IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 3 September 2025, accepted 15 September 2025, date of publication 30 September 2025,
date of current version 14 October 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3616207

RESEARCH ARTICLE

CSRBoost: Clustered Sampling With Resampling Boosting for Imbalanced Dataset Pattern Classification

SEEMA YADAV, DHRUVANSHU JOSHI, SOHAM MULYE, LABIB ASARI, SANDEEP S. UDMALE, (Senior Member, IEEE), AND GIRISH P. BHOLE, (Member, IEEE)
Department of Computer Engineering and Information Technology, Veermata Jijabai Technological Institute (VJTI), Mumbai, Maharashtra 400019, India

Corresponding author: Sandeep S. Udmale (ssudmale@it.vjti.ac.in)

This work was supported by the Centre of Excellence in Artificial Intelligence (CoE AI), Veermata Jijabai Technological Institute (VJTI), Mumbai, India.

ABSTRACT Data mining and machine learning (DM & ML) approaches frequently face class imbalance (CI) issues, especially in binary classification tasks when one class significantly outnumbers the other. Due to their propensity to favor the majority class, traditional DM & ML methods may exhibit biases due to insufficient oversampling and subpar performance in identifying instances of the minority class where minority classes often carry critical importance. It consequently raises concern regarding algorithmic fairness. To address CI difficulties, it is essential to increase the ability to identify different discriminatory patterns in the data by creating a large number of test cases. The proposed approach aims to achieve more fair and unbiased model performance. We provide CSRBoost, an ensemble learning method to tackle the CI problem. Three essential methods are combined in CSRBoost: AdaBoost, undersampling, and oversampling. To improve the model's capacity for generalization and provide a representative and balanced dataset, this approach provides the dynamic adjustments of clusters for sufficient granularity of the majority class in a controlled manner. It holds dataset-relevant structure information. Thus, the various horizons within the datasets remain unbroken. Besides, SMOTE and AdaBoost allow the model to adapt to complex boundaries by enhancing data diversity and minority class representation. CSRBoost is a dependable solution for CI data found in real-world classification tasks, as evidenced by its enhanced performance in handling imbalanced datasets.

INDEX TERMS AdaBoost, class imbalance, clustering, ensemble learning, oversampling, random undersampling (RUS), synthetic minority over-sampling technique (SMOTE).

I. INTRODUCTION

Unbalanced class distribution in a training data set poses significant challenges when constructing an efficient data mining and machine learning (DM & ML) model. This issue is known as class imbalance (CI), and it is primarily observed in binary classification problems. CI is prevalent in many real-world applications, such as medical diagnosis and cybercrime. In the context of CI, a particular class represents a dominant number of samples, known as the majority class. In contrast, another class has a minimal representative sample, known as the minority class [1], [2], [3]. The minority class is the main class of interest in DM & ML tasks. However, determining specific patterns in the minority class due to its limited samples is more challenging than in the majority class. Hence, it leads to DM & ML algorithms being biased toward the majority class, often neglecting the minority class altogether. As a result, algorithms trained on CI datasets may not perform effectively [2], [3], [4]. For example, a binary classification problem where the minority class consists of 1,000 samples and the majority class consists of 99,000 samples. Thus, 99.0% is the imbalance ratio. To reduce the error rate in such a situation, the DM & ML algorithm may classify all

The associate editor coordinating the review of this manuscript and approving it for publication was Senthil Kumar.

© 2025 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

samples into the majority class, which would result in a 1.0% of the misclassification rate. Here, every instance belonging to the minority class is critical and should be correctly identified, as overlooking these samples can lead to significant consequences.

Numerous approaches have been introduced to address CI issues. These techniques fall mainly into the following categories: algorithmic and data-level [5], [6]. The literature gives special consideration to data-level approaches, which concentrate on preparing the unbalanced data sets before building the classifiers. This is because classifier training and data preprocessing tasks can be completed separately [7]. Resampling the unbalanced training data set prior to the model training phase is the foundation of data preprocessing techniques. It is possible to resample the initial unbalanced data set by either undersampling the majority class or oversampling the minority class [5], [6], [8]. Both approaches aim to reduce the degree of imbalance, with a desire for a more well-rounded training regimen to produce superior outcomes. Both sampling techniques have been proven beneficial in imbalanced conditions and are simple to use as well as implement.

Therefore, a progressively undersampling method was developed to decide how many samples should be eliminated without losing valuable information [9]. Analogously, density information of minority classes has been utilized to remove samples from the majority class to mitigate the CI issue [10]. The relevant information of the majority class has been extracted to construct the undersampling approach to assist imbalanced data classification [11]. A convex-hull concept has been adopted to propose a novel clustering-based undersampling method for creating data sample partitions to preserve the general pattern present in feature space [12]. Similarly, an adaptive clustering method has been introduced based on dataset size and sampling characteristics to enhance the efficiency of undersampling in the CI problem [13]. An equalization undersampling technique has been employed to identify the instance contribution facts and retain the highly contributing instances to stop the information loss due to the undersampling approach [14]. A class-overlap issue of CI data has been addressed by identifying the global similarity of datasets through the Schur decomposition-based undersampling method [15]. The metaheuristic-based optimization method has been adopted to determine an optimal subset from the majority class to handle the CI issue [16]. The literature indicates that the various undersampling approaches have functioned on the majority class samples to match the class distribution with the minority class. In this process, they attempted to maintain the general trend of feature space or data representation. As a result, shorter training times have been achieved through undersampling approach but at the expense of potentially losing valuable data. It affects the classifier's performance since it creates a biased sample, leading to underfitting. Consequently, concern regarding algorithmic fairness grows.

Conversely, oversampling preserves all of the information from the original dataset by retaining each example in the resampled data. Depending on the oversampling strategy, the resampled dataset either includes duplicates or newly generated minority class samples. By randomly interpolating pairs of closest neighbors in the minority class, synthetic minority oversampling technique (SMOTE) [17] introduced additional synthetic minority class examples. However, the over-generalization in SMOTE generates noisy samples, and the classes overlap near the class borders. As a result, it affects the performance of various classifiers. Thus, various variants of SMOTE have been reported in the literature. Recently, range-controlled SMOTE enhance the sample generation process by managing the data space [18]. The imbalanced dataset of regression tasks has been resolved by developing geometric SMOTE to improve the prediction of continuous values [19]. k-nearest neighbour (kNN) oversampling method identifies crucial and secure locations for augmentation and creates synthetic data samples for minority classes [20]. SMOTE requires exploring the kNNs to develop synthetic instance. Most of the SMOTE variants, like [21], [22], attempt to resolve the issue of determining the nearest neighbors of samples to solve the CI problem. The literature indicates that SMOTE is one of the most popular methods for balancing datasets. Besides, the various oversampling approaches have functioned on the minority class samples to match the class distribution with the majority class. It improves the capability of DM & ML models to generalize and accurately predict minority classes. However, oversampling necessitates more training time since it expands the size of the training set. Additionally, because it replicates minority class examples, it frequently results in overfitting, and hence, algorithmic fairness is a major concern.

Boosting is an additional method to improve classification performance. Boosting is sometimes described as an improved data sampling method. Unlike many data sampling algorithms designed specifically to solve CI, boosting improves the performance of any weak classifier, regardless of data balance [2], [23], [24]. To enhance, ''reweighting'' or ''resampling'' is applied on misclassified instances. Each iteration's model is then created using the resampled data. A strong model with enhanced generalization capabilities results from this iterative approach [23], [24]. The most popular boosting method, AdaBoost (Adaptive Boosting) [25], iteratively creates an ensemble of models.

Recent advances in deep learning (DL) have enabled achieving success in various real-world applications. Thus, a detailed analysis between DL architectures and CI data sets has been performed to explore their relationship [26]. Similarly, the impact of various sampling methods has been studied using numerous DL architectures to analyze the performance of the CI dataset of network intrusion detection systems [27]. Also, various DL algorithms have been examined under diverse disease CI scenarios through handwritten medical discharge notes [28]. A novel convolutional neural

network (CNN) learning approach has been proposed to tackle the CI learning problem to improve performance [29]. Also, a modified CNN framework has been developed to handle the CI issue in medical imaging [30]. A more realistic dataset has been synthesized from the original dataset using the modified generative adversarial network (GAN) to address the issues of the traditional oversampling method for power systems [31] and other benchmarking standard CI datasets [32]. The literature indicates that the DL model shows notable performance volatility on the test dataset and poor generalization. This is observed due to the unstable gradient updates during the training phase, which makes it harder to balance trade-offs between varying distributions of classes.

Above literature informs that various clustering approaches have been adopted for undersampling, among which k-means clustering stands out for its simplicity [2], [24]. For example, in CUSBoost [33], the k-means clustering approach has been applied to the majority classes as an undersampling approach. A similar type of work was observed in [34] and [35]. However, the challenge of choosing an appropriate 'k' value of the k-means clustering approach has not been addressed, despite extensive research involving a variety of CI datasets over the years. To comprehend its link and overall influence on classifier performance, the proportion of clustering inside the majority class compared to the minority class distribution has also been methodically investigated. Through this technique, we can measure the degree to which the clustering mechanism aligns with the minority class's decision boundary while maintaining the majority class's structural variety. By examining this percentage, we aim to clarify the relationship between the degree of clustering and the resampling balance, emphasising its impact on the classifier's learning dynamics and the generalisation performance that results. Besides, artificial oversampling techniques have proven to be effective in correcting CI datasets. However, concerns over algorithmic fairness are increasing since it has been demonstrated that data-driven algorithms produce biases. This problem arises because the majority class distribution is not considered while building synthetic instances. Building many test cases aims to increase the capacity to recognize various discriminatory patterns in the data, leading to more impartial and equitable model performance.

In this article, we propose a unique hybrid technique, CSRBoost, that combines sample resampling and ensemble techniques to enhance the performance of models trained on CI datasets. CSRBoost is a boosting model framework that addresses CI by combining random undersampling with SMOTE. It constructs the clusters in the majority class based on minority class instances. To improve the model's capacity for generalization and provide a representative and balanced dataset, this approach provides the dynamic adjustments of clusters for sufficient granularity of the majority class in a controlled manner. It holds dataset-relevant structure information. Thus, the various datasets horizons remain unbroken and preserve the local distribution pattern. It operates random undersampling to locate the representatives from clusters developed in the majority class. It reduces the data distribution issue and improves the classifiers ability to detect instances of minority classes. By reducing duplication in the majority class, this technique creates distinct training subsets, enriches the solution space, and prevents bias towards the majority class. After clustering, number of instances in majority and minority class may vary. Then, it employs SMOTE to redefine the class definition. The basic classifiers are trained on these diverse subsets, and in subsequent iterations, the model is improved iteratively by focusing on misclassified cases through boosting. CSRBoost combines these classifiers with boosting to create a robust and effective final model that can handle highly unbalanced input.

The structure of the paper is as follows: Section II presents the theoretical foundation of resampling techniques and Adaboost, as well as providing a detailed discussion of the proposed method. Section III presents the experimental setup and the CI dataset, enabling a comprehensive analysis of the results. Finally, research contributions are highlighted, and the conclusions drawn from findings are summarised in Section IV.

II. THEORETICAL BACKGROUND

This section introduces the basic concepts of SMOTE, random undersampling, and AdaBoost. The fundamental concepts of each technique are first explained, providing an application viewpoint. The proposed hybrid approach, i.e., CSRBoost method, which integrates these methods to solve class imbalance more effectively, is then presented.

A. SYNTHETIC MINORITY OVER-SAMPLING TECHNIQUE (SMOTE)

The SMOTE is a widely used method for addressing CI in datasets [17]. SMOTE improves the effectiveness of classification algorithms on unbalanced data by balancing the class distribution by producing synthetic instances instead of merely replicating minority class examples. Thus, it lowers the chances of overfitting. As a result, models are better at identifying important trends and generalizing them [2], [5], [6], [24]. The algorithmic steps of the method are as follows:

1) Find instances that belong to the minority class: Consider the CI dataset D of size N $(N_{maj} + N_{min})$ with X_{maj} (instances from the majority class) and X_{min} (instances from the minority class). Thus, D is represented as $\{(X_i, Y_i)\}_{i=1}^{N}$. The objective is to generate synthetic samples to increase the number of X_{min} instances.
2) Choose the closest neighbours: Euclidean distance based KNN applied to each instance $x_i \in X_{min}$ to find the closest neighbours (CN). A set of KNNs for x_i can be represented by CN(x_i).
3) Create synthetic data: By choosing one of the CNs, x_{CN}, and interpolating a new point between x_{CN} and x_i, a synthetic instance x_{sys} is created. To ensure that x_{sys} is located on the line segment between x_{CN} and x_i, random number generator $\eta \in [0, 1]$ is used. Thus, the

expression is given as:

$$x_{sys} = \eta \times (x_{CN} - x_i) + x_i \tag{1}$$

4) Repeat: This operation is repeated to balance the distribution of classes until the number of X_{min} instances is raised to the required level.

The dataset is now balanced after executing the above step; classifiers can focus on the X_{min} to better detect rare but important patterns, resulting in improved F1 scores and recall [2], [5], [6], [24]. Consequently, several application domains accept SMOTE to resolve CI problems. It is used in conjunction with several classification processes. Additionally, several studies have demonstrated that SMOTE is superior to alternative oversampling techniques due to its adaptive resolution of the CI problem [18], [19], [20], [21], [22]. Besides, the KNN operated in SMOTE may be far from the target point in high-dimensional datasets, resulting in interpolation and the creation of x_{sys} that may not be suitable for classification. Because SMOTE requires CN calculation, it can be computationally costly for large datasets [2], [5], [6], [18], [19], [20], [21], [22], [24].

B. RANDOM UNDERSAMPLING (RUS)

RUS is one of the most often used data sampling methods, mainly because of its ease of use. In contrast to more intricate data sampling methods, RUS does not try to "intelligently" eliminate examples from the training data. Rather, RUS only randomly eliminates examples from the dominant class until the intended class distribution is reached [36], [37]. The algorithmic steps of the method are as follows:

1) Find size of X_{min}: Consider the CI dataset D with the objective of finding the size N_{min} of the X_{min} instances.
2) RUS for X_{maj}: Choose N_{min} samples randomly belonging to X_{maj} without replacement to create a subset $\bar{X}_{maj}$.
3) Create balanced dataset: Combine $\bar{X}_{maj}$ and X_{min} instances to form a balanced dataset $\bar{D}$.

