economic importance of financial literacy: Theory and evidence. *American Economic Journal: Journal of Economic Literature*, *52*, 5–44.

Ma, L., & Fan, S. (2017). Cure-smote algorithm and hybrid algorithm for feature selection and parameter optimization based on random forests. *BMC Bioinformatics*, *18*, 1–18.

Mantovani, R. G., Rossi, A. L. D., Vanschoren, J., Bischl, B., & Carvalho, A. C. P. L. F. (2015). To tune or not to tune: Recommending when to adjust SVM hyper-parameters via meta-learning. In *2015 international joint conference on neural networks (IJCNN)* (pp. 1–8).

McQuitty, L. L. (1960). Hierarchical linkage analysis for the isolation of types. *Educational and Psychological Measurement*, *20*, 55–67.

Merrild, H., Damgaard, A., & Christensen, T. H. (2008). Life cycle assessment of waste paper management: The importance of technology data and system boundaries in assessing recycling and incineration. *Resources, Conservation and Recycling*, *52*, 1391–1398.

Mullick, S. S., Datta, S., & Das, S. (2019). Generative adversarial minority oversampling. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 1695–1704).

Murtagh, F., & Contreras, P. (2012). Algorithms for hierarchical clustering: An overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*, 86–97.

Napierala, K., & Stefanowski, J. (2016). Types of minority class examples and their influence on learning classifiers from imbalanced data. *Journal of Intelligent Information Systems*, *46*, 563–597.

Nekooeimehr, I., & Lai-Yuen, S. K. (2016). Adaptive semi-unsupervised weighted oversampling (a-suwo) for imbalanced datasets. *Expert Systems with Applications*, *46*, 405–416.

Parthasarathy, S., Jayaraman, V., et al. (2023). Predicting heart failure using smote-enn-xgboost. In *2023 international conference on intelligent data communication technologies and Internet of things (IDCIoT)* (pp. 661–666). IEEE.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Prusty, S., Patnaik, S., & Dash, S. K. (2022). Skcv: Stratified k-fold cross-validation on ml classifiers for predicting cervical cancer. *Frontiers in Nanotechnology*, *4*, Article 972421.

Rivera, W. A. (2017). Noise reduction a priori synthetic over-sampling for class imbalanced data sets. *Information Sciences*, *408*, 146–161.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, *20*, 53–65. https://doi.org/10.1016/0377-0427(87)90125-7. https://www.sciencedirect.com/science/article/pii/0377042787901257.

Sandhan, T., & Choi, J. Y. (2014). Handling imbalanced datasets by partially guided hybrid sampling for pattern recognition. In *2014 22nd international conference on pattern recognition* (pp. 1449–1453). IEEE.

Santoso, B., Wijayanto, H., Notodiputro, K., & Sartono, B. (2017). Synthetic over sampling methods for handling class imbalanced problems: A review. In *IOP conference series: Earth and environmental science. IOP publishing: Vol. 58*, Article 012031.

Schubert, E. (2022). Stop using the elbow criterion for k-means and how to choose the number of clusters instead. arXiv:2212.12189.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 461–464.

Seifoddini, H. K. (1989). Single linkage versus average linkage clustering in machine cells formation applications. *Computers & Industrial Engineering*, *16*, 419–426.

Shi, H., Zhang, Y., Chen, Y., Ji, S., & Dong, Y. (2022). Resampling algorithms based on sample concatenation for imbalance learning. *Knowledge-Based Systems*, *245*, Article 108592. https://doi.org/10.1016/j.knosys.2022.108592. https://www.sciencedirect.com/science/article/pii/S0950705122002659.

Siriseriwan, W., & Sinapiromsaran, K. (2017). Adaptive neighbor synthetic minority oversampling technique under 1nn outcast handling. *Songklanakarin Journal of Science & Technology*, *39*.

Sneath, P. H. (1957). The application of computers to taxonomy. *Microbiology*, *17*, 201–226.

Sokal, R. R. (1958). A statistical method for evaluating systematic relationships. *The University of Kansas Science Bulletin*, *38*, 1409–1438.

Sun, J., Li, H., Fujita, H., Fu, B., & Ai, W. (2020). Class-imbalanced dynamic financial distress prediction based on adaboost-svm ensemble combined with smote and time weighting. *Information Fusion*, *54*, 128–144.

Swana, E. F., Doorsamy, W., & Bokoro, P. (2022). Tomek link and smote approaches for machine fault classification with an imbalanced dataset. *Sensors*, *22*, 3246.

Taherdoost, H. (2017). Determining sample size; how to calculate survey sample size. *International Journal of Economics and Management Systems*, *2*.

Thai-Nghe, N., Gantner, Z., & Schmidt-Thieme, L. (2010). Cost-sensitive learning methods for imbalanced data. In *The 2010 international joint conference on neural networks (IJCNN)* (pp. 1–8). IEEE.

Thorndike, R. (1953). Who belongs in the family? *Psychometrika*, *18*, 267–276.

Tomek, I. (1976). Two modifications of cnn. *IEEE Transactions on Systems, Man and Cybernetics*, *6*, 769–772.

Torres, F. R., Carrasco-Ochoa, J. A., & Martínez-Trinidad, J. F. (2016). Smote-d a deterministic version of smote. In *Pattern recognition: 8th Mexican conference, MCPR 2016, proceedings 8* (pp. 177–188). Springer.

Wang, X., Gong, J., Song, Y., & Hu, J. (2023). Adaptively weighted three-way decision oversampling: A cluster imbalanced-ratio based approach. *Applied Intelligence*, *53*, 312–335.

Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, *58*, 236–244.

Weiss, G. M. (2004). Mining with rarity: A unifying framework. *ACM SIGKDD Explorations Newsletter*, *6*, 7–19.

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man and Cybernetics*, 408–421.

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*, *14*. https://doi.org/10.3390/info14010054. https://www.mdpi.com/2078-2489/14/1/54.

Xia, H., An, W., & Zhang, Z. J. (2023). Credit risk models for financial fraud detection: A new outlier feature analysis method of xgboost with smote. *Journal of Database Management (JDM)*, *34*, 1–20.

Yang, S. J., & Cha, K. J. (2021). Gmote: Gaussian based minority oversampling technique for imbalanced classification adapting tail probability of outliers. ArXiv preprint arXiv:2105.03855.

Zhang, H., & Li, M. (2014). Rwo-sampling: A random walk over-sampling approach to imbalanced data classification. *Information Fusion*, *20*, 99–116.

Zhang, J., Jiang, T., Liu, B., Jiang, X., & Zhao, H. (2008). Systematic benchmarking of microarray data feature extraction and classification. *International Journal of Computer Mathematics*, *85*, 803–811.

Zhang, Z.-W., Jing, X.-Y., & Wang, T.-J. (2017). Label propagation based semi-supervised learning for software defect prediction. *Automated Software Engineering*, *24*, 47–69.

Zoric, A. B. (2019). Benefits of educational data mining. In *Economic and social development: Book of proceedings* (pp. 1–7).