IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 4 March 2025, accepted 15 March 2025, date of publication 28 March 2025, date of current version 8 April 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3555565

RESEARCH ARTICLE

SMOTEHashBoost: Ensemble Algorithm for Imbalanced Dataset Pattern Classification

SEEMA YADAV, DHRUVANSHU JOSHI, SOHAM MULYE, 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 Class imbalance occurs frequently in machine learning, particularly in binary classification tasks where the majority class has a significantly larger number of samples than the minority class. The majority class is often favored by conventional classifiers, which can lead to biases from improper oversampling or subpar performance when detecting instances of the minority class. Consequently, there is growing concern about algorithmic fairness. Thus, the goal is to improve the ability to identify different discriminatory patterns in the data by creating a large number of test cases, which will result in more fair and unbiased model performance. Here, we introduce SMOTEHashBoost, an ensemble learning approach that overcomes class imbalance by combining three core methods: SMOTE oversampling, hash-based undersampling, and AdaBoost boosting. The proposed approach combines weak classifiers to increase the model's ability to generalize while generating a balanced and representative dataset. The effectiveness of SMOTEHashBoost is demonstrated by improved performance in handling imbalanced datasets, which makes it a dependable solution for skewed class distributions in real-world classification problems.

INDEX TERMS Class imbalance, ensemble classifier, HUE, RUSBoost, SMOTE.

I. INTRODUCTION

Generally, standard algorithms for classification assume a balanced data distribution, which indicates that there are an equal or nearly equal number of examples of each class in the dataset [1], [2]. But in many real-world situations, this assumption is frequently incorrect. For example, in the COVID-19 disease classification system [3], the majority class is represented by healthy patient data, whereas COVID-19 patient data form the minority class. Crucially, end users frequently prioritize concerns for the minority class. Thus, the most challenging aspect of training a classification model on imbalanced data is figuring out which class to focus on, usually the minority class. However, if every occurrence is given the same weight, classifiers might ignore the minority class more frequently. Because of its higher representation, the majority class tends to predominate and add more to the overall accuracy (ACC) of the classifier, which could result in the minority class being poorly detected. When there is a large discrepancy in the distribution of classes, traditional data mining techniques tend to favor the over-represented majority class. Due to this bias, the minority class—which is often the positive class—is not as well detected, resulting in misclassifications that cost the most [1], [2], [4], [5]. Thus, creative approaches are needed to guarantee that the model correctly detects these crucial but infrequent occurrences.

Several approaches have been put out to address the class imbalance, the most well-known of which are data processing (sampling), algorithmic (Ensemble methods), and advanced (synthetic data generation) [1], [2], [4], [6]. However, the simplest way is to either decrease the number of instances of majority classes (undersampling) or raise the number of instances of minority classes (oversampling). Consequently, data sampling aims to modify the class distribution in the training data. Undersampling and oversampling can now be done using a variety of techniques. The simplest

The associate editor coordinating the review of this manuscript and approving it for publication was Chien-Ming Chen.

© 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/

method of resampling a dataset is random resampling. Replicating instances from the minority class balances the dataset by random oversampling until the target class ratio is attained. Similarly, random undersampling (RUS) diminishes the majority class by arbitrarily removing cases until the desired balance is reached. Furthermore, there exist increasingly advanced and ''intelligent'' methods for both undersampling and oversampling [1], [2], [4], [6], [7]. But, each oversampling and undersampling strategy has benefits and drawbacks of its own.

Therefore, a simpler and faster imbalance dataset learning algorithm, i.e., RUSBoost, has been proposed to achieve decent performance and to reduce the model training time [8]. The discovery and thinning clusters of the majority class have been proposed by Bach et al., which eliminates the data samples of high-density areas of the majority class [9]. Likewise, the clustering approach has been employed in the data pre-processing stage for an undersampling strategy to construct an effective machine learning model for imbalanced datasets [10]. The imbalanced class distribution problem has been resolved by cluster-based undersampling approaches, which discover the representative samples from minority classes to enhance classification ACC [11]. Besides, the data samples near the majority class's decision boundary have only been accepted with the minority class to train the weighted support vector machine for biological imbalanced data [12]. The principal drawback associated with undersampling is the information loss that arises from eliminating samples from the training dataset [1], [2], [4], [6], [7]. Nonetheless, a significant benefit is that because of the reduced dataset size, it takes less time to train models. As a result, it affects the classifier's performance and causes a loss in generality.

By keeping every example in the resampled data, oversampling, on the other hand, maintains all the information from the original dataset. The resampled dataset contains either newly created minority class examples or duplicates, depending on the oversampling approach used [1], [2], [4], [6], [7]. K-Nearest Neighbor (KNN) OveRsampling method has been applied to minority classes by discovering the important and secure areas for augmentation and producing synthetic data samples [13]. To address the time-consuming process of KNN, a fast oversampling method, i.e., Fast Synthetic Minority Oversampling Technique (SMOTE), has been constructed by performing the feature analysis of minority class [14]. Additionally, a weighted oversampling approach has been applied to the hard-to-learn data points of the minority class to compute appropriate weight using Euclidean distance from the majority class [15]. The Localized Random Affine Shadowsampling has been employed for the minority class by Bej et al. in [16] to estimate the local probability distribution and reduce misclassification properly. It can be concluded that oversampling does not eliminate information, but it does have certain drawbacks. It lengthens the time needed for model training by producing duplicates or new examples. Also, it might result in overfitting if duplicate instances are used [17]. The literature states that SMOTE is a widely utilized approach among various dataset-balancing techniques. Over the years, various variants of SMOTE have been developed to address the data imbalance issue and improve its performance [1], [2], [4], [6], [7].

Recently, an SMOTE-based oversampling approach for the encoder/decoder deep learning framework has been proposed to mitigate the issue of imbalanced data [18]. Similarly, the quality and size of datasets have been improved for deep learning model training through a standard data augmentation approach [19]. The concept of extended oversampling has been introduced to reduce class imbalance in image datasets and enhance the learning of convolutional neural networks [20]. Additionally, generative adversarial networks (GANs) have been adopted as data augmenters to perform the meta-analysis of GitHub projects [21]. Moreover, class rebalancing for minority classes was performed using the meta-learning approach, which aimed to augment the demand space for these classes [22]. Further meta-knowledge has been improved by balancing the learning variables to achieve a better solution [23]. Also, the meta-learning approach has been adopted for a few shot learning algorithms during evaluation through meta-dataset to enhance the performance [24].

Besides the above-advanced techniques, boosting is another technique to enhance classification performance. Boosting enhances the effectiveness of any weak classifier, independent of data balance, in contrast to many data sampling strategies created expressly to address class imbalance [25], [26]. AdaBoost [27] is the most used boosting technique; it builds an ensemble of models iteratively. To make misclassified instances more likely to be correctly classified in the following iteration, the weights of those examples are modified in each iteration. The models jointly contribute to classifying newly found, unlabelled data through a weighted voting method once all iterations have been completed. Minority class instances, which are more likely to be incorrectly classified, are given more weight in the following rounds, making this method especially useful for addressing class imbalance. Boosting is occasionally referred to as an enhanced technique for data sampling [7], [25], [26]. Either ''reweighting'' or ''resampling'' can be used to boost [28], [29]. Not all algorithms can use these weights in their decision-making; in reweighting, the changed example weights are transmitted straight to the base learner in each iteration. Since the training data is resampled based on the specified weights, we apply boosting by resampling in this research. The resampled data is then utilized to create the model for each iteration.

Techniques for artificial oversampling have shown to be quite successful in rectifying skewed datasets [1], [2], [4], [6], [7]. However, given that data-driven algorithms have been shown to generate biases, worries about algorithmic fairness are growing. This issue occurs because when creating synthetic instances, they do not consider the majority class

distribution. The goal is to improve the ability to identify different discriminatory patterns in the data by creating many test cases, which will result in more fair and objective model performance. To improve the performance of models trained on imbalanced datasets, we present in this study a novel hybrid technique called SMOTEHashBoost, which combines boosting and data sampling. SMOTEHashBoost is a boosting model framework that integrates hashing-based sampling and SMOTE to handle class imbalance. Using a hashing-based undersampling ensemble technique, SMOTEHashBoost picks a representative sample of majority class examples almost uniformly at random for constructing the subsets. By creating artificial instances for the under-represented class, the approach uses SMOTE to oversample the minority class and guarantee a balanced dataset in the subset. By doing this, the skewed distribution is lessened, and the classifier's capacity to identify instances of minority classes is enhanced. This method enriches the solution space and avoids bias towards the majority class by minimizing duplication in the majority class, which generates different training subsets. These varied subsets serve as the basis for training the basic classifiers, and in the following rounds, boosting is used to iteratively improve the model by emphasizing misclassified examples. SMOTEHashBoost builds a strong and efficient final model that can handle extremely imbalanced data by merging these classifiers with boosting.

The paper is organized as follows: The theoretical basis and the suggested technique are presented in Section II. We present the dataset and offer a detailed analysis of the findings in Section III. Lastly, Section IV highlights our research contributions and summarizes the conclusions derived from our findings.

II. THEORETICAL BACKGROUND

The fundamental ideas of SMOTE, hash-based sampling, and AdaBoost are presented in this section. First, each technique's underlying ideas are clarified, giving its application perspective. The suggested hybrid strategy is then given, which combines these techniques to address class imbalance more successfully.