RUS is simple to use and computationally cheap compared to more intricate sampling techniques. Greatly cuts down on training time by reducing the amount of the dataset. Limiting the X_{maj} may improve the generalization of models in situations where the X_{maj} controls the decision boundary [36], [38], [39], [40]. However, RUS eliminates potentially significant data from the X_{maj}, which may hinder the model's generalizability. Because sample selection is random, the outcomes of RUS can differ significantly between runs. Thus, ensemble techniques like Adaboost mitigate these issues due to their working principles [36], [37].

C. ADABOOST

AdaBoost's primary concept is to modify the weights of the data points incorrectly classified by classifiers in the previous iteration. As a result, these challenging examples receive greater attention from the following classifiers. Because each time, a classifier is learned based on the performance of those already taught. Although it may also be applied to multiclass issues, it is most commonly employed for binary classification jobs. The main purpose of the potent ensemble learning method is to enhance the performance of weak classifiers [25]. The algorithmic steps of the method are as follows:

1) Weights ($w^i_{(1)}$) initialization: Consider the binary classification problem dataset $D = \{(X_i, Y_i)\}_{i=1}^{N}$. Initially, for each x_i, $w^i_{(1)} = \frac{1}{N}$.
2) Weak classifier ($h_z(x)$) training: To minimize the error ε_z, $h_z(x)$ is trained on the weighted D in each iteration z. If the $h_z(x_i)$ classifies correctly, the function $\jmath()$ returns 0; otherwise, it is 1. This can be calculated as

$$\varepsilon_z = \sum_{i=1}^{N} w^i_{(z)} \cdot \jmath\,(y_i \neq h_z(x_i)) \tag{2}$$

3) Classifier weight (α_z) calculation: Determine the classifier's α_z, which indicates its importance, after training $h_z(x)$. More accurate classifiers are assigned better weights as the ε_z decreases and α_z increases simultaneously.

$$\alpha_z = \frac{1}{2} ln\left(\frac{1-\varepsilon_z}{\varepsilon_z}\right) \tag{3}$$

4) Update w: For the next $h_{z+1}(x)$, the $w^i_{(z)}$ of the misclassified x_i are increased to highlight the challenging instances. The ϱ_z (normalization factor) is used to update the new $w^i_{(z+1)}$ for each x_i as follows:

$$w^i_{(z+1)} = \frac{w^i_{(z)}}{\varrho_z} \times \begin{cases} e^{-\alpha_z} & \text{if } \jmath\,(y_i = h_z(x_i)) \\ e^{\alpha_z} & \text{if } \jmath\,(y_i \neq h_z(x_i)) \end{cases} \tag{4}$$

5) Final classifier ($H(x)$): It is the weighted sum of the ($h_z(x)$). It is expressed as follows:

$$H(x) = sign\left(\sum_{z=1}^{Z} \alpha_z h_z(x)\right) \tag{5}$$

To effectively handle noisy and complex data distributions, each z gives more w to cases that are difficult to classify. AdaBoost significantly improves performance by transforming $h()$ into $H()$ with only slightly higher performance than random guessing. Because AdaBoost doesn't require adjusting a lot of hyperparameters, it is relatively simple to utilize compared to other ensemble algorithms. It can be applied to weak learners, such as decision trees and perceptrons. Compared to other ensemble methods, AdaBoost is less likely to overfit, especially when using simple weak learners like decision stumps. AdaBoost may be problematic when working with noisy data or outliers because of its propensity to overemphasize noisy or erroneous samples by assigning larger weights to misclassified cases. The method might need many iterations or boosting rounds to perform well, increasing processing costs and making the model more complex. AdaBoost may only provide a significant performance boost if the weak classifiers are straightforward and operate almost arbitrarily because it

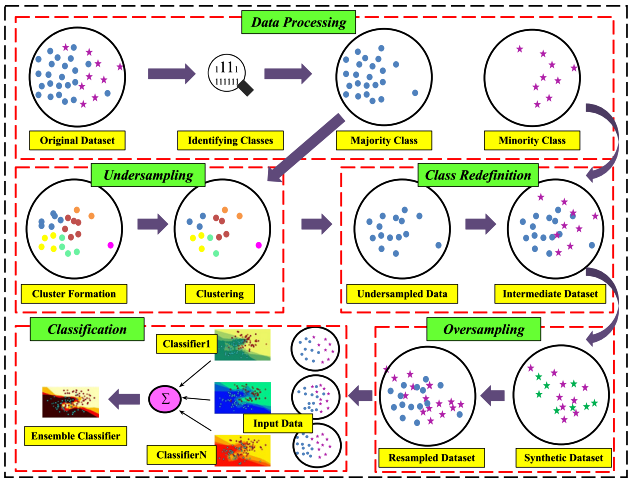


FIGURE 1. Proposed CSRBoost method for class imbalance problem.

depends on bolstering weak learners. Training many weak classifiers can be computationally demanding, especially for large datasets, although it is usually less computationally demanding than methods like random forests or gradient boosting [41], [42].

D. CSRBOOST METHOD

To deal with the common issue of CI in supervised learning tasks, the CSRBoost approach has been developed. CI refers to the unequal distribution of data across classes, often leading to the model performing poorly in the X_{min}. CSRBoost attempts to address this problem by integrating boosting via AdaBoost, cluster-based undersampling, and oversampling through SMOTE into one ensemble learning framework as shown in Fig. 1. CSRBoost algorithm applies intelligent dataset rebalancing with two-step resampling before model training. A cluster representative sample among the majority population is created to apply selective undersampling after minimizing redundancy. As a result, SMOTE, which supplements feature space devoid of overfitting risk through sample creation (pseudo-sample generation), can intervene. After resampling, CSRBoost applies AdaBoost, a widely used boosting approach that changes sample weights on training rounds to focus on harder-to-classify events in subsequent rounds. The model's generalization ability in both classes is strengthened with this repetitive attention. Algorithm 1 captures all of CSRBoost's processes. Balanced classification results, especially with unbalanced datasets, are attributed to the optimal combination of the well-defined pipeline's thorough representation of X_{min} and X_{max} samples during the training phase. The detailed discussion of CSRBoost is as follows:

1) CLUSTERED UNDERSAMPLING OF THE MAJORITY CLASS

Consider the binary classification CI dataset D of size N with X_{min} and X_{max} samples. Initially, CSRBoost identifies and separates X_{min} and X_{max} samples before attempting

Algorithm 1 CSRBoost: Clustered Sampling With Resampling Boosting for Imbalanced Dataset Pattern Classification

Input: Base estimator $h_z()$, number of clusters $n_{clusters}$ (optional), number of estimators n_{est}, number of neighbors k, random state r, clustering percentage p

Output: Ensemble classifier H

Initialize AdaBoost classifier with base estimator $h_z()$ and n_{est}
Identify X_{min} and X_{maj} from training data
Determine N_{min}, N_{maj}, and $n_{clusters}$ (if not provided using Eq. 6.)
Apply K-Means clustering with $n_{clusters}$ on X_{maj}
For each cluster, randomly under-sample X_{maj} samples
Combine X_{min} with the under-sampled X_{maj} samples to form (X', y')
Re-identify new minority class X'_{min} and new majority class X'_{maj} in (X', y')
Apply SMOTE on X'_{min} to balance with X'_{maj}
Train AdaBoost classifier on the balanced dataset
return Ensemble classifier H

to target resampling. To maintain the intrinsic structure of the data, CSRBoost employs a cluster-based undersampling methodology, whereas traditional random undersampling techniques risk losing important X_{max} samples. Two core steps are to organize and maintain the structural diversity of the X_{max} while systematically reducing its predominance during the cluster-based undersampling process:

1) Cluster formation: K-Means clustering [43] is adopted to classify X_{max} samples into distinct categories. This unsupervised algorithm clusters similar samples based on proximity in the feature space to minimize intra-cluster variabilities. The number of clusters $n_{clusters}$ are formed either
   - Predefined by the user, or
   - The creation is based on the size of minority class and dynamically adjusted using the following formula:

$$n_{\text{clusters}} = \max\left(1, \frac{p}{100} \times |X_{min}|\right) \tag{6}$$

   where p represents the clustering percentage parameter (defaulting to 100.0%), and $|X_{min}|$ is the cardinality of the X_{min}.

   The dynamic adjustments ensure that the correct $n_{clusters}$ is established according to the imbalance ratio. It provides sufficient granularity for the X_{maj} and avoiding excessive fragmentation. CSRBoost's use of K-Means clustering lays the foundation for a smarter undersampling algorithm by recognizing and preserving the local data distributions of the X_{maj}.
2) Cluster-level undersampling: To obtain a reduced majority subset X_{sub_maj}, roughly half of the samples of each cluster are randomly selected for undersampling after the clusters have been created as discussed in

Section II-B. A intermediate dataset $D' = (X', y')$ is produced at the end of this step and it is expressed as:

$$X' = X_{min} \cup X_{sub_maj} \tag{7}$$

This localized sampling ensures that any region of the feature space occupied by the X_{maj} will be represented proportionally. Unlike global random undersampling, where randomly removing samples might lose important information, major and potentially anomalous patterns are less likely to be discarded prospectively from each cluster. By holding structure-relevant data points while keeping the decision boundary between classes intact, this cluster-aware undersampling method improves the quality of the resampled dataset. The next step of CSRBoost employs SMOTE and operates on D' to further balance the classes before training the model.

2) MINORITY OVERSAMPLING WITH SMOTE

The class distribution in the D' may change significantly after the cluster-based undersampling stage. Specifically, it's possible that there are more X_{min} samples than X_{maj} ones. A class redefinition phase is included to ensure that X_{min} and X_{maj} are defined consistently across the training pipeline:

1) Class redefinition: Let $|X'_{min}|$ and $|X'_{maj}|$ represent the current cardinalities of the minority and majority class samples, respectively, after undersampling. If $|X'_{\text{min}}| > |X'_{\text{maj}}|$ then the class labels are redefined as follows:
   - The class with fewer samples becomes the new minority class.
   - The class with more samples becomes the new majority class.

   To prevent imbalance reversal and keep the algorithm consistent, this dynamic reallocation ensures that the next oversampling step is always applied to the true minority class.
2) Dataset balancing: SMOTE is a popular oversampling technique that improves classifiers' generalization ability by creating new samples rather than duplicating existing ones. X'_{min} is subjected to the SMOTE to get a more precise class balance. For each $x_i \in X'_{min}$, X_{sys} is computed by steps given in a Section II-A. The resulting balanced dataset (X_{bal}, y_{bal}) is represented as

$$X_{bal} = X_{sys} \cup X'_{min} \cup X'_{maj} \tag{8}$$

SMOTE smooths the decision boundary between classes and balances the class distributions. It ensures the subsequent classifier can more effectively capture the minority class regions without overfitting. The fully balanced dataset resulting from this oversampling step is now ready to be trained using the CSRBoost ensemble.

3) BOOSTING WITH ADABOOST

CSRBoost advances on to the AdaBoost ensemble learning stage to improve the performance of the balanced dataset (X_{bal}, y_{bal}) developed using undersampling and oversampling strategies. AdaBoost is a more powerful classifier that achieves higher accuracy since it can adjust sample weights based on misclassification mistakes. This process gradually tightens the decision boundary and ensures the X_{maj} does not dominate the classifier, which enhances the previous resampling steps.

Undersampling, oversampling, and boosting combine to form a complete ensemble framework that solves the CI problem. Combining these methods gives a more balanced and efficient model for unbalanced classification tasks by greatly improving the classifier's ability to generalize and make correct predictions on both classes. The cluster-aware resampling then boosting method preserves the local structure of X_{maj} and reduces the iterative tasks of removing similar informative samples in a controlled manner. The clustering size is decided by Eq. 6, and different cluster sizes have been investigated. Besides, single-shot resampling has been adopted to enhance computational efficiency since the proposed method is applied to different sample sizes and dimensionality levels of datasets. Consequently, AdaBoost has been employed to address the difficulty of classifying the samples.

The proposed method is summarized as:

- Ensemble learning framework: A potent ensemble learning framework that lessens CI bias and enhances predictive performance is produced by systematically combining undersampling and oversampling, followed by boosting.
- Complementary effects: RUS shrinks the amount of the dataset, while SMOTE interpolates instead of replicates, both of which help prevent overfitting. Each iteration enhances resilience against noise and outliers by focusing on misclassified cases.
- Dataset representatives: A simple approach to managing X_{maj} is demonstrated through the utilization of clustering approach in a controlled manner. It preserves the dataset representative and data locality while lowering computational complexity.
- Adaptive learning for X_{min}: AdaBoost dynamically updates weights to emphasize difficult X_{min} samples, allowing the model to adapt to complex boundaries. Consequently, it results in an enhanced generalization.

CSRBoost, which emphasizes building of balanced datasets through resampling techniques and employs boosting to enhance classification performance, especially for X_{min} occurrences, is a useful way to handle CI.

III. RESULTS AND DISCUSSIONS

Details regarding the experimental setup, datasets, and results of the experiment are given in this section.

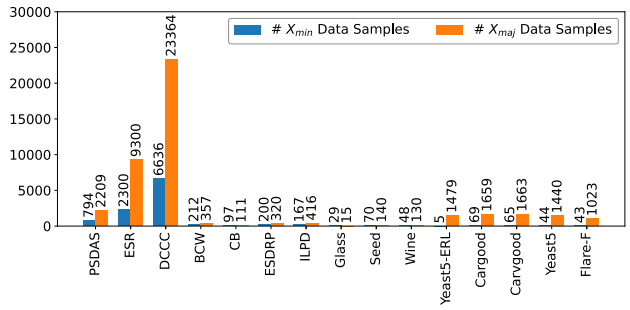


FIGURE 2. Class imbalance within datasets.