A. SYNTHETIC MINORITY OVER-SAMPLING TECHNIQUE (SMOTE)

One popular technique for resolving class imbalance in datasets is the SMOTE [30]. SMOTE balances the class distribution by creating synthetic instances rather than just copying minority class examples, which enhances the efficiency of classification algorithms on imbalanced data. Following are the steps of SMOTE:

1) Identify Minority Class Instances: Let the dataset $D = \{(X_i, Y_i)\}_{i=1}^{N}$ contain N instances, where X_{min} refers to the minority class instances and X_{maj} to the majority class instances. The goal is to increase the number of X_{min} instances by generating synthetic samples.
2) Select Nearest Neighbors: For each instance x_i in the X_{min}, the KNN of that instance are identified based on the Euclidean distance metric. Let $K\text{NN}(x_i)$ represent the set of NNs for x_i.
3) Generate Synthetic Data: A synthetic instance x_{new} is generated by selecting one of the KNNs, x_{KNN}, and interpolating a new point between x_i and x_{KNN}:

$$x_{new} = x_i + \delta \times (x_{KNN} - x_i) \tag{1}$$

where δ is a random number between 0 and 1. This ensures that the synthetic sample lies on the line segment between x_i and x_{KNN}.
4) Repeat: This process is repeated until the X_{min} instances are increased to the desired level, balancing the class distribution.

SMOTE is still one of the most popular and successful methods for handling imbalanced datasets, especially when X_{min} performance is important. Models generalise more effectively and identify significant trends in X_{min} thanks to its capacity to provide meaningful synthetic data instead of replicating samples [1], [2], [4], [6], [7]. SMOTE lowers the chance of overfitting to the X_{min} by creating fresh synthetic cases instead of random oversampling, which duplicates instances. Classifiers can better identify uncommon but significant patterns by concentrating on the X_{min} when the dataset is balanced, which improves recall and F1 scores [1], [2], [4], [6], [7]. SMOTE is acceptable to various application domains. It is employed with different classification procedures. Also, numerous studies have shown that SMOTE is more effective than other oversampling methods because it adaptively addresses the issue of class imbalance [30], [31]. However, it generates artificial samples without assuming the X_{maj} distribution. As a result, classification performance may be negatively influenced by noisy or overlapping data samples, particularly close to the decision border. The KNN in high-dimensional datasets may be far from the target point, rendering interpolation and producing synthetic points that might not be suitable for classification. SMOTE can be computationally expensive for large datasets or when KNN searches are expensive because it necessitates the computation of NNs. SMOTE can produce synthetic instances that obfuscate class boundaries when the X_{min} and X_{maj} distributions overlap, which raises the possibility of misclassification [1], [2], [4], [6], [7].

B. HASH-BASED UNDERSAMPLING ENSEMBLE (HUE) METHOD

Using a hash-based undersampling technique, a hash function maps the data points into a lower-dimensional space. The X_{maj} data is then uniformly selected or sampled using these mappings, resulting in representative and diversified subsets. This process is repeated by the Hash-based Undersampling Ensemble (HUE) method [32] to produce numerous balanced datasets, each of which is used to train a base classifier. The classifiers' predictions are combined into a single final

prediction using an ensemble framework. Following are the steps of HUE:

1) Dataset representation: Let the dataset D consists of X_{min} and X_{maj} class instances. The goal is to reduce the size of the X_{maj} class while preserving diversity.
2) Hash function $hf(x)$: Define $hf(x)$: $\mathbb{R}^d \rightarrow \{0, 1, \ldots.., k-1\}$, which maps the x_i into a hash space of size k. Within hash space, randomly sample a small subset of X_{maj} examples. This ensures that the X_{maj} examples are undersampled uniformly across the feature space, rather than being biased towards any particular region.
3) Ensemble construction: Repeat the undersampling process m times to create m different undersampled datasets $D_1, D_2, \ldots.., D_m$, each containing all X_{min} class examples and a subset of X_{maj} class examples. For each dataset D_i, train a base classifier $h_{b_i}(x)$.
4) Prediction aggregation: Combine the predictions from the m $h_{b_i}(x)$ using a majority vote or weighted vote approach. The final ensemble classifier $EC(x)$ can be defined as:

$$EC(x) = \underset{y}{argmax} \sum_{i=1}^{m} w_i \cdot I\left(h_{b_i}(x) = y\right) \tag{2}$$

where $I()$ is an indicator function and w_i is a weight of $h_{b_i}(x)$.

Comparing hashing approaches to standard sampling methods lowers the computing cost by enabling scalable and effective sampling of the X_{maj}, even in high-dimensional datasets. Hashing ensures that sampled subsets are diverse, producing a more resilient ensemble. Each base classifier in the ensemble is trained using a marginally different dataset to encourage model variety. Because hash-based undersampling eliminates redundant data and ensures the model learns from more balanced input, it lowers the chance of overfitting to the X_{maj}. The hash function selection has a significant impact on the method's performance. Inadequate selection of hash methods can result in the grouping of incompatible data points, which can lower classification performance and cause suboptimal sampling. The approach can cause overfitting to the X_{min} because it always keeps the complete X_{min}. This is especially true if there are very few minority cases in comparison to the X_{maj}. Hashing shrinks the quantity of the dataset and could lead some models to ignore intricate correlations between the features in favour of concentrating exclusively on simpler patterns within the data.

C. AdaBoost

The main idea behind AdaBoost is that it focuses on the data points misclassified by previous classifiers and adjusts their weights. Hence, the subsequent classifiers pay more attention to these difficult examples. AdaBoost is an adaptive algorithm because each new classifier is trained with respect to the performance of those already trained. It is widely used for binary classification tasks but can also be extended to multiclass problems. AdaBoost (Adaptive Boosting) is

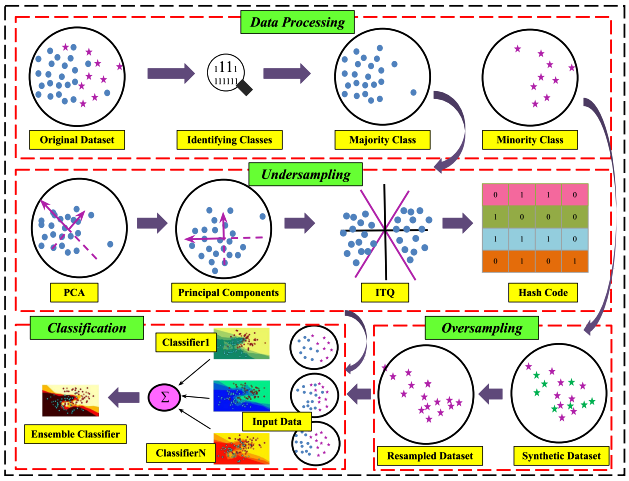


FIGURE 1. Proposed SMOTEHashBoost method for class imbalance problem.

a powerful ensemble learning technique primarily used to improve the performance of weak classifiers [27]. Following are the steps of AdaBoost:

1) Initialize weights ($w_i^{(1)}$): Dataset D contains the feature vector x_i and corresponding binary class label y_i and then $w_i^{(1)} = 1/N$ for each x_i.
2) Train weak classifier ($wc_t(x)$): In each iteration t, $wc_t(x)$ is trained on weighted D to minimize the error ϵ_t and can be computed as

$$\epsilon_t = \sum_{i=1}^{N} w_i^{(t)} \cdot I\left(wc_t(x_i) \neq y_i\right) \tag{3}$$

where $I()$ returns 1 if the classifier misclassifies x_i, and 0 otherwise.
3) Compute classifier weight (α_t): After training $wc_t(x)$, calculate its α_t, which measures the classifier's importance. This weight is given by:

$$\alpha_t = \frac{1}{2} ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \tag{4}$$

The α_t increases as the ϵ_t decreases, giving more importance to more accurate classifiers.
4) Update weights: The $w_i^{(t)}$ of the misclassified x_i are increased to emphasize the difficult examples for the next $wc_{t+1}(x)$. The new $w_i^{(t+1)}$ for each x_i are updated with normalization factor Z_t as:

$$w_i^{(t+1)} = \frac{w_i^{(t)}}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } I\left(wc_t(x_i) = y_i\right) \\ e^{\alpha_t} & \text{if } I\left(wc_t(x_i) \neq y_i\right) \end{cases} \tag{5}$$

5) Final classifier ($WC(x)$): The $WC(x$ is a weighted sum of the $wc_t(x)$, given by:

$$WC(x) = sign\left(\sum_{t=1}^{T} \alpha_t wc_t(x)\right) \tag{6}$$

AdaBoost remarkably enhances performance by turning weak learners into powerful classifiers with only marginally

Algorithm 1 SMOTEHashBoost: Ensemble Algorithm for Imbalanced Dataset Pattern Classification

Input: Base estimator $wc_t()$, sampling method s, number of iterations n_{iter}, number of neighbors k, number of estimators n_{est}
Output: Ensemble classifier WC

Initialize AdaBoost classifier with base estimator $wc_t()$ and n_{est}
Determine X_{maj} and X_{min} classes and their sizes
Perform SMOTE on X_{min} to generate X_{syn}
Set k to min(k, number of X_{min} - 1) if possible
Compute number of bits n_{bits} using Eq. 8
Perform PCA on X_{maj} to determine $X_{PCA} \in \mathbb{R}^{n_{maj} \times n_{bits}}$
Compute rotation matrix R using ITQ on X_{PCA}
Generate H for X_{maj} samples using sign($X_{PCA} \cdot R$)
for each subspace in $2^{n_{bits}}$ **do**
    Select a $X_{sub} \subset X_{maj}$ based on sampling method s
    Combine X_{min}, X_{syn} and X_{sub} samples to form training set (X', y')
    Train base classifier $wc_t()$ on (X', y')
end for
return Ensemble classifier WC using Eq. 6

better ACC than random guessing. Each iteration places greater emphasis on cases that are challenging to categorize, which aids in the efficient management of noisy and complicated data distributions. Compared to other ensemble approaches, AdaBoost is comparatively easy to use because it does not involve tweaking many hyperparameters. Decision trees, perceptrons, and other weak learners can all benefit from its application. AdaBoost is less prone to overfitting than other ensemble techniques, particularly when utilizing basic weak learners like decision stumps. Due to its tendency to overemphasize noisy or erroneous samples by giving misclassified cases higher weights, AdaBoost might be unreliable when dealing with noisy data or outliers. To achieve high performance, the technique may require a significant number of iterations or boosting rounds, which could raise the cost of computing and complicate the model. AdaBoost relies on strengthening weak learners; hence, it may only offer considerable performance gain if the weak classifiers are simple and perform nearly randomly. Although it is typically less computationally demanding than techniques like random forests or gradient boosting, training numerous weak classifiers can be computationally demanding, particularly for large datasets [25], [26].

D. SMOTEHashBoost METHOD

The **SMOTEHashBoost** methodology is designed to address the class imbalance by integrating SMOTE oversampling, hash-based undersampling, and AdaBoost into a cohesive ensemble learning framework, as shown in figure 1. The SMOTEHashBoost steps are shown in Algorithm 1. This approach ensures that both X_{min} and X_{maj} classes are represented in the training process while focusing on difficult-to-classify instances through boosting. The methodology is divided into three primary phases: balancing the dataset, hash-based undersampling, and ensemble learning with AdaBoost.

1) DATASET BALANCING

Imbalanced datasets are characterized by a disproportionately small number of X_{min} samples, which can cause classifiers to favor the X_{max}. Given a dataset D and $y_i \in \{0, 1\}$ represents the class labels (with 0 for the X_{maj} class and 1 for the X_{min} class), the SMOTE algorithm is employed to address this issue by artificially generating new X_{min} class samples. For each $x_i \in X_{min}$, synthetic samples X_{syn} are generated by Eq. 1. The result is a balanced dataset (X_{res}, y_{res}),

$$X_{res} = X_{maj} \cup X_{min} \cup X_{syn} \tag{7}$$

and y_{res} represents the corresponding labels. This balanced dataset is then used for the next phase, ensuring that both classes contribute equally to the learning process.

2) HASH-BASED UNDERSAMPLING

While SMOTE addresses the under-representation of the X_{min}, the X_{maj} may still overwhelm the classifier due to its large size. Thus, hash-based undersampling is applied to reduce the number of X_{maj} class samples while maintaining diversity across the feature space. Thus, following are the steps:

- Dimensionality reduction: Let $X_{maj} \in \mathbb{R}^{n_{maj} \times d}$ represent the X_{maj} samples, where # X_{maj} samples (n_{maj}) $\gg$ # X_{min} samples (n_{min}). Several studies have demonstrated that principal component analysis (PCA) provides good generalization ability and performs successive projection directions while retaining a greater amount of data variance. The variance allows learning of different patterns present in the data to improve ACC [33], [34], [35]. Thus, in this work, PCA [36] was applied to reduce dimensionality, denoted by n_{bits}. It is calculated dynamically based on the ratio of X_{maj} to X_{min} samples and given as:

$$n_{bits} = min\left(\left\lceil log_2\left(3 \cdot \frac{n_{maj}}{n_{min}}\right)\right\rceil, d\right) \tag{8}$$

  The reduced majority samples are denoted as $X_{PCA} \in \mathbb{R}^{n_{maj} \times n_{bits}}$. This step ensures that only the most informative features are retained, making subsequent undersampling more efficient while preserving the structure of the data.
- Hashing: Further, Iterative Quantization (ITQ) has been successfully utilized in various studies [34], [37] to reduce the quantization error resulting from the orthogonal transformation of PCA, i.e., dimensionality reduction, and improve the system's overall performance. Thus, in this work, once dimensionality reduction is performed, ITQ bib;34 is applied to generate binary hash codes for the X_{maj} samples. ITQ finds an optimal rotation matrix to minimize quantization error, ensuring that samples with similar characteristics are mapped to

the same hash subspaces. These hash codes partition the X_{maj} into distinct regions, enabling more structured and representative samples. Let $R \in \mathbb{R}^{n_{bits} \times n_{bits}}$ be a rotation matrix that minimizes quantization error:

$$R^* = \underset{R}{argmin} \, \|X_{PCA} - sign(VR)\|_F^2 \tag{9}$$

where $X_{PCA} \in \mathbb{R}^{n_{maj} \times n_{bits}}$ is the PCA-transformed data. The majority samples are now represented by binary hash codes $H = \text{sign}\,(X_{PCA} \cdot R)$.

- Hashing subspaces: From each hash subspace $H_k \subset H$, a subset of $X_{sub} \subset X_{maj}$ samples is selected according to a predefined sampling method. The goal is to ensure that the selected X_{sub} samples are diverse and representative of the overall distribution while avoiding over-representation from any specific subspace. The combination of dimensionality reduction, hashing, and weighted sampling ensures that the X_{maj} is undersampled in a balanced and efficient manner. The final result is a balanced training dataset (X', y'), which includes all X_{min} samples (real and synthetic) and a representative subset of X_{maj} samples from different regions of the feature space.

$$X' = X_{sub} \cup X_{min} \cup X_{syn} \tag{10}$$

3) BOOSTING WITH AdaBoost

After creating a (X', y') dataset, AdaBoost is applied to iteratively improve the model's classification performance by focusing on difficult-to-classify instances. This boosting process enhances the predictive power of the ensemble by combining the strengths of multiple $wc_t(x)$.

The proposed method is summarized as:

- Balanced ensemble learning: Combining oversampling the X_{min} and undersampling the X_{maj} in a controlled manner, followed by boosting, creates a powerful ensemble learning framework that reduces class imbalance bias and improves predictive ACC.
- Algorithmic robustness: Through the use of PCA and ITQ for dimensionality reduction and hashing, a complex approach to manage the X_{maj} is illustrated. While reducing computational complexity, this method maintains the dataset's representative characteristics.
- Dynamic feature selection: The dynamic determination of the n_{bits} supports more efficient learning without overfitting by ensuring that only the most informative features are used during undersampling.
- Boosted learning for X_{min} representation: The AdaBoost phase ensures that difficult-to-classify instances, often from the X_{min}, are given more attention, improving the classifier's ability to generalize.

Class imbalance can be effectively addressed using SMOTEHashBoost, which focuses on creating balanced datasets and uses boosting to improve classification performance, particularly for X_{min} occurrences.

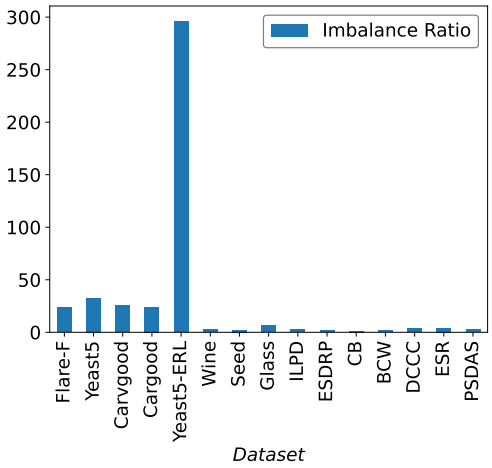


FIGURE 2. Imbalance ratio of various datasets.

III. RESULTS AND DISCUSSIONS

This section provides information about the datasets and experiment results.

A. DATASET DESCRIPTION

The proposed methodology was evaluated on four different categories of datasets, i.e., low dimensionality - high sample size (LDHSS), low dimensionality - low sample size (LDLSS), high dimensionality - low sample size (HDLSS), and high dimensionality - high sample size (HDHSS). Thus, these categories consist of a total of fifteen diverse multivariate datasets. A comparative information of each dataset is provided in table 1. The table offers a thorough analysis of multiple datasets across a number of characteristics, with a primary focus on class imbalance. Although the amount of features varies between datasets and appears to be unrelated to class imbalance, it can affect model performance and complexity. The datasets are from many fields and are either gathered from real-world situations or laboratory settings. To facilitate binary classification, we converted the datasets by identifying the most underrepresented class and designating it as the X_{min} and rest of the other as a X_{maj}. The ratio of the X_{maj} size to the X_{min} size is used in the imbalance ratio to quantify the imbalance, as shown in figure 2. With a very high imbalance ratio (295.8), yeast5-ERL demonstrates the X_{maj}'s dominance. Additionally, flare-F and yeast5 are difficult datasets for classification algorithms due to their large imbalance ratios (23.79 and 32.73, respectively). Predictions made by machine learning models trained on these datasets may be significantly skewed towards the X_{maj}, particularly if the imbalance problem is not addressed. This could result in inaccurate performance metrics like ACC. Figure 2 and table 1 indicates that the proposed work is evaluated on diverse domains with different features and varying imbalance ratios of the datasets.

The classifier's performance is evaluated using AUC-ROC (Area under the Receiver Operating Characteristic Curve)

TABLE 1. Comparative information of different dataset.