A. EXPERIMENTAL SETUP

A Google Colab[1] running an Intel Xeon CPU with two virtual CPUs and 13 GB of RAM was used to run the code. The CSRBoost approach has been implemented using the Python[2] programming language. Numerous Python libraries have been used, including matplotlib, NumPy, scikit-learn, and others. To guarantee balanced class distributions across folds, the dataset was partitioned using stratified K-Fold cross-validation [43]. Usually, a 5-fold Stratified K-Fold was employed, which divided the data into five sections, with one fold being used for testing and the other four for training. This procedure was carried out 20 times to record the average result. The SMOTE makes use of the five nearest neighbors. The base classifier is a decision tree [43].

Performance indicators such as G-Mean, average precision (AP) score, F1-score, and AUC-ROC (Area under the Receiver Operating Characteristic Curve) are adopted to evaluate the classifier's performance [43]. They are provided as

$$G-Mean=\sqrt{\frac{1}{1+\frac{FP}{TN}}\times\frac{1}{1+\frac{FN}{TP}}} \tag{9}$$

where true negatives (TN) means # instances are predicated correctly as "negative," false positives (FP) means # instances are predicted inaccurately as "positive," false negatives (FN) means # instances are predicted inaccurately as "negative," and true positives (TP) means instances are predicated correctly as "positive."

$$APscore=\sum_{\theta=0}^{\Theta-1}[RL_{TD}(\theta)-RL_{TD}(\theta+1)]\cdot PN_{TD}(\theta) \tag{10}$$

where precision $PN_{TD}(\theta)$ and recall $RL_{TD}(\theta)$ for threshold θ and they are computed for thresholds Θ - 1.

$$F1-score=\frac{2TP}{FP+FN+2TP} \tag{11}$$

[1]https://colab.research.google.com/
[2]https://www.python.org/

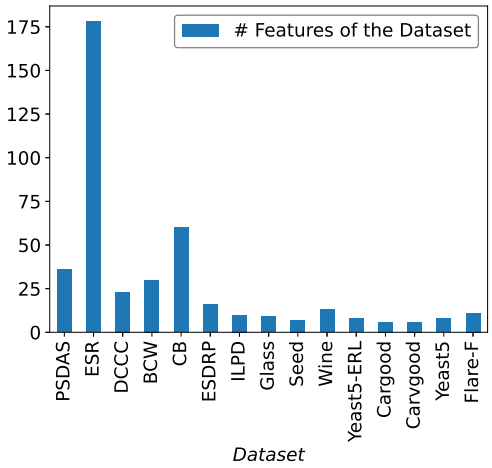


FIGURE 3. Feature distribution of datasets.

B. DATASET DESCRIPTION

Extensive tests on common benchmark CI datasets are necessary to validate the proposed methodology. The proposed methodology was evaluated on four different categories of CI datasets. These categories cover a broad spectrum of classification problems, such as different sample sizes and dimensionality levels, ensuring robust evaluation. The following are the details.

1) High Dimensionality - High Sample Size (HDHSS)
   - Predict Students' Dropout and Academic Success (PSDAS) [44]
   - Epileptic Seizure Recognition (ESR) [45]
   - Default of Credit Card Clients (DCCC) [46]
2) High Dimensionality - Low Sample Size (HDLSS)
   - Breast Cancer Wisconsin (Diagnostic) (BCW) [47]
   - Connectionist Bench (CB) [48]
   - Early Stage Diabetes Risk Prediction (ESDRP) [49]
3) Low Dimensionality - Low Sample Size (LDLSS)
   - Indian Liver Patient Dataset (ILPD) [50]
   - Glass [51]
   - Seed [52]
   - Wine [53]
4) Low Dimensionality - High Sample Size (LDHSS)
   - Yeast5-ERL [54]
   - Cargood [55]
   - Carvgood [55]
   - Yeast5 [54]
   - Flare-F [56]

The datasets are divided as: Real world (Flare-F, Yeast5, Yeast5-ERL, Wine, Seed, ILPD, ESDRP, BCW, DCCC, ESR, and PSDAS) and Synthetic (Carvgood, Cargood, Glass, and CB). Figs. 2 - 4 presents a comparative analysis of each dataset. With a primary focus on CI, it comprehensively studies several datasets across various criteria. The number of features can impact model complexity and performance, even though it differs between datasets and isn't connected to the

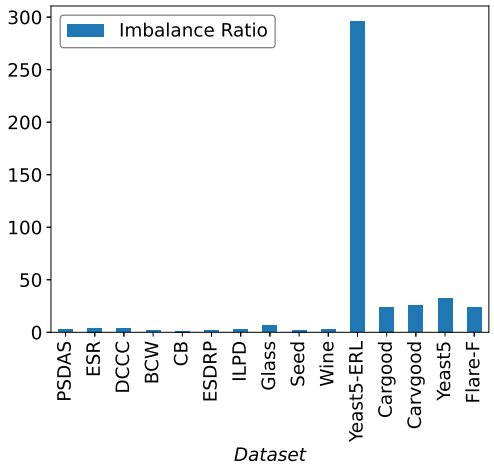


FIGURE 4. Imbalance ratio of various datasets.

CI problem. The datasets originated from various fields and are either collected in lab settings or in real-world scenarios. We transformed the datasets by determining which class was the most under-represented and designated it as the X_{min} and the remaining classes as X_{maj} in order to make binary classification easier. As seen in Fig. 4, the imbalance ratio uses the ratio of the $|X_{maj}|$ to the $|X_{min}|$ to quantify the level of imbalance. The dominance of X_{maj} is demonstrated by yeast5-ERL, which has an extremely high imbalance ratio, and it is 295.8. Moreover, because of their high imbalance ratios of 32.73 and 23.79, respectively, yeast5 and flare-F are challenging datasets for classification algorithms. The proposed CSRBoost method is assessed on a variety of domains with a range of characteristics and imbalance ratios of the datasets, as shown in Figs. 2 - 4. If the imbalance issue is not resolved, DM & ML models trained on the CI datasets may produce predictions that are noticeably biased in favor of the X_{maj}. As a result, performance metrics such as accuracy (ACC (%)) could be misleading.

C. PERFORMANCE ANALYSIS OF CSRBOOST ON CLUSTERING-BASED SAMPLING TECHNIQUE

The CSRBoost methods performance is assessed and presented in table 1 for a range of datasets under gradually decreasing clustering percentages from 100.0% to 10.0% based on the X_{min} samples. The goal is comprehending how classification performance is impacted when the $n_{clusters}$ is reduced. It assists in understanding the clustering-based sampling technique.

The HDHSS datasets show stable ACC with different % of clustering. The ACC of the PSDAS dataset remains consistently around 71.0%, showing minimal variation even at 10.0% clustering. This dataset is robust to subsampling, as evidenced by the stability of all metrics. High ACC (nearly 91.0%) and high AUC (nearly 0.89) were maintained across all clustering levels for the ESR dataset. The F1-score, AP score, and G-Mean show consistency, signifying strong model generalization. However, slight reductions in the F1-score and AP score were observed as the cluster size decreased. For the DCCC dataset, the performance is very low. All metrics achieved inferior values, indicating that data quality or separability is poor and not influenced by clustering density.

For the HDLSS dataset, the BCW and ESDRP datasets demonstrate decent performance for the proposed clustering-based sampling techniques at various levels. The BSW dataset achieves an ACC of approximately 94.0%, along with high and consistent metrics across all levels. Although there is a slight drop in ACC and AP score at lower percentages, it remains within an acceptable margin of noise. Clustering compression is a safe method, as seen by this consistent performance. The CB dataset, on the other hand, shows greater variation across parameters, especially in AP score and F1-score. Clustering affects performance, with variances ranging from 2.0 to 3.0%. Because of the CI, the dataset exhibits a considerable subsampling sensitivity. With an ACC of over 97.0% and an AUC of over 0.97, the ESDRP dataset performs quite well. However, when the clustering percentage drops, performance gradually deteriorates. In particular, the ACC decreases by roughly 4.0% from 100.0% to 10.0% clustering process, while the F1-score and G-Mean both exhibit modest decrease. The dataset does well overall but might benefit from more thorough clustering.

Most of the LDLSS datasets have achieved decent and stable performance. With an ACC of roughly 66.0% and an AUC of around 0.66, the ILPD dataset, on the other hand, performs poorly to moderately. The metrics are consistent but not very impressive, which indicates the presence of feature noise or label overlap, which does not appear to be impacted by clustering. In contrast, the glass dataset has good and consistent metrics, with an AUC of almost 0.90 and an ACC of roughly 95.5%. Performance deterioration is minimal, and mid-level clustering shows a small improvement. The clustering percentage does not impact this result significantly. The seed dataset performs exceptionally well on every metric. With standard deviations of roughly 0.01–0.02, the ACC, AUC, F1-score, AP score, and G-Mean are within the range of 96.0 – 97.0%, with minimal variation. Besides, the wine dataset exhibits great stability, with metrics like AUC and G-Mean continuously lingering near the top end, around 0.98. The performance does not significantly decline when clustering decreases, suggesting a robust feature-class relationship.

Similar to prior categories of datasets, marginal impact is observed on some LDHSS datasets. Clustering has a very strong effect on the yeast5-ERL dataset. There is considerable variation in other parameters; the ACC varies from 77.83% to 99.93%. At roughly 22.0%, the standard deviations are noticeably high. It implies thorough clustering is essential because the data may be brittle, noisy, or sparse. With an ACC of approximately 99.0%, an AUC of approximately 0.99, and an F1-score of approximately 0.96, the cargood and carvgood

TABLE 1. Comparison with clustering percentage based on X_{min} samples.