Dataset	Domain	Source	# Features	# Classes	# Data samples	Classes and their # data samples	X_{min} label	# X_{min} data samples	# X_{maj} data samples
	Categories: Low Dimensionality - High Sample Size (LDHSS)								
Flare-F [38]	Astronomy/ Astrophysics	Real world	11	2	1066	('negative': 1023), ('positive': 43)	positive	43	1023
Yeast5 [39]	Biology / Bioinformatics	Real world	8	2	1484	('negative': 1440), ('positive': 44)	positive	44	1440
CarvGood [40]	Automotive	Synthetic	6	4	1728	('acc': 384), ('good': 69), ('unacc': 1210), ('vgood': 65)	vgood	65	1663
CarGood [40]	Automotive	Synthetic	6	4	1728	('acc': 384), ('good': 69), ('unacc': 1210), ('vgood': 65)	good	69	1659
Yeast5-ERL [39]	Biology / Bioinformatics	Real world	8	10	1484	('CYT': 463), ('ERL': 5), ('EXC': 35), ('ME1': 44), ('ME2': 51), ('ME3': 163), ('MIT': 244), ('NUC': 429), ('POX': 20), ('VAC': 30)	ERL	5	1479
	Categories: Low Dimensionality - Low Sample Size (LDLSS)								
Wine [41]	Physics and Chemistry	Real world	13	3	178	(0: 59), (1: 71), (2: 48)	2	48	130
Seed [42]	Biology / Bioinformatics	Real world	7	3	210	(1.0: 70), (2.0: 70), (3.0: 70)	2	70	140
Glass [43]	Physics and Chemistry	Synthetic	9	6	214	(1.0: 70), (2.0: 76), (3.0: 17), (5.0: 13), (6.0: 9), (7.0: 29)	7	29	185
Indian Liver Patient Dataset (ILPD) [44]	Health and Medicine	Real world	10	2	583	(1.0: 416), (2.0: 167)	2	167	416
	Categories: High Dimensionality - Low Sample Size (HDLSS)								
Early Stage Diabetes Risk Prediction (ESDRP) [45]	Health and Medicine	Real world	16	2	520	Positive: 320, Negative: 200	Negative	200	320
Connectionist Bench (CB) [46]	Physics and Chemistry	Synthetic	60	2	208	Mine: 111, Rock: 97	Rock	97	111
Breast Cancer Wisconsin (Diagnostic) (BCW) [47]	Health and Medicine	Real world	30	2	569	M: 212, B: 357	M	212	357
	Categories: High Dimensionality - High Sample Size (HDHSS)								
Default of Credit Card Clients (DCCC) [48]	Business	Real world	23	2	30000	Yes (6636), No (23364)	Yes	6636	23364
Epileptic Seizure Recognition (ESR)	Healthcare	Real world	178	2	11,500	Seizure (2,300), Non-seizure (9,200)	Seizure	2,300	9,300
Predict Students' Dropout and Academic Success (PSDAS) [49]	Social Science	Real world	36	3	4424	Dropout (1421) , Graduate (2209) , Enrolled (794)	Enrolled	794	2209

(AUC abbreviation used for AUC-ROC), F1-score, average precision (AP), and G-Mean performance metrics [36]. They are given as

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

where # instances are accurately signified as ''positive'', i.e., true positives (TP), # instances are incorrectly signified as ''negative,'' i.e., false negatives (FN), and # instances are incorrectly signified as ''positive,'' i.e., false positives (FP).

$$AP = \sum_{a=0}^{n-1} [th_r(a) - th_r(a+1)] \cdot th_p(a) \tag{12}$$

where precision $th_p(a)$ and recall $th_r(a)$ for threshold a and they are computed for thresholds n - 1.

$$G - Mean = \sqrt{\frac{1}{1 + \frac{FN}{TP}} \times \frac{1}{1 + \frac{FP}{TN}}} \tag{13}$$

where # instances are accurately signified as ''negative'', i.e., true negatives (TN).

The Python[1] programming language has been used to implement the proposed method. Various libraries like scikit-learn, NumPy, matplotlib, etc., have been employed from Python. The code was executed on a Google Colab[2] (Processor: Intel Xeon CPU with 2 vCPUs and RAM: 13 GB). The dataset was divided using Stratified K-Fold Cross-Validation to ensure balanced class distributions across folds [36]. Typically, a 5-fold Stratified K-Fold was used, meaning the data was split into five parts, with four folds used for training and one fold for testing, and this process was repeated across multiple runs, i.e., 20 times. The five nearest neighbors are utilized in the SMOTE. A decision tree serves as the base classifier. For GAN and SMOTified-GAN, the setting given in [50] is adopted for result generation.

B. PERFORMANCE ANALYSIS OF SMOTEHashBoost ON VARIOUS SAMPLING STRATEGIES

The table 2 compares different sampling strategies used in the SMOTEHashBoost algorithm across various datasets. The sampling strategies include reciprocal, random, linearity, negative exponent, and limit, and their effectiveness is measured in terms of ACC (%) for each dataset and evaluated using various metrics.

For LDHSS category datasets, the choice of strategy does not significantly impact ACC. This effect observed due to the abundance of data provides sufficient diversity, reduces sensitivity to individual samples, and adequately represents the data's underlying patterns. Nevertheless, AUC shows sensitivity to the strategy, particularly for mid-performing datasets like flare-F. Concerning ACC, the reciprocal method outperformed the others in the flare-F dataset, with a 94.84% classification ACC with a lower AUC of 0.63, showing modest class distinction. The flare-F dataset experiences more variation in its ACC and AUC values. With a decent AUC of 0.92 and the best ACC of 98.45%, the reciprocal approach performed exceptionally well overall for the yeast5 dataset. Other techniques still needed to improve AUC, yet they were still competitive. The reciprocal, linearity, and negative exponent strategies performed quite well with a nearly ideal AUC of 0.99 and 99.94% ACC for the carvgood dataset. The carvgood dataset experiences less variation in its ACC and AUC values. Concerning the cargood dataset, random followed closely after with 99.19% ACC and 0.97 AUC, while reciprocal yielded the best ACC at 99.42% with an AUC of 0.97. This dataset's worst performance, with an ACC of 99.07% and an AUC of 0.95, was the negative exponent method. Although they both had ACC rates above 99.0%, limit and linearity had comparable performance with moderate AUCs. The random, linearity, and negative exponent strategies performed exceptionally well, with a near-perfect AUC of 0.99 and a 99.93% ACC rate for the yeast5-ERL dataset. The limit and reciprocal methods had an ACC of 99.87%, while the limit strategy's AUC was somewhat lower (0.97).

The wine dataset yielded almost ideal performance for the reciprocal, random, and limit strategies, with an ACC of 99.44% and an AUC of 0.99. With little variations in AUC for the seed dataset (Refer the figure 3(f)), every strategy achieved identically high ACC (98.57%). For the glass dataset, ACC (96.28%) and AUC (0.95 and 0.94, respectively) were obtained using the random and negative exponent methods. With an ACC of 69.3% and an AUC of 0.65, the random method yielded the best results when applied to the ILPD dataset. The reciprocal was marginally less effective, with 68.1% ACC and 0.64 AUC. The limit was the least accurate, with an ACC of 67.4% and an AUC of 0.64. The ILPD dataset experiences more variation in its ACC and AUC values. For LDLSS category datasets, random and negative exponent sampling strategies often yield better results, particularly for AUC. This pattern observed due to their ability for enhancing diversity and adapt to constrained training data. For HDLSS category dataset, ESDRP and BCW dataset demonstrates excellent results (ACC > 94.0%, AUC $\approx$ 0.97) for all strategies, indicating robustness to different sampling approaches. However, CB dataset exhibits significant variation in ACC (75.0 – 82.0%) with different strategies and AUC (0.75 – 0.83), with the random strategy performing best. HDLSS category datasets tend to benefit more from the random strategy due to their tendency to explore diverse feature combinations. ESR dataset achieves high ACC (>91.0%) and AUC (>0.90), with the random strategy being particularly effective. Datasets like DCCC and PSDAS show moderate to low performance across all strategies (ACC < 78.0%, AUC < 0.65). ESR and PSDAS datasets exhibit less variability in performance. HDHSS

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

TABLE 2. Different sampling strategies for SMOTEHashBoost.

Dataset	Sampling stratergies									
	Reciprocal		Random		Linearity		Negative Exponent		Limit	
	ACC (%)	AUC	ACC (%)	AUC	ACC (%)	AUC	ACC (%)	AUC	ACC (%)	AUC
Flare-F	94.84±0.38	0.63±0.02	94.37±4.33	0.62±0.04	94.75±0.53	0.66±0.02	94.75±0.51	0.62±0.03	94.75±0.29	0.63±0.02
Yeast5	98.45±0.23	0.92±0.02	98.38±0.18	0.92±0.03	98.25±0.23	0.94±0.02	98.45±0.19	0.93±0.03	98.32±0.19	0.91±0.02
Carvgood	99.94±0.01	0.99±0.02	99.83±0.12	0.99±0.01	99.94±0.02	0.99±0.01	99.94±0.10	0.99±0.01	99.88±0.01	0.99±0.01
Cargood	99.42±0.21	0.97±0.02	99.19±0.33	0.97±0.03	99.19±0.35	0.96±0.03	99.07±0.29	0.95±0.03	99.13±0.33	0.96±0.03
Yeast5-ERL	99.87±0.03	0.99±0.01	99.93±0.03	0.99±0.01	99.93±0.02	0.99±0.01	99.93±0.03	0.99±0.01	99.87±0.03	0.97±0.03
Wine	99.44±0.29	0.99±0.01	99.44±0.66	0.99±0.01	98.89±0.86	0.99±0.01	98.89±1.08	0.99±0.01	99.44±0.53	0.99±0.01
Seed	98.57±0.71	0.98±0.01	98.57±0.80	0.98±0.01	98.57±0.61	0.98±0.01	98.57±0.85	0.98±0.01	98.57±0.42	0.98±0.01
Glass	94.41±0.93	0.95±0.02	96.28±0.96	0.95±0.02	94.87±1.19	0.92±0.02	96.28±0.81	0.94±0.01	94.88±1.23	0.93±0.02
ILPD	68.10±1.69	0.64±0.02	69.30±1.57	0.65±0.02	67.59±1.58	0.63±0.02	68.26±1.76	0.65±0.02	67.40±1.61	0.64±0.02
ESDRP	97.69±0.69	0.98±0.01	97.50±0.64	0.98±0.01	97.12±0.38	0.97±0.01	97.31±0.48	0.97±0.01	97.12±0.58	0.97±0.01
CB	75.01±2.43	0.75±0.03	82.75±2.61	0.83±0.03	77.44±3.30	0.77±0.04	80.30±3.40	0.81±0.04	75.99±1.99	0.76±0.02
BCW	94.38±0.71	0.94±0.01	95.26±0.53	0.95±0.01	94.38±0.75	0.94±0.01	94.55±0.95	0.94±0.01	94.55±0.95	0.95±0.01
DCCC	71.38±0.18	0.63±0.01	76.64±0.21	0.64±0.01	71.53±0.23	0.62±0.01	71.66±0.28	0.63±0.02	71.51±0.23	0.62±0.01
ESR	91.15±0.26	0.91±0.01	93.46±0.19	0.94±0.01	91.29±0.20	0.91±0.01	91.15±0.20	0.91±0.01	91.09±0.22	0.91±0.01
PSDAS	77.26±0.32	0.63±0.01	75.84±0.53	0.64±0.01	78.62±0.23	0.64±0.03	75.93±0.65	0.65±0.01	75. 79±0.60	0.64±0.01

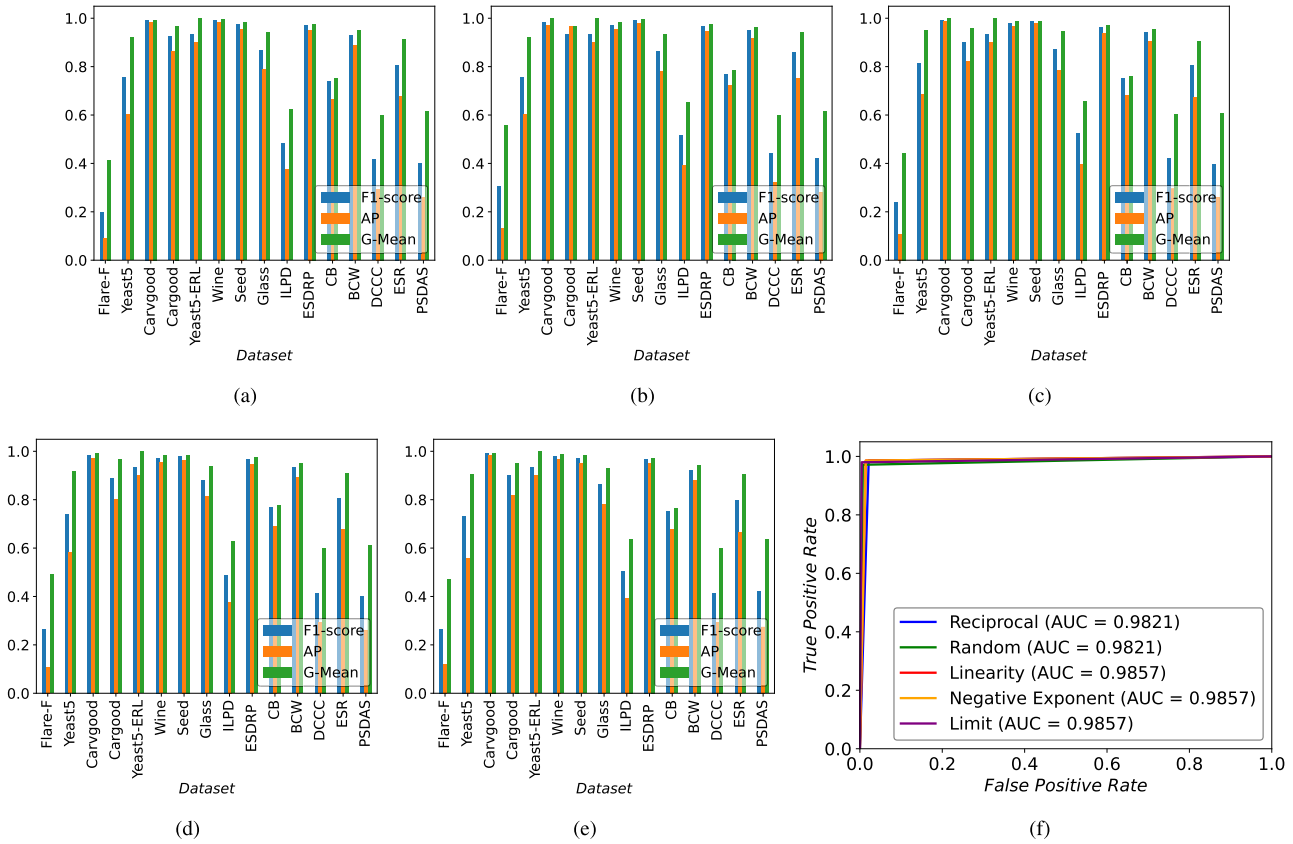


FIGURE 3. The F1-score, AP, and G-Mean for various sampling strategies of the SmoteHashBoost method: (a) Reciprocal, (b) Random, (c) Linearity, (d) Negative Exponent, and (e) Limit. (f) AUC curve of various sampling strategies of the SMOTEHashBoost method on the seed dataset.

category datasets are less sensitive to strategy changes, especially for moderately performing datasets. However, random yields slight improvements in both metrics.

In terms of ACC and AUC, the reciprocal method consistently performs well across datasets, particularly for datasets belonging to the LDHSS and LDLSS category. Thus, it is acceptable because of its strength in offering high classification ACC and appropriate class separation with less variation in the performance. The random technique can produce large AUC values, showing its effectiveness in class distinction, but it often exhibits slightly poorer ACC than the reciprocal strategy for LDHSS and LDLSS category datasets. Random strategy performs better on datasets belonging to the HDLSS and HDHSS category than other techniques, indicating that it may be better suited for complicated datasets with unequal class distributions. As seen in the flare-F and yeast5 datasets, the linearity technique typically yields the best AUC even if it frequently achieves slightly lower

TABLE 3. Comparison with various state-of-the-art techniques.