Dataset		Clustering percentage based on X_{min} samples									
		100.0%	90.0%	80.0%	70.0%	60.0%	50.0%	40.0%	30.0%	20.0%	10.0%
PSDAS	ACC (%)	72.85 ± 0.8	71.20 ± 0.01	71.04 ± 0.01	71.23 ± 0.01	70.60 ± 0.01	70.95 ± 0.01	71.06 ± 0.01	71.16 ± 0.01	71.22 ± 0.01	71.46 ± 0.01
	AUC	0.66 ± 0.01	0.64 ± 0.01	0.64 ± 0.01	0.65 ± 0.01	0.64 ± 0.01	0.64 ± 0.01	0.64 ± 0.01	0.64 ± 0.01	0.64 ± 0.01	0.64 ± 0.01
	F1-score	0.40 ± 0.01	0.40 ± 0.01	0.40 ± 0.01	0.40 ± 0.01	0.39 ± 0.01	0.40 ± 0.01	0.40 ± 0.01	0.40 ± 0.01	0.39 ± 0.01	0.40 ± 0.01
	AP score	0.25 ± 0.01	0.25 ± 0.01	0.26 ± 0.07	0.26 ± 0.01	0.25 ± 0.01	0.25 ± 0.01	0.25 ± 0.01	0.25 ± 0.01	0.25 ± 0.01	0.25 ± 0.01
	G-Mean	0.63 ± 0.01	0.63 ± 0.01	0.63 ± 0.01	0.64 ± 0.01	0.63 ± 0.01	0.63 ± 0.01	0.63 ± 0.01	0.63 ± 0.01	0.63 ± 0.01	0.63 ± 0.01
ESR	ACC (%)	92.05 ± 0.3	91.18 ± 0.00	91.21 ± 0.00	91.03 ± 0.00	90.84 ± 0.00	90.60 ± 0.00	90.46 ± 0.00	90.34 ± 0.00	90.16 ± 0.00	90.19 ± 0.00
	AUC	0.90 ± 0.00	0.89 ± 0.00	0.89 ± 0.00	0.89 ± 0.00	0.89 ± 0.00	0.89 ± 0.00	0.89 ± 0.00	0.89 ± 0.00	0.89 ± 0.00	0.89 ± 0.00
	F1-score	0.80 ± 0.01	0.79 ± 0.00	0.80 ± 0.00	0.79 ± 0.01	0.79 ± 0.01	0.78 ± 0.01	0.78 ± 0.01	0.78 ± 0.00	0.78 ± 0.01	0.78 ± 0.01
	AP score	0.67 ± 0.01	0.66 ± 0.01	0.66 ± 0.01	0.66 ± 0.01	0.65 ± 0.01	0.65 ± 0.01	0.65 ± 0.01	0.64 ± 0.01	0.64 ± 0.01	0.64 ± 0.01
	G-Mean	0.89 ± 0.00	0.89 ± 0.00	0.89 ± 0.00	0.89 ± 0.00	0.89 ± 0.00	0.89 ± 0.00	0.89 ± 0.00	0.89 ± 0.00	0.89 ± 0.00	0.89 ± 0.00
DCCC	ACC (%)	68.32 ± 0.26	67.51 ± 0.00	67.39 ± 0.00	67.34 ± 0.01	67.23 ± 0.00	67.09 ± 0.00	66.92 ± 0.00	66.98 ± 0.00	66.94 ± 0.00	67.02 ± 0.00
	AUC	0.64 ± 0.00	0.63 ± 0.00	0.63 ± 0.00	0.63 ± 0.01	0.63 ± 0.00	0.62 ± 0.00	0.62 ± 0.00	0.62 ± 0.00	0.62 ± 0.00	0.62 ± 0.00
	F1-score	0.42 ± 0.00	0.42 ± 0.00	0.42 ± 0.00	0.42 ± 0.01	0.42 ± 0.00	0.42 ± 0.00	0.42 ± 0.00	0.42 ± 0.00	0.42 ± 0.00	0.42 ± 0.00
	AP score	0.29 ± 0.00	0.29 ± 0.00	0.29 ± 0.00	0.29 ± 0.01	0.29 ± 0.00	0.29 ± 0.00	0.29 ± 0.00	0.29 ± 0.00	0.29 ± 0.00	0.29 ± 0.00
	G-Mean	0.62 ± 0.00	0.62 ± 0.00	0.62 ± 0.00	0.62 ± 0.01	0.62 ± 0.00	0.62 ± 0.00	0.62 ± 0.00	0.62 ± 0.00	0.62 ± 0.00	0.62 ± 0.00
BCW	ACC (%)	94.37 ± 1.11	92.07 ± 1.00	92.37 ± 0.83	92.14 ± 0.83	91.93 ± 1.06	91.88 ± 0.99	91.52 ± 0.77	91.40 ± 0.98	91.27 ± 0.80	91.44 ± 0.92
	AUC	0.94 ± 0.01	0.92 ± 0.01	0.92 ± 0.01	0.92 ± 0.01	0.92 ± 0.01	0.92 ± 0.01	0.92 ± 0.01	0.92 ± 0.01	0.91 ± 0.01	0.92 ± 0.01
	F1-score	0.90 ± 0.01	0.90 ± 0.01	0.90 ± 0.01	0.90 ± 0.01	0.90 ± 0.01	0.89 ± 0.01	0.89 ± 0.01	0.89 ± 0.01	0.89 ± 0.01	0.89 ± 0.01
	AP score	0.84 ± 0.02	0.84 ± 0.02	0.84 ± 0.02	0.84 ± 0.02	0.83 ± 0.02	0.83 ± 0.02	0.82 ± 0.01	0.82 ± 0.02	0.82 ± 0.02	0.82 ± 0.02
	G-Mean	0.92 ± 0.01	0.92 ± 0.01	0.92 ± 0.01	0.92 ± 0.01	0.92 ± 0.01	0.92 ± 0.01	0.92 ± 0.01	0.92 ± 0.01	0.91 ± 0.01	0.92 ± 0.01
CB	ACC (%)	76.43 ± 2.43	69.90 ± 2.35	71.30 ± 2.62	68.89 ± 0.65	71.40 ± 3.27	70.55 ± 3.19	70.79 ± 2.72	69.36 ± 2.64	69.34 ± 3.42	69.36 ± 2.82
	AUC	0.76 ± 0.03	0.70 ± 0.02	0.71 ± 0.03	0.69 ± 0.01	0.71 ± 0.03	0.71 ± 0.03	0.71 ± 0.03	0.70 ± 0.03	0.70 ± 0.03	0.70 ± 0.03
	F1-score	0.69 ± 0.03	0.68 ± 0.03	0.70 ± 0.03	0.67 ± 0.01	0.70 ± 0.03	0.70 ± 0.03	0.71 ± 0.03	0.70 ± 0.02	0.70 ± 0.03	0.70 ± 0.03
	AP score	0.63 ± 0.02	0.62 ± 0.02	0.63 ± 0.03	0.61 ± 0.01	0.63 ± 0.03	0.62 ± 0.03	0.62 ± 0.02	0.61 ± 0.02	0.61 ± 0.03	0.61 ± 0.02
	G-Mean	0.71 ± 0.03	0.69 ± 0.03	0.71 ± 0.03	0.68 ± 0.01	0.71 ± 0.03	0.70 ± 0.03	0.71 ± 0.03	0.69 ± 0.03	0.69 ± 0.03	0.69 ± 0.03
ESDRP	ACC (%)	97.31 ± 0.85	96.08 ± 0.54	95.75 ± 0.72	96.08 ± 0.65	95.61 ± 0.66	95.07 ± 0.85	94.50 ± 1.07	94.39 ± 0.81	94.56 ± 1.11	93.83 ± 0.86
	AUC	0.97 ± 0.01	0.96 ± 0.01	0.96 ± 0.01	0.96 ± 0.00	0.96 ± 0.01	0.95 ± 0.01	0.95 ± 0.01	0.95 ± 0.01	0.95 ± 0.01	0.94 ± 0.01
	F1-score	0.95 ± 0.01	0.95 ± 0.01	0.95 ± 0.01	0.95 ± 0.01	0.94 ± 0.01	0.94 ± 0.01	0.93 ± 0.01	0.93 ± 0.01	0.93 ± 0.01	0.92 ± 0.01
	AP score	0.92 ± 0.02	0.92 ± 0.01	0.91 ± 0.01	0.92 ± 0.01	0.91 ± 0.01	0.90 ± 0.02	0.88 ± 0.02	0.88 ± 0.02	0.88 ± 0.02	0.87 ± 0.02
	G-Mean	0.96 ± 0.01	0.96 ± 0.01	0.96 ± 0.01	0.96 ± 0.01	0.96 ± 0.01	0.95 ± 0.01	0.95 ± 0.01	0.95 ± 0.01	0.95 ± 0.01	0.94 ± 0.01
ILPD	ACC (%)	66.72 ± 2.06	63.56 ± 2.00	62.89 ± 1.77	63.92 ± 0.65	63.31 ± 1.87	63.01 ± 1.20	63.04 ± 2.17	63.05 ± 1.83	63.30 ± 1.82	63.47 ± 1.78
	AUC	0.66 ± 0.01	0.62 ± 0.02	0.62 ± 0.02	0.63 ± 0.01	0.62 ± 0.02	0.62 ± 0.02	0.61 ± 0.02	0.61 ± 0.02	0.62 ± 0.02	0.62 ± 0.02
	F1-score	0.48 ± 0.03	0.48 ± 0.03	0.48 ± 0.03	0.49 ± 0.01	0.48 ± 0.02	0.47 ± 0.02	0.47 ± 0.03	0.47 ± 0.03	0.48 ± 0.02	0.48 ± 0.02
	AP score	0.36 ± 0.02	0.36 ± 0.02	0.36 ± 0.02	0.36 ± 0.01	0.36 ± 0.02	0.36 ± 0.01	0.35 ± 0.02	0.36 ± 0.02	0.36 ± 0.01	0.36 ± 0.02
	G-Mean	0.62 ± 0.02	0.62 ± 0.02	0.61 ± 0.02	0.62 ± 0.01	0.61 ± 0.02	0.61 ± 0.02	0.61 ± 0.02	0.61 ± 0.02	0.61 ± 0.02	0.62 ± 0.02
Glass	ACC (%)	95.8 ± 1.23	93.53 ± 1.05	93.23 ± 1.28	93.88 ± 0.65	93.86 ± 1.18	93.41 ± 0.94	93.53 ± 1.66	93.27 ± 1.52	93.57 ± 1.11	93.46 ± 1.11
	AUC	0.93 ± 0.01	0.90 ± 0.02	0.90 ± 0.02	0.90 ± 0.01	0.90 ± 0.01	0.90 ± 0.02	0.91 ± 0.02	0.90 ± 0.02	0.90 ± 0.02	0.91 ± 0.02
	F1-score	0.79 ± 0.03	0.78 ± 0.03	0.78 ± 0.03	0.79 ± 0.01	0.79 ± 0.03	0.77 ± 0.03	0.79 ± 0.05	0.78 ± 0.04	0.79 ± 0.03	0.79 ± 0.03
	AP score	0.67 ± 0.05	0.66 ± 0.04	0.65 ± 0.05	0.67 ± 0.01	0.67 ± 0.05	0.65 ± 0.04	0.67 ± 0.07	0.65 ± 0.06	0.66 ± 0.04	0.66 ± 0.04
	G-Mean	0.90 ± 0.02	0.90 ± 0.02	0.90 ± 0.02	0.90 ± 0.01	0.90 ± 0.02	0.89 ± 0.02	0.90 ± 0.02	0.89 ± 0.03	0.90 ± 0.02	0.90 ± 0.02
Seed	ACC (%)	98.1 ± 0.72	97.07 ± 0.98	97.40 ± 0.71	96.93 ± 0.65	96.57 ± 1.24	96.69 ± 0.73	96.05 ± 1.50	96.10 ± 0.96	96.19 ± 0.92	95.81 ± 0.92
	AUC	0.98 ± 0.07	0.97 ± 0.01	0.97 ± 0.01	0.97 ± 0.01	0.96 ± 0.01	0.97 ± 0.01	0.96 ± 0.01	0.96 ± 0.01	0.96 ± 0.01	0.96 ± 0.01
	F1-score	0.96 ± 0.01	0.96 ± 0.01	0.96 ± 0.01	0.96 ± 0.01	0.95 ± 0.02	0.95 ± 0.01	0.94 ± 0.02	0.94 ± 0.01	0.94 ± 0.01	0.94 ± 0.01
	AP score	0.93 ± 0.02	0.93 ± 0.02	0.94 ± 0.02	0.93 ± 0.01	0.92 ± 0.03	0.92 ± 0.02	0.91 ± 0.03	0.91 ± 0.02	0.91 ± 0.02	0.90 ± 0.02
	G-Mean	0.97 ± 0.01	0.97 ± 0.01	0.97 ± 0.01	0.97 ± 0.01	0.96 ± 0.01	0.97 ± 0.01	0.96 ± 0.01	0.96 ± 0.01	0.96 ± 0.01	0.96 ± 0.01
Wine	ACC (%)	98.86 ± 1.31	96.01 ± 1.50	96.32 ± 1.36	95.64 ± 0.65	95.56 ± 1.72	95.79 ± 1.50	95.68 ± 1.61	96.23 ± 1.58	96.54 ± 1.21	96.19 ± 1.26
	AUC	0.98 ± 0.01	0.96 ± 0.02	0.96 ± 0.02	0.96 ± 0.01	0.95 ± 0.02	0.95 ± 0.01	0.96 ± 0.02	0.96 ± 0.02	0.96 ± 0.02	0.96 ± 0.01
	F1-score	0.93 ± 0.02	0.93 ± 0.03	0.93 ± 0.02	0.93 ± 0.01	0.92 ± 0.03	0.93 ± 0.02	0.92 ± 0.03	0.93 ± 0.03	0.94 ± 0.02	0.93 ± 0.02
	AP score	0.88 ± 0.04	0.88 ± 0.04	0.89 ± 0.04	0.89 ± 0.01	0.87 ± 0.04	0.88 ± 0.04	0.88 ± 0.04	0.89 ± 0.04	0.90 ± 0.03	0.89 ± 0.03
	G-Mean	0.95 ± 0.02	0.96 ± 0.02	0.96 ± 0.02	0.96 ± 0.01	0.95 ± 0.02	0.95 ± 0.01	0.95 ± 0.02	0.96 ± 0.02	0.96 ± 0.02	0.96 ± 0.02
Yeast5-ERL	ACC (%)	99.93 ± 0.27	99.93 ± 0.26	99.88 ± 26.80	98.85 ± 0.18	90.86 ± 0.25	87.88 ± 0.30	79.86 ± 0.26	73.86 ± 0.29	83.81 ± 20.57	77.83 ± 0.19
	AUC	0.60 ± 0.31	0.71 ± 0.31	0.77 ± 0.30	0.78 ± 0.20	0.72 ± 0.30	0.68 ± 0.35	0.66 ± 0.31	0.71 ± 0.33	0.82 ± 0.24	0.75 ± 0.24
	F1-score	0.47 ± 0.31	0.57 ± 0.31	0.67 ± 0.29	0.66 ± 0.20	0.59 ± 0.31	0.57 ± 0.35	0.53 ± 0.32	0.59 ± 0.33	0.66 ± 0.25	0.61 ± 0.25
	AP score	0.43 ± 0.29	0.53 ± 0.29	0.63 ± 0.28	0.61 ± 0.19	0.54 ± 0.30	0.53 ± 0.33	0.48 ± 0.30	0.54 ± 0.31	0.60 ± 0.23	0.57 ± 0.24
	G-Mean	0.55 ± 0.35	0.68 ± 0.36	0.75 ± 0.33	0.77 ± 0.22	0.69 ± 0.36	0.65 ± 0.39	0.62 ± 0.37	0.69 ± 0.38	0.80 ± 0.28	0.73 ± 0.28
Cargood	ACC (%)	98.9 ± 0.30	98.18 ± 0.43	98.34 ± 0.30	98.20 ± 0.35	98.41 ± 0.39	98.40 ± 0.37	98.30 ± 0.42	98.20 ± 0.42	98.17 ± 0.36	98.13 ± 0.49
	AUC	0.99 ± 0.02	0.93 ± 0.03	0.93 ± 0.02	0.92 ± 0.02	0.93 ± 0.03	0.94 ± 0.02	0.93 ± 0.02	0.93 ± 0.02	0.92 ± 0.02	0.93 ± 0.02
	F1-score	0.80 ± 0.03	0.80 ± 0.05	0.81 ± 0.03	0.79 ± 0.04	0.82 ± 0.04	0.82 ± 0.04	0.81 ± 0.05	0.80 ± 0.04	0.79 ± 0.04	0.79 ± 0.05
	AP score	0.66 ± 0.04	0.66 ± 0.07	0.68 ± 0.05	0.66 ± 0.06	0.69 ± 0.07	0.70 ± 0.06	0.68 ± 0.07	0.66 ± 0.07	0.64 ± 0.06	0.66 ± 0.08
	G-Mean	0.92 ± 0.02	0.92 ± 0.03	0.92 ± 0.02	0.92 ± 0.03	0.93 ± 0.03	0.94 ± 0.02	0.93 ± 0.02	0.92 ± 0.02	0.91 ± 0.03	0.92 ± 0.03
Carvgood	ACC (%)	99.83 ± 0.12	99.65 ± 0.13	99.67 ± 0.16	99.69 ± 0.13	99.69 ± 0.16	99.60 ± 0.17	99.68 ± 0.13	99.68 ± 0.15	99.68 ± 0.14	99.69 ± 0.10
	AUC	0.99 ± 0.01	0.99 ± 0.01	0.99 ± 0.01	0.99 ± 0.01	0.99 ± 0.01	0.98 ± 0.02	0.99 ± 0.01	0.99 ± 0.01	0.99 ± 0.01	0.98 ± 0.01
	F1-score	0.96 ± 0.02	0.96 ± 0.02	0.96 ± 0.02	0.96 ± 0.02	0.96 ± 0.02	0.95 ± 0.02	0.96 ± 0.02	0.96 ± 0.02	0.96 ± 0.02	0.96 ± 0.01
	AP score	0.92 ± 0.03	0.92 ± 0.03	0.92 ± 0.03	0.93 ± 0.03	0.93 ± 0.03	0.91 ± 0.04	0.92 ± 0.03	0.93 ± 0.03	0.92 ± 0.03	0.92 ± 0.02
	G-Mean	0.99 ± 0.01	0.99 ± 0.01	0.99 ± 0.01	0.99 ± 0.01	0.99 ± 0.01	0.98 ± 0.02	0.98 ± 0.01	0.99 ± 0.01	0.99 ± 0.01	0.98 ± 0.01
Yeast5	ACC (%)	98.32 ± 0.24	97.73 ± 0.23	97.68 ± 0.29	97.61 ± 0.31	97.73 ± 0.28	97.68 ± 0.33	97.63 ± 0.23	97.67 ± 0.23	97.74 ± 0.26	97.67 ± 0.25
	AUC	0.93 ± 0.00	0.89 ± 0.02	0.88 ± 0.03	0.88 ± 0.03	0.89 ± 0.03	0.90 ± 0.03	0.88 ± 0.02	0.89 ± 0.03	0.89 ± 0.02	0.89 ± 0.03
	F1-score	0.69 ± 0.03	0.68 ± 0.03	0.67 ± 0.04	0.66 ± 0.04	0.68 ± 0.04	0.68 ± 0.04	0.66 ± 0.03	0.67 ± 0.04	0.68 ± 0.03	0.67 ± 0.03
	AP score	0.50 ± 0.04	0.50 ± 0.04	0.48 ± 0.05	0.47 ± 0.05	0.49 ± 0.05	0.50 ± 0.06	0.47 ± 0.04	0.49 ± 0.05	0.50 ± 0.03	0.48 ± 0.05
	G-Mean	0.89 ± 0.02	0.89 ± 0.03	0.87 ± 0.03	0.87 ± 0.03	0.89 ± 0.03	0.89 ± 0.03	0.87 ± 0.03	0.88 ± 0.03	0.89 ± 0.02	0.88 ± 0.03
Flare-F	ACC (%)	93.43 ± 0.65	92.50 ± 0.64	92.54 ± 0.77	92.34 ± 0.74	92.46 ± 0.84	92.41 ± 0.73	92.30 ± 0.72	92.37 ± 0.86	92.44 ± 0.72	92.45 ± 0.50
	AUC	0.67 ± 0.01	0.64 ± 0.03	0.65 ± 0.04	0.65 ± 0.03	0.64 ± 0.03	0.65 ± 0.03	0.64 ± 0.03	0.64 ± 0.03	0.64 ± 0.05	0.65 ± 0.03
	F1-score	0.22 ± 0.05	0.27 ± 0.04	0.27 ± 0.05	0.27 ± 0.04	0.26 ± 0.04	0.27 ± 0.04	0.25 ± 0.04	0.27 ± 0.05	0.26 ± 0.07	0.27 ± 0.04
	AP score	0.10 ± 0.02	0.12 ± 0.02	0.12 ± 0.03	0.12 ± 0.02	0.11 ± 0.02	0.12 ± 0.02	0.11 ± 0.02	0.12 ± 0.03	0.11 ± 0.03	0.12 ± 0.02
	G-Mean	0.48 ± 0.08	0.55 ± 0.05	0.55 ± 0.07	0.56 ± 0.06	0.54 ± 0.06	0.56 ± 0.05	0.53 ± 0.07	0.55 ± 0.06	0.53 ± 0.10	0.56 ± 0.06