Dataset		SMOTEHashBoost	HUE	RUSBoost	Adaboost	SMOTE-Tomek	SMOTE-ENN	Borderline-SMOTE	ADASYN	GAN	SMOTified-GAN
Flare-F	ACC (%)	94.37±4.33	81.43±0.59	82.55±0.70	94.94±2.26	94.09±0.44	93.34±0.72	94.65±0.45	94.56±0.54	95.58±0.7	95.93±0.71
	AUC	0.62±0.04	0.86±0.02	0.83±0.02	0.55±0.13	0.57±0.02	0.72±0.03	0.62±0.02	0.61±0.03	0.95±0.02	0.94±0.02
Yeast5	ACC (%)	98.38±0.18	95.96±0.27	97.03±0.24	98.52±0.23	98.31±0.43	98.18±0.23	98.38±0.29	98.59±0.23	96.77±0.93	96.23±1.88
	AUC	0.92±0.03	0.97±0.01	0.97±0.01	0.85±0.02	0.91±0.03	0.96±0.03	0.93±0.03	0.93±0.03	0.95±0.02	0.97±0.02
Carvgood	ACC (%)	99.83±0.12	98.44±0.03	99.02±0.17	99.94±0.02	99.94±0.01	99.88±0.12	99.88±0.01	99.88±0.10	94.78±0.86	93.14±1.77
	AUC	0.99±0.02	0.99±0.01	0.99±0.01	0.99±0.01	0.99±0.01	0.99±0.01	0.99±0.01	0.98±0.01	0.99±0.01	0.99±0.01
Cargood	ACC (%)	99.19±0.33	95.49±0.28	99.19±0.30	99.88±0.08	99.48±0.23	98.73±0.33	99.31±0.31	99.25±0.32	94.25±1.74	93.09±0.97
	AUC	0.97±0.03	0.97±0.01	0.98±0.01	0.99±0.01	0.96±0.02	0.95±0.2	0.95±0.02	0.96±0.03	0.99±0.01	0.98±0.01
Yeast5-ERL	ACC (%)	99.93±0.03	99.46±0.46	99.87±0.04	99.93±0.02	99.8±0.02	99.93±0.06	99.93±0.02	N.A.	N.A.	N.A.
	AUC	0.99±0.03	0.99±0.01	0.99±0.01	0.99±0.01	0.99±0.01	0.99±0.01	0.99±0.01	N.A.	N.A.	N.A.
Wine	ACC (%)	99.44±0.66	94.43±1.22	99.43±0.96	98.89±1.03	98.3±0.84	97.21±1.51	98.89±1.02	98.89±1.07	93.89±1.60	96.67±1.9
	AUC	0.99±0.02	0.98±0.01	0.99±0.01	0.98±0.02	0.98±0.02	0.97±0.02	0.98±0.02	0.98±0.02	0.96±0.04	0.97±0.03
Seed	ACC (%)	98.57±0.80	98.57±0.58	98.10±0.65	90.48±1.80	98.57±1.05	98.57±1.26	98.57±1.02	98.1±1.35	87.14±2.93	86.31±3.85
	AUC	0.98±0.01	0.98±0.01	0.98±0.01	0.89±0.02	0.98±0.02	0.98±0.02	0.98±0.01	0.98±0.02	0.98±0.01	0.98±0.01
Glass	ACC (%)	96.28±0.96	94.88±0.94	97.21±1.09	99.53±0.44	96.74±0.74	96.27±0.95	96.28±1.14	96.74±1.01	56.74±7.13	55.47±6.84
	AUC	0.95±0.02	0.95±0.01	0.94±0.02	0.99±0.01	0.93±0.02	0.94±0.02	0.93±0.02	0.94±0.01	0.89±0.02	0.89±0.02
ILPD	ACC (%)	69.30±1.57	66.55±1.42	71.52±0.93	67.92±1.89	69.12±1.40	68.63±1.62	69.48±1.72	67.76±1.79	67.52±11.17	70.09±9.99
	AUC	0.65±0.02	0.71±0.02	0.66±0.02	0.62±0.03	0.63±0.02	0.69±0.02	0.65±0.02	0.64±0.03	0.74±0.05	0.76±0.05
ESDRP	ACC (%)	97.50±0.64	97.19±0.79	98.65±0.50	97.50±0.59	97.50±0.68	93.27±1.19	97.12±0.72	97.69±0.63	91.97±2.00	92.98±1.09
	AUC	0.97±0.01	0.97±0.01	0.99±0.01	0.97±0.01	0.97±0.01	0.94±0.01	0.97±0.01	0.98±0.01	0.98±0.01	0.99±0.01
CB	ACC (%)	82.75±2.61	78.81±2.37	82.26±2.09	76.42±1.99	78.37±1.95	75.01±2.16	77.44±2.57	N.A.	86.79±2.50	87.62±3.67
	AUC	0.83±0.03	0.78±0.03	0.72±0.02	0.76±0.02	0.78±0.02	0.74±0.02	0.77±0.03	N.A.	0.96±0.01	0.96±0.01
BCW	ACC (%)	95.26±0.53	96.11±0.74	97.19±0.47	94.03±0.81	94.38±0.99	94.38±0.77	94.03±0.76	94.38±1.02	89.78±6.45	94.17±2.72
	AUC	0.95±0.01	0.96±0.01	0.97±0.01	0.94±0.01	0.94±0.01	0.94±0.01	0.94±0.01	0.94±0.02	0.99±0.01	0.99±0.01
DCCC	ACC (%)	76.64±0.21	68.66±0.20	N.A.	72.99±0.16	72.49±0.25	70.52±0.22	72.45±0.27	72.27±0.23	80.89±2.47	81.68±2.12
	AUC	0.64±0.01	0.68±0.01	N.A.	0.62±0.02	0.62±0.01	0.65±0.01	0.62±0.01	0.62±0.01	0.94±0.01	0.95±0.01
ESR	ACC (%)	93.46±0.19	95.55±0.18	N.A.	94.18±0.20	91.81±0.21	91.72±0.36	91.93±0.22	89.95±0.27	95.89±0.54	95.84±0.49
	AUC	0.94±0.01	0.94±0.01	N.A.	0.91±0.01	0.89±0.01	0.89±0.01	0.89±0.01	0.89±0.01	0.97±0.01	0.97±0.01
PSDAS	ACC (%)	75.84±0.53	74.25±0.50	N.A.	75.72±0.47	75.99±0.49	72.67±0.71	75.99±0.62	76.11±0.50	63.80±6.81	64.46±9.93
	AUC	0.64±0.01	0.71±0.01	N.A.	0.65±0.01	61.81±0.01	0.65±0.01	0.62±0.01	0.64±0.01	0.82±0.03	0.82±0.04

ACC than other strategies. The linearity approach effectively enhances the model's class discrimination capabilities, which is advantageous when class differentiation is more important than ACC. Regarding ACC and AUC, the negative exponent method performs quite consistently across datasets, frequently matching linearity's performance. Although there are better performers than it, in situations such as the carvgood dataset, it provides a good compromise between ACC and class separation. Also, it perform better than linearity strategy for the datasets belonging to the HDLSS and HDHSS category. The performance of the limit approach varies. It outperforms other best practices in some datasets (like wine), but ranks lower in others (like ILPD). It frequently performs more inconsistently, particularly regarding AUC values, suggesting that it might not always offer the best class discrimination. Low correlation was observed in datasets like ILPD and DCCC, suggesting that ACC alone does not reflect the model's ability to discriminate between classes in imbalanced datasets. HDLSS datasets benefit the most from random strategies, indicating their robustness for complex datasets. Datasets with near-perfect baseline performance (e.g., Carvgood, Yeast5-ERL) show negligible impact from different strategies, indicating inherent dataset characteristics outweigh sampling influence. The performance of the proposed method significantly varies with high-dimensional datasets compared to low-dimensional datasets depending on the representation of both majority and minority classes. High-dimensional datasets tend to exhibit more sensitivity to sampling strategies, especially when sample sizes are small (e.g., CB and BCW).

AUC is more sensitive to the choice of sampling strategy than ACC, highlighting the need to prioritize sampling techniques for balanced performance. Across most datasets, the random strategies performed well in terms of both ACC and AUC, particularly in the wine, carvgood, and yeast5-ERL datasets. The linearity strategy tended to produce slightly lower ACC but often resulted in higher AUC values, particularly in datasets belonging to the HDLSS and HDHSS category, suggesting better performance in distinguishing between classes. Similarly, reciprocal stratergy slightly produces better results than random strategy for the dtatasets belonging to LDHSS and LDLSS category. On some datasets, such as carvgood and glass, the negative exponent and limit strategies nonetheless performed competitively, despite having lower accuracies and AUC values overall. Performance differed according to the dataset, showing that the hashing algorithm selected can have a big influence on the result in terms of classification ACC and AUC. SMOTEHashBoost's research of these hashing algorithms shows that while linearity primarily focuses on improving class separability, reciprocal and random do well overall.

The results of diverse sampling procedures for the proposed method, i.e., SMOTEHashBoost, across multiple data sets, are shown in figure 3. The three main metrics used to assess performance are F1-Score, AP, and G-Mean. SMOTEHashBoost consistently performs well across all metrics and sampling schemes, demonstrating its effectiveness for these datasets. Across most datasets, G-Mean is relatively stable compared to F1 and AP scores, suggesting good handling of class imbalances. Across most datasets, when F1-Score and G-Mean are high, the AP tends to follow the same pattern. It suggests that for the majority of datasets, maximizing the F1-score and G-Mean simultaneously is likely to result in higher precision across all recall levels. Random sampling

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

Dataset	SMOTEHashBoost	HUE	RUSBoost	Adaboost	SMOTE-Tomek	SMOTE-ENN	Borderline-SMOTE	ADASYN	GAN	SMOTified-GAN	SMOTEBoost	HashBoost
Flare-F	41	17	337	1	2	2	1	25	193	195	37	1
Yeast5	9	12	101	1	2	2	1	2	238	240	2	1
Carvgood	5	5	89	2	2	2	1	2	309	311	2	1
Cargood	6	7	123	1	2	2	1	2	309	306	2	1
Yeast5-ERL	3	22	31	1	1	1	1	N.A.	N.A.	N.A.	1	1
Wine	1	1	10	14	11	2	14	1	110	119	1	1
Seed	1	1	8	1	1	1	1	1	101	103	1	1
Glass	1	2	58	2	1	1	1	1	113	115	1	1
ILPD	3	2	183	1	1	1	1	2	132	133	1	1
ESDRP	1	1	59	1	2	2	2	1	137	142	1	1
CB	1	1	51	1	2	2	2	N.A.	105	110	1	1
BCW	2	2	235	1	3	3	3	2	136	136	1	1
DCCC	325	190	N.A.	35	205	240	330	286	3251	3251	55	12
ESR	1481	827	N.A.	795	2269	37499	371	1853	1197	1197	1095	49
PSDAS	45	13	N.A.	2	36	41	26	15	476	476	7	1

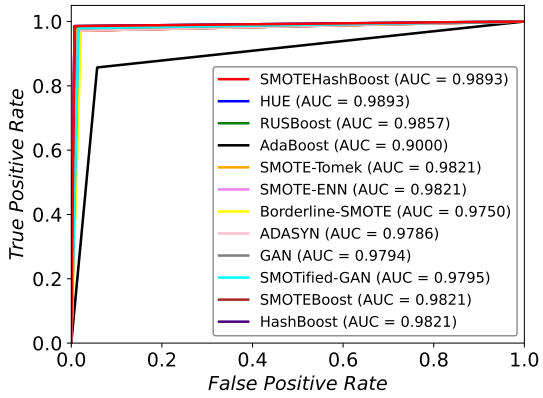


FIGURE 4. ROC curve for seed dataset for various state-of-the-art techniques.