datasets demonstrate near-perfect results in which the model is well-fitted and highly confident, with metrics remaining flat and stable even at a 10.0% threshold, where full clustering provides only minimal advantages. Performance on the yeast5 dataset is consistent and reasonably good. The AUC and F1-score somewhat decrease at lower clustering levels, but the findings are still within a reasonable range. Besides, it performs better than its yeast5-ERL version and shows good resistance. Conversely, the flare-F dataset shows low metric scores, with an AP score of 0.10 to 0.12 and F1-score of roughly 0.22 to 0.27. The AUC is roughly 0.65, and the ACC is 92.0%. It suggests a lack of useful features or label imbalance. These performance limitations are not addressed by increasing clustering.

Small standard deviations are seen in most datasets, suggesting that the model performs consistently. Across all clustering percentages, the datasets carvgood, cargood, ESR, seed, ESDRP, and wine achieve constant AUC $\geq$ 0.95 and high ACC $>$ 95.0%. In certain situations, such as PSDAS, where the ACC rises, and the AUC of Yeast5-ERL rises, performance is marginally improved by lowering clustering. AUC and G-Mean exhibit the most consistency across all clustering levels for most datasets. Even for sensitive datasets, the G-Mean stays within a small range. In robust datasets such as ESR, wine, and seed, AUC variances are nearly insignificant. This thorough analysis shows that lowering clustering percentages can frequently be done safely across a wide range of datasets without sacrificing the efficacy of the model. However, the datasets' properties are particularly important: datasets with intrinsic structure (e.g.: carvgood, seed) are generally more stable, while noisy or unbalanced datasets (e.g.: flare-F, yeast5-ERL) might need full clustering for best results.

D. COMPARISON WITH VARIOUS STATE-OF-THE-ART TECHNIQUES

CSRBoost, SMOTified-GAN [32], GAN [32], ADASYN [57], Borderline-SMOTE [58], SMOTE-edited nearest neighbor (SMOTE-ENN) [38], SMOTE-Tomek [38], [59], Adaboost [25], RUSBoost [37], and hash-based undersampling ensemble (HUE) [60] are all thoroughly compared in the table 2. Time series datasets such as DCCC, ESR, and PSDAS are unsuitable for RUSBoost since they lose a lot of information because most classes are reduced [61]. Furthermore, because none of the neighbors in ADASYN, GAN, and SMOTified-GAN belong to the majority class, they are inappropriate for the Yeast5-ERL and CB datasets. The configuration described in [32] is employed for both GAN and SMOTified-GAN in order to generate results.

All the state-of-the-art methods achieve AUC $>$ 0.90 and ACC $>$ 90.0% on most datasets. Both SMOTified-GAN and GAN show a decent performance in terms of AUC $>$ 0.95 and ACC $>$ 90.0% for CI datasets such as ESR, BCW, ESDRP, wine, cargood, carvgood, yeast5, and flare-F. However, compared to traditional approaches, SMOTified-GAN and GAN have higher processing requirements. With a 72.85% ACC and a G-Mean of 0.63, CSRBoost performs admirably on most measures. Particularly when contrasted with alternative techniques like SMOTified-GAN (ACC: 64.46%) and GAN (ACC: 63.80%) for the PSDAS dataset. Although CSRBoost performs well in terms of ACC, for the ESR dataset, SMOTified-GAN and GAN outperform it. Comparing CSRBoost to SMOTified-GAN and GAN reveals that it maintains a competitive G-Mean, indicating that it better balances and regulates the minority class. CSRBoost performs well in terms of ACC, attaining 94.37% for the BCW dataset. It is similar to SMOTified-GAN (94.17%) and SMOTE-Tomek (94.38%). Furthermore, SMOTified-GAN displays a comparable result of 0.94, although CSRBoost displays a high G-Mean of 0.92. Among the top three methods for the ESDRP dataset is CSRBoost. At 98.65%, RUSBoost attains the best accuracy. CSRBoost exhibits remarkably low volatility and consistently performs well across all five metrics, indicating consistent and broadly applicable performance. Additionally, despite significantly trailing in AP score, Borderline-SMOTE, SMOTE-Tomek, and SMOTE-ENN produce competitive outcomes, especially in terms of G-Mean and F1-scores.

ADASYN, Borderline-SMOTE, SMOTE-Tomek, and Adaboost are high-performance methods for the ILDP dataset that produce F1-scores of 0.97–0.98, AP scores of up to 0.97, and G-Mean values of 0.96–0.99. CSRBoost performs steadily and moderately on all criteria. CSRBoost ranks among the top performers for the glass dataset, demonstrating its fierce competition. In minority-specific measures, it lags somewhat behind specialized resampling techniques like SMOTE-ENN, but it performs noticeably better than GAN-based methods. CSRBoost continuously exhibits strong performance across all criteria for seed and wine datasets. In every category, it is in the top three and is more stable and interpretable than adversarial techniques. For Yeast5-ERL, every approach except CSRBoost achieves almost perfect ACC and AUC. Similarly, using ACC and AUC measures, cargood, carvgood, yeast5, flare-F dataset perform almost flawlessly.

CSRBoost demonstrates robust and consistent performance across a wide spectrum of class-imbalanced datasets, especially excelling in structured environments with moderate imbalance and well-separated feature spaces. Its hybrid framework proves highly effective for maintaining accuracy and generalization. In summary, CSRBoost is a strong, versatile method for class imbalance problems, particularly when data is moderately imbalanced and well-structured.

E. ABLATION ANALYSIS

Table 3 presents the ablation analysis comparing CSRBoost, SMOTEBoost, and RUSBoost's classification performance on several datasets. With remarkable results on various datasets, CSRBoost usually balances AUC and ACC well. Better generalization and efficacy in identifying minority classes are indicated by its capacity to retain a higher AUC

TABLE 2. Comparison with various state-of-the-art techniques.

Dataset		CSRBoost	SMOTified-GAN	GAN	ADASYN	Borderline-SMOTE	SMOTE-Tomek	SMOTE-ENN	Adaboost	RUSBoost	HUE
	ACC (%)	72.85±0.8	64.46±9.93	63.80±6.81	76.11±0.50	75.99±0.62	75.99±0.49	72.67±0.71	75.72±0.47	N.A.	74.25±0.50
	AUC	0.66±0.01	0.82±0.04	0.82±0.03	0.64±0.01	0.62±0.01	0.61±0.01	0.65±0.01	0.65±0.01	N.A.	0.71±0.01
PSDAS	F1-score	0.40±0.01	0.64±0.1	0.64±0.07	0.38±0.01	0.38±0.01	0.39±0.01	0.42±0.01	0.38±0.01	N.A.	0.49±0.01
	AP score	0.25±0.01	0.70±0.11	0.69±0.07	0.25±0.00	0.25±0.01	0.26±0.01	0.27±0.01	0.25±0.01	N.A.	0.32±0.01
	G-Mean	0.63±0.01	0.32±0.21	0.32±0.21	0.58±0.01	0.59±0.01	0.59±0.01	0.64±0.01	0.58±0.01	N.A.	0.74±0.01
	ACC (%)	92.05±0.3	95.84±0.49	95.89±0.54	89.95±0.27	91.93±0.22	91.81±0.21	91.72±0.36	94.18±0.20	N.A.	95.55±0.18
	AUC	0.90±0.00	0.97±0.01	0.97±0.01	0.89±0.01	0.89±0.01	0.89±0.01	0.89±0.01	0.91±0.01	N.A.	0.94±0.01
ESR	F1-score	0.80±0.01	0.96±0.00	0.96±0.01	0.77±0.01	0.82±0.00	0.81±0.00	0.81±0.01	0.85±0.01	N.A.	0.87±0.01
	AP score	0.67±0.01	0.95±0.01	0.95±0.01	0.63±0.01	0.69±0.01	0.69±0.01	0.68±0.01	0.76±0.01	N.A.	0.77±0.01
	G-Mean	0.89±0.00	0.92±0.01	0.92±0.02	0.89±0.00	0.90±0.00	0.90±0.00	0.90±0.01	0.91±0.00	N.A.	0.96±0.01
	ACC (%)	68.32±0.26	81.68±2.12	80.89±2.47	72.27±0.23	72.45±0.27	72.49±0.25	70.52±0.22	72.99±0.16	N.A.	68.66±0.20
	AUC	0.64±0.00	0.95±0.01	0.94±0.01	0.62±0.01	0.62±0.01	0.62±0.01	0.65±0.01	0.62±0.02	N.A.	0.68±0.01
DCCC	F1-score	0.42±0.00	0.82±0.00	0.81±0.00	0.41±0.00	0.41±0.00	0.41±0.00	0.45±0.00	0.41±0.00	N.A.	0.49±0.00
	AP score	0.29±0.00	0.88±0.00	0.87±0.00	0.29±0.00	0.29±0.00	0.29±0.00	0.31±0.00	0.30±0.00	N.A.	0.33±0.00
	G-Mean	0.62±0.00	0.82±0.00	0.81±0.00	0.59±0.00	0.59±0.00	0.59±0.00	0.64±0.00	0.59±0.00	N.A.	0.69±0.00
	ACC (%)	94.37±1.11	94.17±2.72	89.78±6.45	94.38±1.02	94.03±0.76	94.38±0.99	94.38±0.77	94.03±0.81	97.19±0.47	96.11±0.74
	AUC	0.94±0.01	0.99±0.01	0.99±0.01	0.94±0.02	0.94±0.01	0.94±0.01	0.94±0.01	0.94±0.01	0.97±0.01	0.96±0.01
BCW	F1-score	0.90±0.01	0.94±0.02	0.90±0.01	0.93±0.01	0.92±0.01	0.93±0.01	0.92±0.01	0.93±0.01	0.96±0.01	0.94±0.01
	AP score	0.84±0.02	0.99±0.00	0.99±0.01	0.88±0.02	0.87±0.02	0.90±0.02	0.88±0.01	0.88±0.02	0.75±0.01	0.89±0.02
	G-Mean	0.92±0.01	0.94±0.03	0.90±0.03	0.94±0.01	0.93±0.01	0.94±0.01	0.93±0.01	0.94±0.01	0.97±0.01	0.95±0.01
	ACC (%)	76.43±2.43	87.62±3.67	86.79±2.50	N.A.	77.44±2.57	78.37±1.95	75.01±2.16	76.42±1.99	82.26±2.09	78.81±2.37
	AUC	0.76±0.03	0.96±0.01	0.96±0.01	N.A.	0.77±0.03	0.78±0.02	0.74±0.02	0.76±0.02	0.72±0.02	0.78±0.03
CB	F1-score	0.69±0.03	0.88±0.04	0.87±0.02	N.A.	0.77±0.03	0.74±0.03	0.73±0.02	0.74±0.02	0.80±0.03	0.80±0.02
	AP score	0.63±0.02	0.93±0.01	0.94±0.01	N.A.	0.71±0.02	0.67±0.02	0.68±0.02	0.66±0.02	0.97±0.02	0.73±0.03
	G-Mean	0.71±0.03	0.88±0.04	0.86±0.04	N.A.	0.78±0.03	0.75±0.02	0.75±0.02	0.75±0.02	0.82±0.02	0.81±0.03
	ACC (%)	97.31±0.85	92.98±1.09	91.97±2.00	97.69±0.63	97.12±0.72	97.50±0.68	93.27±1.19	97.50±0.59	98.65±0.50	97.19±0.79
	AUC	0.97±0.01	0.99±0.01	0.98±0.01	0.98±0.01	0.97±0.01	0.97±0.01	0.94±0.01	0.97±0.01	0.99±0.01	0.97±0.01
ESDRP	F1-score	0.95±0.01	0.93±0.01	0.92±0.02	0.97±0.01	0.97±0.01	0.97±0.01	0.93±0.01	0.97±0.01	0.98±0.01	0.97±0.01
	AP score	0.92±0.02	0.99±0.00	0.99±0.01	0.95±0.01	0.95±0.02	0.95±0.01	0.88±0.02	0.95±0.01	0.94±0.01	0.94±0.01
	G-Mean	0.96±0.01	0.92±0.02	0.91±0.03	0.98±0.01	0.98±0.01	0.98±0.01	0.95±0.01	0.97±0.01	0.98±0.00	0.98±0.01
	ACC (%)	66.72±2.06	70.09±9.99	67.52±11.17	67.76±1.79	69.48±1.72	69.12±1.40	68.63±1.62	67.92±1.89	71.52±0.93	66.55±1.42
	AUC	0.66±0.01	0.76±0.05	0.74±0.05	0.64±0.03	0.65±0.02	0.63±0.02	0.69±0.02	0.62±0.03	0.66±0.02	0.71±0.02
ILPD	F1-score	0.48±0.03	0.77±0.10	0.74±0.11	0.97±0.03	0.98±0.03	0.97±0.02	0.95±0.02	0.98±0.04	0.53±0.02	0.58±0.02
	AP score	0.36±0.02	0.88±0.07	0.86±0.07	0.95±0.02	0.97±0.01	0.94±0.01	0.92±0.02	0.96±0.02	0.41±0.01	0.42±0.02
	G-Mean	0.62±0.02	0.55±0.23	0.54±0.23	0.97±0.02	0.98±0.03	0.99±0.02	0.96±0.02	0.99±0.03	0.66±0.02	0.70±0.02
	ACC (%)	95.8±1.23	55.47±6.84	56.74±7.13	96.74±1.01	96.28±1.14	96.74±0.74	96.27±0.95	99.53±0.44	97.21±1.09	94.88±0.94
	AUC	0.93±0.01	0.89±0.02	0.89±0.02	0.94±0.01	0.93±0.02	0.93±0.02	0.94±0.02	0.99±0.01	0.94±0.02	0.95±0.01
Glass	F1-score	0.79±0.03	0.55±0.07	0.56±0.07	0.96±0.03	0.96±0.03	0.97±0.03	0.98±0.03	0.90±0.19	0.91±0.03	0.82±0.02
	AP score	0.67±0.05	0.65±0.05	0.65±0.05	0.95±0.05	0.93±0.05	0.96±0.04	0.97±0.04	0.86±0.17	0.86±0.05	0.71±0.03
	G-Mean	0.90±0.02	0.88±0.01	0.88±0.01	0.97±0.02	0.97±0.01	0.97±0.03	0.98±0.02	0.92±0.21	0.94±0.02	0.93±0.01
	ACC (%)	98.1±0.72	86.31±3.85	87.14±2.93	98.1±1.35	98.57±1.02	98.57±1.05	98.57±1.26	90.48±1.80	98.10±0.65	98.57±0.58
	AUC	0.98±0.03	0.98±0.01	0.98±0.01	0.98±0.02	0.98±0.01	0.98±0.02	0.98±0.02	0.89±0.02	0.98±0.01	0.98±0.01
Seed	F1-score	0.96±0.01	0.86±0.03	0.87±0.03	0.50±0.02	0.49±0.01	0.51±0.02	0.57±0.02	0.49±0.03	0.98±0.01	0.98±0.01
	AP score	0.93±0.02	0.96±0.01	0.96±0.01	0.38±0.03	0.38±0.02	0.39±0.02	0.43±0.03	0.38±0.04	0.97±0.02	0.97±0.02
	G-Mean	0.97±0.01	0.85±0.04	0.86±0.05	0.63±0.01	0.62±0.01	0.64±0.01	0.70±0.01	0.62±0.02	0.99±0.01	0.99±0.01
	ACC (%)	98.86±1.31	96.67±1.9	93.89±1.60	98.89±1.07	98.89±1.02	98.3±0.84	97.21±1.51	98.89±1.03	99.43±0.96	94.43±1.22
	AUC	0.98±0.01	0.97±0.03	0.96±0.04	0.98±0.02	0.98±0.02	0.98±0.02	0.97±0.02	0.98±0.02	0.99±0.01	0.98±0.01
Wine	F1-score	0.93±0.02	0.77±0.12	0.74±0.16	0.87±0.02	0.88±0.02	0.88±0.02	0.88±0.03	0.87±0.03	0.99±0.02	0.97±0.01
	AP score	0.88±0.04	0.88±0.09	0.85±0.12	0.78±0.03	0.80±0.03	0.81±0.02	0.80±0.04	0.80±0.04	0.99±0.03	0.95±0.02
	G-Mean	0.95±0.02	0.55±0.38	0.54±0.35	0.94±0.02	0.94±0.02	0.92±0.01	0.93±0.02	0.94±0.02	0.99±0.01	0.98±0.01
	ACC (%)	99.93±0.27	N.A.	N.A.	N.A.	99.93±0.02	99.8±0.02	99.93±0.06	99.93±0.02	99.87±0.04	99.46±0.46
	AUC	0.60±0.31	N.A.	N.A.	N.A.	0.99±0.01	0.99±0.01	0.99±0.01	0.99±0.01	0.99±0.01	0.99±0.01
Yeast5-ERL	F1-score	0.47±0.31	N.A.	N.A.	N.A.	0.73±0.31	0.93±0.30	0.93±0.26	0.93±0.36	0.87±0.15	0.65±0.07
	AP score	0.43±0.29	N.A.	N.A.	N.A.	0.70±0.29	0.90±0.29	0.90±0.25	0.90±0.34	0.80±0.17	0.52±0.07
	G-Mean	0.55±0.35	N.A.	N.A.	N.A.	0.80±0.34	1.00±0.33	1.00±0.29	1.00±0.39	1.00±0.04	1.00±0.00
	ACC (%)	98.9±0.30	93.09±0.97	94.25±1.74	99.25±0.32	99.31±0.31	99.48±0.23	98.73±0.33	99.88±0.08	99.19±0.30	95.49±0.28
	AUC	0.99±0.01	0.98±0.01	0.99±0.01	0.96±0.03	0.95±0.02	0.96±0.02	0.95±0.2	0.99±0.01	0.98±0.01	0.97±0.01
Cargood	F1-score	0.80±0.03	0.93±0.01	0.94±0.01	0.91±0.04	0.84±0.04	0.88±0.03	0.84±0.03	0.99±0.01	0.84±0.02	0.64±0.01
	AP score	0.66±0.04	0.98±0.00	0.99±0.00	0.84±0.06	0.73±0.06	0.78±0.04	0.72±0.05	0.99±0.02	0.73±0.03	0.47±0.01
	G-Mean	0.92±0.02	0.85±0.06	0.9±0.05	0.94±0.02	0.94±0.02	0.94±0.02	0.95±0.02	0.99±0.01	0.99±0.01	0.98±0.00
	ACC (%)	99.83±0.12	93.14±1.77	94.78±0.86	99.88±0.10	99.88±0.01	99.94±0.01	99.88±0.12	99.94±0.02	99.02±0.17	98.44±0.03
	AUC	0.99±0.01	0.99±0.01	0.99±0.01	0.98±0.01	0.99±0.01	0.99±0.01	0.99±0.01	0.99±0.01	0.99±0.01	0.99±0.01
Carvgood	F1-score	0.96±0.02	0.93±0.02	0.95±0.01	0.98±0.01	0.99±0.02	0.98±0.02	0.96±0.03	0.98±0.02	0.88±0.02	0.83±0.01
	AP score	0.92±0.03	0.98±0.01	0.99±0.00	0.97±0.03	0.99±0.03	0.97±0.04	0.93±0.05	0.97±0.03	0.78±0.03	0.72±0.01
	G-Mean	0.99±0.01	0.82±0.20	0.932±0.0	0.98±0.01	1.00±0.01	0.99±0.01	0.98±0.01	0.99±0.01	0.99±0.00	0.99±0.00
	ACC (%)	98.32±0.24	96.23±1.88	96.77±0.93	98.59±0.23	98.38±0.29	98.31±0.43	98.18±0.23	98.52±0.23	97.03±0.24	95.96±0.27
	AUC	0.93±0.01	0.97±0.02	0.95±0.02	0.93±0.03	0.93±0.03	0.91±0.03	0.96±0.03	0.85±0.02	0.97±0.01	0.97±0.01
Yeast5	F1-score	0.69±0.03	0.96±0.02	0.97±0.01	0.75±0.04	0.71±0.05	0.76±0.06	0.75±0.03	0.72±0.04	0.67±0.02	0.59±0.01
	AP score	0.50±0.04	0.45±0.06	0.46±0.10	0.57±0.05	0.53±0.07	0.59±0.07	0.60±0.04	0.54±0.05	0.50±0.02	0.43±0.01
	G-Mean	0.89±0.02	0.55±0.28	0.66±0.21	0.88±0.03	0.90±0.04	0.90±0.04	0.97±0.02	0.85±0.03	0.97±0.01	0.97±0.01
	ACC (%)	93.43±0.65	95.93±0.71	95.58±0.7	94.56±0.54	94.65±0.45	94.09±0.44	93.34±0.72	94.94±2.26	82.55±0.70	81.43±0.59
	AUC	0.67±0.01	0.94±0.02	0.95±0.02	0.61±0.03	0.62±0.02	0.57±0.02	0.72±0.03	0.55±0.13	0.83±0.02	0.86±0.02
Flare-F	F1-score	0.22±0.05	0.96±0.01	0.96±0.01	0.26±0.04	0.26±0.05	0.27±0.04	0.35±0.05	0.19±0.05	0.28±0.01	0.28±0.01
	AP score	0.10±0.02	0.40±0.03	0.44±0.04	0.11±0.01	0.11±0.02	0.11±0.01	0.18±0.03	0.08±0.02	0.15±0.01	0.15±0.01
	G-Mean	0.48±0.08	0.44±0.07	0.53±0.09	0.51±0.06	0.47±0.09	0.49±0.08	0.65±0.05	0.37±0.10	0.84±0.01	0.86±0.02

TABLE 3. Ablation analysis.

Dataset		CSRBoost	SMOTEBoost	RUSBoost
PSDAS	ACC (%)	72.85±0.8	75.24±0.52	N.A.
	AUC	0.66±0.01	0.63±0.01	N.A.
ESR	ACC (%)	92.05±0.3	91.58±0.28	N.A.
	AUC	0.90±0.00	0.89±0.01	N.A.
DCCC	ACC (%)	68.32±0.26	72.69±0.20	N.A.
	AUC	0.64±0.00	0.62±0.01	N.A.
BCW	ACC (%)	94.37±1.11	94.73±0.89	97.19±0.47
	AUC	0.94±0.01	0.95±0.01	0.97±0.01
CB	ACC (%)	76.43±2.43	76.91±1.80	82.26±2.09
	AUC	0.76±0.03	0.77±0.02	0.72±0.02
ESDRP	ACC (%)	97.31±0.85	97.19±0.82	98.65±0.50
	AUC	0.97±0.01	0.97±0.01	0.99±0.01
ILPD	ACC (%)	66.72±2.06	68.78±1.80	71.52±0.93
	AUC	0.66±0.01	0.63±0.02	0.66±0.02
Glass	ACC (%)	95.8±1.23	96.28±0.97	97.21±1.09
	AUC	0.93±0.01	0.94±0.02	0.94±0.02
Seed	ACC (%)	98.1±0.72	98.05±0.77	98.10±0.65
	AUC	0.98±0.07	0.98±0.01	0.98±0.01
Wine	ACC (%)	98.86±1.31	98.33±1.26	99.43±0.96
	AUC	0.98±0.01	0.98±0.02	0.99±0.01
Yeast5-ERL	ACC (%)	99.93±0.27	99.23±0.24	99.87±0.04
	AUC	0.60±0.31	0.99±0.01	0.99±0.01
Cargood	ACC (%)	98.9±0.3	99.13±0.29	99.19±0.30
	AUC	0.99±0.02	0.95±0.02	0.98±0.01
Carvgood	ACC (%)	99.83±0.12	99.78±0.16	99.02±0.17
	AUC	0.99±0.01	0.99±0.01	0.99±0.01
Yeast5	ACC (%)	98.32±0.24	97.52±0.19	97.03±0.24
	AUC	0.93±0.00	0.89±0.02	0.97±0.01
Flare-F	ACC (%)	93.43±0.65	93.93±0.43	82.55±0.7
	AUC	0.67±0.01	0.61±0.02	0.83±0.02

despite displaying somewhat lower accuracy. Datasets like Yeast5-ERL, which exhibits notably worse AUC performance and suggests susceptibility to specific data distributions, should be handled cautiously. RUSBoost performs well in several areas, but its lower AUC results might require careful consideration.

Class separation is given precedence over overall accuracy by CSRBoost; this trend is especially noticeable in the PSDAS, DCCC, and ILPD datasets. CSRBoost attains an accuracy of 72.85% and an AUC of 0.66 in the PSDAS dataset. Despite having a lower overall accuracy, CSRBoost is superior at class distinction, as evidenced by SMOTEBoost's greater accuracy of 75.24% but lower AUC of 0.63. Despite having a lesser accuracy, CSRBoost outperforms SMOTEBoost on the DCCC dataset, with an AUC of 0.64 instead of 0.62. RUSBoost achieves the maximum accuracy (71.52%) in the ILPD dataset and has an AUC of 0.66 with CSRBoost. SMOTEBoost, on the other hand, has a worse performance with an AUC of 0.63. For the ESR dataset, CSRBoost and SMOTEBoost perform very similarly, with an accuracy of almost 92.0% and an AUC of roughly 0.90. This suggests that all approaches are reliable and only slightly different. CSRBoost achieves nearly perfect performance for glass, seed, wine, cargood, and carvgood datasets.