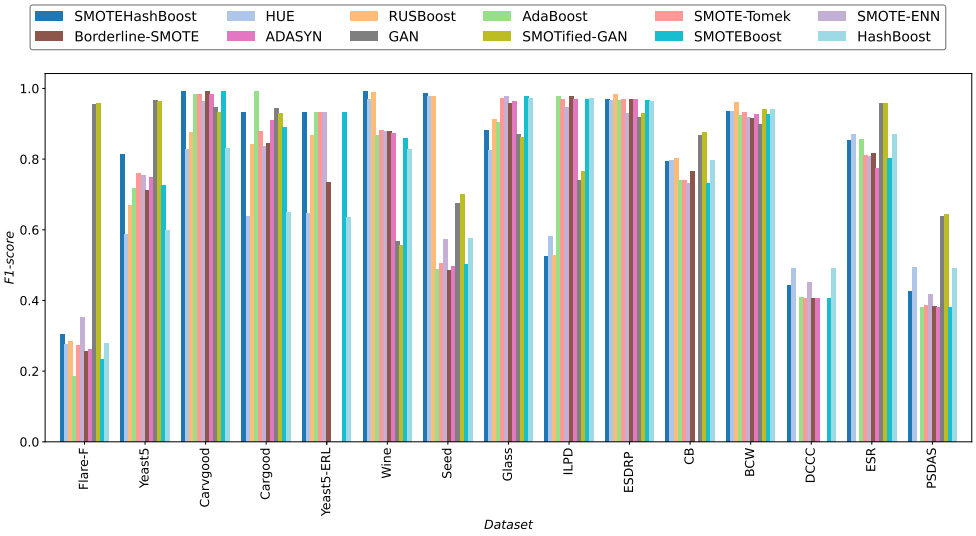
shows better generalization across datasets, particularly for high-performing datasets like wine and carvgood. The ILPD, DCCC, PSDAS and flare-F datasets seem highly sensitive to the sampling strategy employed. Some datasets, particularly glass, yeast5, CB, ESR, and cargood show variability in performance depending on the sampling strategy. As a result, choosing the appropriate sampling procedure have a prominent effect. For these datasets, linearity and random sampling procedures typically yield high results, whereas reciprocal and negative exponent sampling strategies may result in lower ratings. All things considered, random sampling yields the most reliable findings across a variety of datasets, although more specialised approaches like reciprocal or linearity might be helpful in particular situations like wine, carvgood, and glass. The type and degree of class imbalance in the dataset affect SMOTEHashBoost's overall efficacy.

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

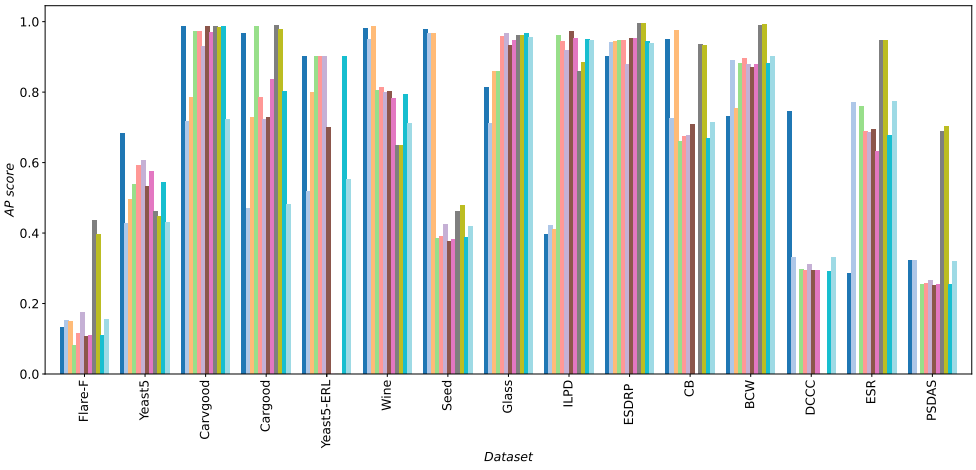
The table 3 and figure 4 comprehensively compares distinct classification methods—SMOTEHashBoost, HUE [32], RUSBoost [8], Adaboost [27], SMOTE-Tomek [51], [52], SMOTE-ENN [52], Borderline-SMOTE [53], ADASYN [31], GAN [50] and SMOTified-GAN [50]—assessed across multiple datasets. RUSBoost does not apply to time series datasets like DCCC, ESR, and PSDAS, as it loses significant information due to a reduction in majority classes [2]. Additionally, ADASYN, GAN, and SMOTified-GAN are unsuitable for Yeast5-ERL and CB datasets as no neighbors belong to the majority class.

Both GAN and SMOTified-GAN demonstrate the best performance in terms of ACC (95.93%) alongside AUC (0.94–0.95), which shows they are suitable for flare-F dataset analysis. Similarly, they dominate for CB, DCCC, and ESR datasets. The majority of techniques demonstrate excellent performance on yeast5 as well as yeast5-ERL through their high ACC rates surpassing 98.0%. The analysis of AUC values indicates that predictive powers remain in a similar range between all techniques, from 0.97 to 0.99. Adaboost reaches the highest ACC level of 99.88%, while all studied techniques demonstrate excellent performance levels above 99.0% for both carvgood and cargood datasets. SMOTEHashBoost obtains 99.44% ACC when used on the wine dataset, though the performance of GAN and SMOTified-GAN falls below 93.0–96.0% ACC. GAN and SMOTified-GAN experience a major reduction in performance for the seed dataset, which results in ACC falling to 86.0–87.0%. The combination of GAN and SMOTified-GAN fails to deliver satisfactory results (ACC 55.0%), thus proving it is unusable with the glass dataset. All techniques demonstrate inferior performance levels compared to other datasets when using ILPD as the test set, which indicates strong classification challenges. RUSBoost demonstrates superior performance than other techniques on ESDRP and BCW data collection points. ADASYN demonstrates superior performance than other methods in the PSDAS dataset.

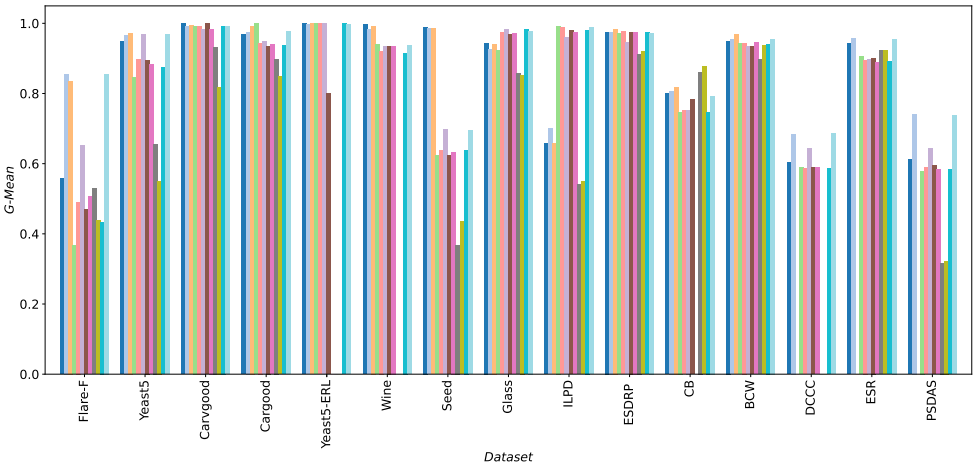
The SMOTified-GAN and GAN demonstrates superior performance on most datasets with ACC outcomes and achieves secondary best results for AUC performance after GAN and SMOTified-GAN. However, the computational requirements for GANs and SMOTified-GANs are higher compared to other methods. Some experiments show that



(a)



(b)



(c)

FIGURE 5. The performance metrics comparison of various state-of-the-art techniques: (a) F1-score, (b) AP score, (c) G-Mean score.

Adaboost and SMOTE-Tomek accomplish ACC results that are on par with other models. Among all methods, HUE demonstrates inferior performance in both ACC and AUC measurements. Data-dependent barriers affect the optimal results from GAN and SMOTified-GAN since they show diminished performance on the Glass dataset. The consistency level of Adaboost and SMOTE-based methods extends to various datasets.

SMOTEHashBoost shows reliable performance in dataset testing because it produces the highest ACC score in many cases and places a close second in the rest. AUC score evaluation shows the strong discriminatory ability of these models. The results using Flare-F and ILPD datasets show inconsistent performance statistics regarding SMOTEHashBoost. SMOTEHashBoost delivers the optimal ACC levels, among other techniques, for the Yeast5-ERL, Wine, and Seed datasets. The model consistently achieves high positions among the top three in most datasets as it demonstrates consistent performance across diverse tasks. The AUC metrics of SMOTEHashBoost demonstrate healthy competition, although they fail to match GAN and RUSBoost performance when dealing with highly imbalanced datasets, including Flare-F and ILPD. SMOTEHashBoost delivers 0.99-area under receiver operating characteristic (AUC) in its tests of Carvgood Cargood, Yeast5-ERL Wine, and Seed tasks while demonstrating powerful classification functionality. The datasets Carvgood, Cargood, and Wine demonstrate almost flawless ACC because their class separability is well-established.

Electing GAN-based methods delivers the most success in extremely imbalanced datasets, although Boosting methods, including Adaboost and SMOTEHashBoost, achieve superior performance on structured datasets. A dataset with clear separability represents an excellent situation for SMOTE-HashBoost to succeed. The fusion techniques ADASYN and SMOTified-GAN deliver minimal enhancement for datasets with medium class distribution inequality.

The F1-Score and AP results for Flare-F Yeast5 and Carvgood datasets reach exceptional levels using SMOTified-GAN and GAN, as shown in figure 5. All datasets demonstrate good performance using Borderline-SMOTE, SMOTE-ENN, and SMOTE-Tomek techniques. The outcome of both SMOTEHashBoost and SMOTEBoost shows balanced performance. The G-Mean values of SMOTEHashBoost remain above 0.94 in most datasets, demonstrating excellent sensitivity and specificity balance. AdaBoost demonstrates average success rates, but it performs poorly on Flare-F. The G-Means of SMOTE-ENN and SMOTE-Tomek techniques stay at high levels, indicating that both methods effectively regulate sensitivity and specificity values. The SMOTEHashBoost imputation method ranks as a leader in imbalanced dataset resampling strategies, achieving superior results than traditional techniques. The technique demonstrates strong performance in F1-Scores and G-Means, which means it optimally balances data sets while retaining essential information. It is a good option for applications needing sensitivity to X_{min} since it excels at striking a balance between precision and recall. SMOTEHashBoost emerges as the best general-purpose model for class imbalance, consistently performing consistently across most datasets and metrics.