A tradeoff between generalizing for the X_{min} and favoring the X_{maj} is suggested by the fact that RUSBoost usually produces the maximum ACC at the expense of the AUC. For the CB dataset, RUSBoost specifically exhibits the highest ACC of 82.26% but the lowest AUC of 0.72. On the other hand, CSRBoost exhibits a better balance in the precision-recall tradeoff despite having a somewhat lower accuracy. On the ESDRP dataset, all methods exhibit remarkable performance, with accuracies surpassing 97.0%. In this dataset, RUSBoost performs best, with an accuracy of 98.65% and an AUC of 0.99. For the BCW dataset, RUSBoost performs very well, with the greatest ACC of 97.19% and AUC of 0.97. SMOTEBoost and CSRBoost perform similarly, with AUC values ranging from 0.94 to 0.95 and ACC of about 94.0%. CSRBoost and SMOTEBoost outperform RUSBoost regarding ACC for the flare-F dataset. RUSBoost, on the other hand, performs noticeably better in terms of AUC (0.83 versus 0.67/0.61), suggesting that it is more effective in classification in an unbalanced setting.

Though often outpaced in both metrics, SMOTEBoost exhibits a balanced performance. With extremely imbalanced or complex datasets like Flare-F and Yeast5-ERL, the efficacy of this approach varies considerably. Since depending only on ACC might be deceptive, it is imperative to consider AUC as a measure of robustness, especially when working with imbalanced datasets.

F. TIME TO RUN ANALYSIS OF VARIOUS STATE-OF-THE-ART TECHNIQUES

Table 4 shows the execution times of various state-of-the-art methods across several datasets, expressed in seconds. Smaller datasets (like CB and Wine) are best suited for CSRBoost, but large-scale or high-dimensional data (like ESR and DCCC) exhibit poor scalability. Although at the expense of time, it provides higher outcomes and robustness. Performance and time efficiency are balanced in both GAN and SMOTified-GAN. Although they may be scaled fairly easily, they are more costly than simple approaches. Large datasets can have a consistently high execution time, ranging from 100 to 3251 seconds. With execution durations usually measured in single or low double digits for most datasets, traditional boosting techniques like AdaBoost and RUSBoost are the fastest overall. HUE rarely takes longer than 20 seconds and has excellent efficiency and consistency across various datasets. The complexity of the dataset causes CSRBoost and SMOTE-ENNs execution times to grow dramatically. AdaBoost and HUE, on the other hand, have shown consistent performance across all dataset sizes. In contrast, the execution time of GAN-based techniques increases quadratically with dataset size.

In general, CSRBoost takes longer time to compute than other approaches, particularly when working with bigger or more intricate datasets like ESR and DCCC. Although they are typically faster, traditional oversampling methods like ADASYN and SMOTE versions lack CSRBoost's sophisticated boosting features. In most situations, GAN-based techniques—such as SMOTified-GAN and GAN—remain noticeably faster than CSRBoost despite their somewhat lengthy execution periods. Except for smaller datasets where the execution time differences are minimal, HUE and AdaBoost are usually the fastest methods.

In trend, CSRBoost's variance is comparable to that of GANs, but when working with big datasets, it greatly

TABLE 4. Time to run (in seconds) comparison with various state-of-the-art techniques.

Dataset	CSRBoost	SMOTified-GAN	GAN	ADASYN	Borderline-SMOTE	SMOTE-ENN	SMOTE-Tomek	Adaboost	RUSBoost	HUE	SMOTEBoost
PSDAS	764	476	476	15	26	41	36	2	N.A.	13	7
ESR	8091	1197	1197	1853	371	37499	2269	795	N.A.	827	1095
DCCC	31053	3251	3251	286	330	240	205	35	N.A.	190	55
BCW	25	136	136	2	3	3	3	1	235	2	1
CB	9	110	105	N.A.	2	2	2	1	51	1	1
ESDRP	477	142	137	1	2	2	2	1	59	1	1
ILPD	58	133	132	2	1	1	1	1	183	2	1
Glass	9	115	113	1	1	1	1	2	58	2	1
Seed	20	103	101	1	1	1	1	1	8	1	1
Wine	15	119	110	1	14	2	11	14	10	1	1
Yeast5-ERL	6	N.A.	N.A.	N.A.	1	1	1	1	31	22	1
Cargood	36	306	309	2	1	2	2	1	123	7	2
Carvgood	34	311	309	2	1	2	2	2	89	5	2
Yeast5	24	240	238	2	1	2	2	1	101	12	2
Flare-F	28	195	193	25	1	2	2	1	337	17	37

TABLE 5. Wilcoxon signed-rank comparison with the proposed CSRBoost method.

Dataset	SMOTified-GAN	GAN	ADASYN	Borderline-SMOTE	SMOTE-ENN	SMOTE-Tomek	Adaboost	RUSBoost	HUE	SMOTEBoost
PSDAS	3.71E-18↓	3.63E-18↓	3.63E-14↑	5.91E-18↑	2.74E-18↑	7.46E-08↑	1.95E-06↑	N.A.	2.49E-18↓	7.85E-18↑
ESR	1.57E-18↓	1.68E-18↓	1.83E-12↑	4.57E-11↑	1.98E-11↑	1.16E-12↑	4.1E-11↓	N.A.	1.7E-18↓	1.06E-13↑
DCCC	1.76E-18↓	1.3E-18↓	4.35E-18↑	3.13E-17↑	3.82E-17↑	7.16E-11↓	3.69E-17↑	1.52E-23↑	1.77E-18↓	1.78E-17↑
BCW	2.92E-18↓	3.1E-18↓	2.88E-01–	6.16E-01–	7.52E-01–	7.47E-02–	1.16E-01–	1.67E-18↓	2.41E-14↓	5.15E-6↓
CB	3.54E-18↓	3.55E-18↓	3.59E-18↑	7.51E02–	2.50E-07↓	3.13E-05↑	8.07E-01–	6.37E-14↑	1.39E-05↓	1.44E-02↓
ESDRP	1.04E-15↓	2.44E-08↓	4.80E-09↓	8.74E-01–	1.57E-01–	1.43E-17↑	7.17E-01–	1.15E-15↓	1.87E-01–	6.72E-01–
ILPD	1.13E-17↓	2.22E-16↓	6.86E-08↑	2.84E-04↑	5.24E-15↑	6.71E-16↓	4.09E-16↑	4.41E-01–	1.08E-17↓	2.04E-15↑
Glass	2.54E-17↑	1.39E-17↑	8.13E-10↓	9.01E-01–	4.59E-01–	4.88E-03↓	2.88E-18↓	2.10E-05↓	9.8E-16↓	8.27E-04↓
Seed	5.51E-01–	4.42E-01–	5.85E-01–	7.92E-01–	5.89E-01–	9.20E-01–	1.15E-17↑	3.58E-01–	8.98E-01–	6.28E-01–
Wine	4.03E-03↑	2.39E-07↑	8.34E-01–	2.49E-01–	7.49E-01–	1.40E-05↑	9.01E-01–	3.06E-11↓	7.00E-01–	6.91E-01–
Yeast5-ERL	N.A.	N.A.	N.A.	2.98E-16↓	2.71E-16↓	2.98E-16↓	4.58E-16↓	3.21E-16↓	2.77E-16↓	3.25E-16↓
Cargood	7.89E-10↑	2.03E-01–	4.68E-13↑	1.22E-17↑	2.35E-14↑	6.22E-10↑	2.46E-01–	3.05E-09↑	1.01E-15↑	9.58E-18↑
Carvgood	9.01E-01–	4.40E-01–	2.42E-09↑	6.64E-01–	7.84E-01–	5.51E-01–	2.16E-01–	3.81E-01–	8.81E-01–	7.21E-01–
Yeast5	5.39E-17↓	4.00E-09↓	4.77E-01–	8.41E-01–	1.04E-07↑	8.59E-15↓	3.34E-18↑	3.07E-18↓	6.56E-18↓	3.69E-17↑
Flare-F	3.41E-18↓	3.37E-18↓	5.14E-12↑	4.54E-18↑	3.44E-18↑	6.08E-17↓	7.99E-14↑	3.56E-18↓	3.43E-18↓	3.45E-18↑

outperforms it. It implies that CSRBoost may involve cascading dependencies and has a non-linear temporal complexity. CSRBoost offers good accuracy and durability across various datasets despite its computationally demanding nature. The size and imbalance of the dataset cause the execution time to increase significantly. CSRBoost frequently works better than many other techniques for small to medium-scale tasks. Despite its increased time, CSRBoost is still recommended for large, unbalanced datasets where infrastructure supports is crucial.

G. WILCOXON SIGNED-RANK ANALYSIS OF VARIOUS STATE-OF-THE-ART TECHNIQUES

The Wilcoxon signed-rank test [62] results, which contrast the suggested CSRBoost with the state-of-the-art methods, are shown in the table 5. The Wilcoxon test's p-value is displayed in each cell. The arrow shows the direction of superiority (↑: CSRBoost significantly better, ↓: CSRBoost significantly worse, and −: No significant difference). The majority of p-values that have been recorded are incredibly modest (10^{-18} to 10^{-8}), suggesting that CSRBoost and its competitors differ significantly. In certain instances (e.g., Seed, Carvgood), p-values are more than 0.05, indicating no meaningful difference. It demonstrates that CSRBoost's superiority is not constant across datasets. Table 5 implies that generative models (GAN-based) are better at capturing the data distribution than the proposed CSRBoost method. Compared to ADASYN, Borderline-SMOTE, SMOTE-ENN, SMOTE-Tomek, and SMOTEBoost, CSRBoost yields mixed results. This suggests that while resampling-based boosting techniques are competitive with CSRBoost's design (clustered undersampling + boosting), they are not always superior. CSRBoost often outperforms AdaBoost on datasets such as PSDAS, DCCC, ILPD, Yeast5, and Flare-F. However, a few datasets (Glass, ESR, and CB) show comparable or superior performance with AdaBoost. This highlights the importance of using resampling techniques in boosting methods for imbalanced learning.

Table 5 indicates that significant comparative patterns may be seen in the Wilcoxon signed-rank test results. In several datasets, CSRBoost outperforms ADASYN, SMOTE-ENN, SMOTE-Tomek, and SMOTEBoost statistically significantly, confirming the efficacy of clustered undersampling combined with boosting. However, CSRBoost routinely outperforms GAN-based methods (GAN, SMOTified-GAN), suggesting that deep generative resampling is still more effective at capturing minority-class structure. Additionally, dataset-specific behaviours are revealed: CSRBoost performs best on structured or high sample size datasets like Cargood and Flare-F, but its performance deteriorates on low sample size or highly skewed datasets like BCW. These findings demonstrate that although CSRBoost may compete with traditional resampling-boosting ensembles, its benefit is conditional. To increase robustness, future extensions

should investigate integrating CSRBoost with generative resampling.

IV. CONCLUSION

CSRBoost is an ensemble learning framework designed for imbalanced classification tasks in this paper. CSRBoost combines a model-level boosting strategy with data-level resampling techniques to provide a principled solution for class imbalance. CSRBoost efficiently reduces class imbalance while preserving the structural integrity of the original data distribution by fusing adaptive boosting with an intelligent class-specific resampling algorithm. The method leverages a dynamic clustering approach to preserve the intra-class density, data locality, and relevant structure. Through comprehensive validation of multiple datasets of different categories, it has been found that CSRBoost achieves decent performance by providing the appropriate representatives from each class to construct a balanced dataset for classifier training. The framework also emphasizes challenging instances during the training phase to enhance classification accuracy. Results indicate that CSRBoost performs well compared to traditional methods across a variety of datasets with different levels of imbalance ratios. Additionally, CSRBoost is a dependable and scalable solution for situations with unbalanced data because it showed statistically significant gains with reduced variation. Furthermore, the proposed CSRBoost approach is versatile, as various types of clusters can be created by adopting multiple techniques with varying resampling approaches and base classifiers. Finally, it completely depends on the dataset's dimensionality and complexity, as seen in the result.

Future work will explore the integration of CSRBoost with deep learning architectures and its extension to multi-class and streaming data environments.

REFERENCES

[1] Y. Ma, Y. Tian, N. Moniz, and N. V. Chawla, "Class-imbalanced learning on graphs: A survey," *ACM Comput. Surv.*, vol. 57, no. 8, pp. 1–16, Aug. 2025.

[2] S. Rezvani and X. Wang, "A broad review on class imbalance learning techniques," *Appl. Soft Comput.*, vol. 143, Aug. 2023, Art. no. 110415.

[3] A. S. Tarawneh, A. B. Hassanat, G. A. Altarawneh, and A. Almuhaimeed, "Stop oversampling for class imbalance learning: A review," *IEEE Access*, vol. 10, pp. 47643–47660, 2022.

[4] S. Yadav, D. Joshi, S. Mulye, S. S. Udmale, and G. P. Bhole, "SMOTEHashBoost: Ensemble algorithm for imbalanced dataset pattern classification," *IEEE Access*, vol. 13, pp. 58092–58107, 2025.

[5] G. Aguiar, B. Krawczyk, and A. Cano, "A survey on learning from imbalanced data streams: Taxonomy, challenges, empirical study, and reproducible experimental framework," *Mach. Learn.*, vol. 113, no. 7, pp. 4165–4243, Jul. 2024.

[6] S. Susan and A. Kumar, "The balancing trick: Optimized sampling of imbalanced datasets—A brief survey of the recent state of the art," *Eng. Rep.*, vol. 3, no. 4, Apr. 2021, Art. no. e12298.

[7] W.-C. Lin, C.-F. Tsai, Y.-H. Hu, and J.-S. Jhang, "Clustering-based undersampling in class-imbalanced data," *Inf. Sci.*, vols. 409–410, pp. 17–26, Oct. 2017.

[8] Z. Liu, Y. Li, N. Chen, Q. Wang, B. Hooi, and B. He, "A survey of imbalanced learning on graphs: Problems, techniques, and future directions," *IEEE Trans. Knowl. Data Eng.*, vol. 37, no. 6, pp. 1–20, Jun. 2025.

[9] X. Xie, H. Liu, S. Zeng, L. Lin, and W. Li, "A novel progressively undersampling method based on the density peaks sequence for imbalanced data," *Knowl.-Based Syst.*, vol. 213, Feb. 2021, Art. no. 106689.

[10] Z. Sun, W. Ying, W. Zhang, and S. Gong, "Undersampling method based on minority class density for imbalanced data," *Expert Syst. Appl.*, vol. 249, Sep. 2024, Art. no. 123328.

[11] J. Hoyos-Osorio, A. Alvarez-Meza, G. Daza-Santacoloma, A. Orozco-Gutierrez, and G. Castellanos-Dominguez, "Relevant information undersampling to support imbalanced data classification," *Neurocomputing*, vol. 436, pp. 136–146, May 2021.

[12] A. Farshidvard, F. Hooshmand, and S. A. MirHassani, "A novel two-phase clustering-based under-sampling method for imbalanced classification problems," *Expert Syst. Appl.*, vol. 213, Mar. 2023, Art. no. 119003.

[13] Q. Zhou and B. Sun, "Adaptive K-means clustering based under-sampling methods to solve the class imbalance problem," *Data Inf. Manage.*, vol. 8, no. 3, Sep. 2024, Art. no. 100064.

[14] L. Bai, T. Ju, H. Wang, M. Lei, and X. Pan, "Two-step ensemble undersampling algorithm for massive imbalanced data classification," *Inf. Sci.*, vol. 665, Apr. 2024, Art. no. 120351.

[15] Q. Dai, J.-W. Liu, and Y.-H. Shi, "Class-overlap undersampling based on Schur decomposition for class-imbalance problems," *Expert Syst. Appl.*, vol. 221, Jul. 2023, Art. no. 119735.

[16] P. Soltanzadeh, M. R. Feizi-Derakhshi, and M. Hashemzadeh, "Addressing the class-imbalance and class-overlap problems by a metaheuristic-based under-sampling approach," *Pattern Recognit.*, vol. 143, Nov. 2023, Art. no. 109721.

[17] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.

[18] P. Soltanzadeh and M. Hashemzadeh, "RCSMOTE: Range-controlled synthetic minority over-sampling technique for handling the class imbalance problem," *Inf. Sci.*, vol. 542, pp. 92–111, Jan. 2021.

[19] L. Camacho, G. Douzas, and F. Bacao, "Geometric SMOTE for regression," *Expert Syst. Appl.*, vol. 193, May 2022, Art. no. 116387.

[20] A. Islam, S. B. Belhaouari, A. U. Rehman, and H. Bensmail, "KNNOR: An oversampling technique for imbalanced datasets," *Appl. Soft Comput.*, vol. 115, Jan. 2022, Art. no. 108288.

[21] F. Rodríguez-Torres, J. F. Martínez-Trinidad, and J. A. Carrasco-Ochoa, "An oversampling method for class imbalance problems on large datasets," *Appl. Sci.*, vol. 12, no. 7, p. 3424, Mar. 2022.

[22] J. Li, Q. Zhu, Q. Wu, and Z. Fan, "A novel oversampling technique for class-imbalanced learning based on SMOTE and natural neighbors," *Inf. Sci.*, vol. 565, pp. 438–455, Jul. 2021.

[23] I. D. Mienye and Y. Sun, "A survey of ensemble learning: Concepts, algorithms, applications, and prospects," *IEEE Access*, vol. 10, pp. 99129–99149, 2022.

[24] W. Chen, K. Yang, Z. Yu, Y. Shi, and C. L. P. Chen, "A survey on imbalanced learning: Latest research, applications and future directions," *Artif. Intell. Rev.*, vol. 57, no. 6, pp. 1–51, May 2024.

[25] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. ICML*, 1996, pp. 148–156.

[26] K. Ghosh, C. Bellinger, R. Corizzo, P. Branco, B. Krawczyk, and N. Japkowicz, "The class imbalance problem in deep learning," *Mach. Learn.*, vol. 113, no. 7, pp. 4845–4901, Jul. 2024.

[27] A. Abdelkhalek and M. Mashaly, "Addressing the class imbalance problem in network intrusion detection systems using data resampling and deep learning," *J. Supercomput.*, vol. 79, no. 10, pp. 10611–10644, Jul. 2023.

[28] H. Lu, L. Ehwerhemuepha, and C. Rakovski, "A comparative study on deep learning models for text classification of unstructured medical notes with various levels of class imbalance," *BMC Med. Res. Methodol.*, vol. 22, no. 1, p. 181, Dec. 2022.

[29] Z. Chen, J. Duan, L. Kang, and G. Qiu, "Class-imbalanced deep learning via a class-balanced ensemble," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 10, pp. 5626–5640, Oct. 2022.

[30] A. Bria, C. Marrocco, and F. Tortorella, "Addressing class imbalance in deep learning for small lesion detection on medical images," *Comput. Biol. Med.*, vol. 120, May 2020, Art. no. 103735.

[31] Y. Li, J. Cao, Y. Xu, L. Zhu, and Z. Y. Dong, "Deep learning based on transformer architecture for power system short-term voltage stability assessment with class imbalance," *Renew. Sustain. Energy Rev.*, vol. 189, Jan. 2024, Art. no. 113913.

[32] A. Sharma, P. K. Singh, and R. Chandra, "SMOTified-GAN for class imbalanced pattern classification problems," *IEEE Access*, vol. 10, pp. 30655–30665, 2022.
[33] F. Rayhan, S. Ahmed, A. Mahbub, R. Jani, S. Shatabda, and D. M. Farid, "CUSBoost: Cluster-based under-sampling with boosting for imbalanced classification," in *Proc. 2nd Int. Conf. Comput. Syst. Inf. Technol. Sustain. Solution (CSITSS)*, Dec. 2017, pp. 1–5.
[34] A. Salehi and M. Khedmati, "Hybrid clustering strategies for effective oversampling and undersampling in multiclass classification," *Sci. Rep.*, vol. 15, no. 1, p. 3460, Jan. 2025.
[35] M. Bach, P. Trofimiak, D. Kostrzewa, and A. Werner, "CLEANSE—Cluster-based undersampling method," *Proc. Comput. Sci.*, vol. 225, pp. 4541–4550, Jan. 2023.
[36] C. Yang, E. A. Fridgeirsson, J. A. Kors, J. M. Reps, and P. R. Rijnbeek, "Impact of random oversampling and random undersampling on the performance of prediction models developed using observational health data," *J. Big Data*, vol. 11, no. 1, p. 7, Jan. 2024.
[37] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "RUSBoost: A hybrid approach to alleviating class imbalance," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 40, no. 1, pp. 185–197, Jan. 2010.
[38] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explor. Newslett.*, vol. 6, no. 1, pp. 20–29, Jun. 2004.
[39] B. Krawczyk, "Learning from imbalanced data: Open challenges and future directions," *Prog. Artif. Intell.*, vol. 5, no. 4, pp. 221–232, Nov. 2016.
[40] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.
[41] A. Ferreira and M. A. T. Figueiredo, "Boosting algorithms: A review of methods, theory, and applications," in *Ensemble Machine Learning: Methods and Applications*. New York, NY, USA: Springer, 2012, pp. 35–85.
[42] H. Binder, O. Gefeller, M. Schmid, and A. Mayr, "The evolution of boosting algorithms," *Methods Inf. Med.*, vol. 53, no. 6, pp. 419–427, 2014.
[43] M. J. Zaki and W. Meira, *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2014.
[44] V. Realinho, M. Vieira Martins, J. Machado, and L. Baptista, "Predict students' dropout and academic success," UCI Mach. Learn. Repository, 2021, doi: 10.24432/C5MC89.
[45] R. G. Andrzejak, K. Lehnertz, F. Mormann, C. Rieke, P. David, and C. E. Elger, "Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 64, no. 6, Nov. 2001, Art. no. 061907.
[46] I.-C. Yeh, "Default of credit card clients," UCI Mach. Learn. Repository, 2009, doi: 10.24432/C55S3H.
[47] W. Wolberg, O. Mangasarian, N. Street, and W. Street, "Breast cancer Wisconsin (diagnostic)," UCI Mach. Learn. Repository, 1993, doi: 10.24432/C5DW2B.
[48] T. Sejnowski and R. Gorman, "Connectionist bench (sonar, mines vs. rocks)," 1988, doi: 10.24432/C5T01Q.
[49] "Early stage diabetes risk prediction," UCI Mach. Learn. Repository, Tech. Rep., 2020, doi: 10.24432/C5VG8H.
[50] B. Ramana and N. Venkateswarlu, "ILPD (Indian liver patient dataset)," UCI Mach. Learn. Repository, 2022, doi: 10.24432/C5D02C.
[51] B. German, "Glass Identification," UCI Mach. Learn. Repository, 1987, doi: 10.24432/C5WW2P.
[52] M. Charytanowicz, J. Niewczas, P. Kulczycki, P. Kowalski, and S. Lukasik, "Seeds," UCI Mach. Learn. Repository, 2010, doi: 10.24432/C5H30K.
[53] S. Aeberhard and M. Forina, "Wine," UCI Mach. Learn. Repository, 1992, doi: 10.24432/C5PC7J.
[54] K. Nakai, "Yeast," UCI Mach. Learn. Repository, 1991, doi: 10.24432/C5KG68.
[55] M. Bohanec, "Car evaluation," UCI Mach. Learn. Repository, 1988, doi: 10.24432/C5JP48.
[56] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *J. Multiple-Valued Log. Soft Comput.*, vol. 17, pp. 255–287, Jun. 2011.
[57] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IEEE World Congr. Comput. Intell.)*, Jun. 2008, pp. 1322–1328.
[58] H. Han, W. Wang, and B. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," in *Proc. Int. Conf. Intell. Comput.*, 2005, pp. 878–887.
[59] G. E. A. P. A. Batista, A. L. C. Bazzan, and M. C. Monard, "Balancing training data for automated annotation of keywords: A case study," *Wob*, vol. 3, pp. 10–18, Dec. 2003.
[60] W. W. Y. Ng, S. Xu, J. Zhang, X. Tian, T. Rong, and S. Kwong, "Hashing-based undersampling ensemble for imbalanced pattern classification problems," *IEEE Trans. Cybern.*, vol. 52, no. 2, pp. 1269–1279, Feb. 2022.
[61] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, *Learning From Imbalanced Data Sets*, vol. 10. Cham, Switzerland: Springer, 2018.
[62] J. Bird, "Wilcoxon signed-rank test," in *Wiley Encyclopedia of Clinical Trials*. Wiley, 2007, pp. 1–3.



SEEMA YADAV is currently pursuing the Ph.D. degree in computer engineering with Veermata Jijabai Technological Institute (VJTI), Mumbai, India. She is an Assistant Professor with the Department of Information Technology, K. J. Somaiya Institute of Technology, Sion, Mumbai. Her current research interests include machine learning, data science, and optimization and pattern analysis.



DHRUVANSHU JOSHI is currently pursuing the B.Tech. degree in computer engineering from Veermata Jijabai Technological Institute (VJTI), Mumbai, Maharashtra, India. He has contributed to Google Summer of Code 2024 at INCF and interned at PyMC Labs, focusing on Bayesian modeling and statistical methods. A global runner-up in the OpenCV Spatial AI competition 2022, his research interests include artificial intelligence, machine learning, computer vision, and optimization algorithms, with work spanning multimodal deep learning systems and data-driven decision-making.



SOHAM MULYE is currently pursuing the B.Tech. degree in computer engineering from Veermata Jijabai Technological Institute (VJTI), Mumbai, Maharashtra, India. He has made significant contributions as both a Developer and a Mentor in Google Summer of Code, participating, in 2023 and mentoring, in 2024 at INCF. He secured 2nd place globally in the OpenCV Spatial AI competition for his work on depth estimation. He has also interned at Citi Bank India, where he built scalable web applications to improve operational efficiency. His research interests include machine learning, computer vision, and algorithm optimization, with experience in developing real-world applications, such as EEG-based sleep stage classification and automated cost-analysis systems.



LABIB ASARI is currently pursuing the B.Tech. degree in computer engineering with Veermata Jijabai Technological Institute (VJTI), Mumbai, Maharashtra, India. His previous research focuses on class-imbalance, tokenization methods, GPU-based NoC characterization, and cache-side attacks. As a Google Summer of Code Mentor and Contributor for GNUAstro, he integrated GPU support for astronomical data processing, accelerating parallel workloads by 400x and developed a resilient error-handling framework. As a Summer Intern at Procter and Gamble, he automated large-scale data harmonization of numerous disparate data signals using NLP and generative AI. He has also contributed to AI applications at multiple early-stage startups. His research interests include GPU optimization, algorithm design, robotics, machine learning, and NLP and LLMs.



SANDEEP S. UDMALE (Senior Member, IEEE) received the Ph.D. degree in computer science and engineering from Indian Institute of Technology (IIT-BHU), Varanasi, Uttar Pradesh, India, in 2019. He is currently an Assistant Professor with the Department of Computer Engineering and Information Technology, Veermata Jijabai Technological Institute (VJTI), Mumbai, India. His current research interests include machine learning, data science, optimization, and pattern analysis. He is a member of the ACM and the Computer Society of India.



GIRISH P. BHOLE (Member, IEEE) received the Ph.D. degree in engineering from University of Mumbai, Mumbai, Maharashtra, India, in 2008. He is currently an Adjunct Professor with the Department of Computer Engineering and Information Technology, Veermata Jijabai Technological Institute (VJTI), Mumbai. He has extensive experience in setting up and managing the data center. His research interests include machine learning, data science, the IoT, and wireless networks.