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

The execution time of different machine learning techniques measured in seconds across multiple datasets appears in a table 4. The execution time for AdaBoost remains the fastest throughout most datasets, so it becomes highly efficient in computations. GAN-based models (GAN and SMOTified-GAN) require the longest run times when processing HDHSS datasets. Implementing GAN-based methods (GAN and SMOTified-GAN) involves high computation costs while potentially generating better synthetic data. SMOTE-Tomek, SMOTE-ENN, Borderline-SMOTE, and ADASYN hybrid models and their counterparts demonstrate typical execution times ranging from 1-3 seconds but show prolonged execution for datasets with HDHSS. In terms of efficiency, SMOTE-ENN operates as the slowest of all methods built on SMOTE. The execution time of RUSBoost exceeds that of other boosting methods since it operates at a range of 10 to 100 times slower. Among the listed methods, SMOTEHash-Boost and HUE perform moderately by surpassing AdaBoost in execution time but remaining quicker than GAN-based strategies. The execution time of SMOTEHashBoost aligns well with its data effectiveness level. SMOTEHashBoost operates within the same or marginally higher time range as standard SMOTE algorithms. The execution times of SMOTEHashBoost indicate good adaptability with LDLSS, HDLSS, and LDHSS data sets. The SMOTEHashBoost algorithm encounters larger execution time increases when processing the HDHSS dataset. SMOTEHashBoost demonstrates high efficiency and scalability for synthetic data augmentation through its superior execution speeds compared to GAN-based oversampling. SMOTEHashBoost offers an efficient data balancing solution that maintains effective results without being the swiftest technique for imbalanced data handling.

E. ABLATION ANALYSIS

An ablation study uses the figure 5 and tables 4 and 5 to evaluate performance differences between SMOTEHash-Boost, SMOTEBoost [54], and HashBoost through multiple datasets. The accuracy results from SMOTEHashBoost exceed those of SMOTEBoost and HashBoost for almost every dataset tested. The performance evaluation of SMOTE-HashBoost displays strong indicators across all datasets except for minor variations in AUC scores. HashBoost shows weaker ability than both SMOTEHashBoost and SMOTEBoost in several datasets since it demonstrates lower

TABLE 5. Ablation analysis.

Dataset		SMOTEHashBoost	SMOTEBoost	HashBoost
Flare-F	ACC (%)	94.37±4.33	93.93±0.43	80.96±0.65
	AUC	0.62±0.04	0.61±0.02	0.84±0.02
Yeast5	ACC (%)	98.38±0.18	97.52±0.19	95.96±0.23
	AUC	0.92±0.03	0.89±0.02	0.97±0.01
Carvgood	ACC (%)	99.83±0.12	99.78±0.16	98.38±0.04
	AUC	0.99±0.02	0.99±0.01	0.99±0.01
Cargood	ACC (%)	99.19±0.33	99.13±0.29	95.72±0.40
	AUC	0.97±0.03	0.95±0.02	0.97±0.01
Yeast5-ERL	ACC (%)	99.93±0.03	99.23±0.24	98.85±0.58
	AUC	0.99±0.03	0.99±0.01	0.99±0.01
Wine	ACC (%)	99.44±0.66	98.33±1.26	98.33±1.21
	AUC	0.99±0.02	0.98±0.02	0.98±0.02
Seed	ACC (%)	98.57±0.80	98.05±0.77	97.62±0.90
	AUC	0.98±0.01	0.98±0.01	0.98±0.01
Glass	ACC (%)	96.28±0.96	96.28±0.97	93.89±0.82
	AUC	0.95±0.02	0.94±0.02	0.92±0.01
ILPD	ACC (%)	69.30±1.57	68.78±1.80	62.44±1.30
	AUC	0.65±0.02	0.63±0.02	0.60±0.02
ESDRP	ACC (%)	97.50±0.64	97.19±0.82	96.92±1.00
	AUC	0.97±0.01	0.97±0.01	0.97±0.01
CB	ACC (%)	82.75±2.61	76.91±1.80	77.89±3.02
	AUC	0.83±0.03	0.77±0.02	0.79±0.03
BCW	ACC (%)	95.26±0.53	94.73±0.89	94.73±0.48
	AUC	0.95±0.01	0.95±0.01	0.97±0.01
DCCC	ACC (%)	76.64±0.23	72.69±0.20	64.98±0.20
	AUC	0.64±0.01	0.62±0.01	0.63±0.01
ESR	ACC (%)	93.46±0.19	91.58±0.28	92.93±0.14
	AUC	0.94±0.01	0.89±0.01	0.94±0.02
PSDAS	ACC (%)	75.84±0.53	75.24±0.52	68.69±0.44
	AUC	0.64±0.01	0.63±0.01	0.64±0.01

accuracy results in these cases. In most cases, the AUC results of SMOTEHashBoost match those of SMOTEBoost. The highest accuracy gain is seen in the CB dataset, where SMOTEHashBoost outperforms SMOTEBoost by 5.84% and HashBoost by 13.41% in the flare-F dataset. The Carvgood dataset exhibits the lowest accuracy gain among all testing datasets since each method delivers similar results. The stability of SMOTEHashBoost appears higher due to its lower standard deviations in multiple instances. The accuracy of the HashBoost solution presents unstable characteristics due to its significant accuracy fluctuations. The ablation study reveals important information about how SMOTE and HashBoost affect classification results by identifying their respective performance strengths and weaknesses. Besides, HashBoost and SMOTEHashBoost show almost similar outcomes to GAN-based approaches for F1-score, AP, and G-Mean score. The SMOTEHashBoost takes a slightly longer computing time than the SMOTEBoost and HashBoost in HDHSS datasets. However, computational time is nearly identical for LDHSS, LDLSS, and HDLSS datasets. The SMOTEHashBoost combines three methods. Hence, the computational time is longer than that of SMOTEBoost and HashBoost, which is evident in the HDHSS datasets.

IV. CONCLUSION

This paper introduces a novel ensemble learning technique, SMOTEHashBoost, to address dataset class imbalance. By integrating AdaBoost boosting, hash-based undersampling, and SMOTE oversampling, the proposed method balances the dataset and improves classification performance, particularly for minority classes. Through extensive testing on multiple datasets, we demonstrated that SMOTEHashBoost consistently outperforms traditional methods. The hybrid technique ensures that minority and majority class samples are sufficiently represented during the training phase while concentrating on difficult-to-classify examples. One of the main takeaways from the results is that SMOTEHashBoost can maintain its robustness across a range of datasets with varying levels of class imbalance. Its ability to successfully combine oversampling, undersampling, and boosting approaches makes it useful for handling complex imbalanced datasets. Furthermore, the framework's versatility allows it to be adjusted to different sampling strategies and base classifiers.

Ultimately, the effectiveness of any strategy is dependent on the dataset's complexity, dimensionality, and sample size, and none will provide optimal results in all scenarios. Future studies could extend the application of SMOTEHashBoost to multi-class imbalanced problems and investigate additional optimization techniques to increase this algorithm's computing efficiency. Moreover, the performance of SMOTEHashBoost shows significant variation when applied to high-dimensional datasets. This variability is mainly due to hash collisions in the feature space and the curse of dimensionality, which leads to reduced feature separability. Thus, for large-scale and high-dimensional datasets, performance may be improved by including deep learning models in the ensemble.

REFERENCES

[1] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[2] 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.

[3] E. Chamseddine, N. Mansouri, M. Soui, and M. Abed, "Handling class imbalance in COVID-19 chest X-ray images classification: Using SMOTE and weighted loss," *Appl. Soft Comput.*, vol. 129, Nov. 2022, Art. no. 109588.

[4] B. Krawczyk, "Learning from imbalanced data: Open challenges and future directions," *Prog. Artif. Intell.*, vol. 5, no. 4, pp. 221–232, Nov. 2016.

[5] S. N. Kalid, K.-C. Khor, K.-H. Ng, and G.-K. Tong, "Detecting frauds and payment defaults on credit card data inherited with imbalanced class distribution and overlapping class problems: A systematic review," *IEEE Access*, vol. 12, pp. 23636–23652, 2024.

[6] 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.

[7] G. M. Weiss, "Mining with rarity: A unifying framework," *ACM SIGKDD Explorations Newslett.*, vol. 6, no. 1, pp. 7–19, Jun. 2004.

[8] 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. Hum.*, vol. 40, no. 1, pp. 185–197, Jan. 2010.

[9] M. Bach, A. Werner, and M. Palt, "The proposal of undersampling method for learning from imbalanced datasets," *Proc. Comput. Sci.*, vol. 159, pp. 125–134, Jan. 2019.

[10] 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.

[11] S.-J. Yen and Y.-S. Lee, "Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset," in *Intelligent Control and Automation*, vol. 344. Berlin, Germany: Springer, 2006, pp. 731–740.

[12] A. Anand, G. Pugalenthi, G. B. Fogel, and P. N. Suganthan, "An approach for classification of highly imbalanced data using weighting and undersampling," *Amino Acids*, vol. 39, no. 5, pp. 1385–1391, Nov. 2010.
[13] 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.
[14] 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.
[15] S. Barua, Md. M. Islam, X. Yao, and K. Murase, "MWMOTE-majority weighted minority oversampling technique for imbalanced data set learning," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 2, pp. 405–425, Feb. 2014.
[16] S. Bej, N. Davtyan, M. Wolfien, M. Nassar, and O. Wolkenhauer, "LoRAS: An oversampling approach for imbalanced datasets," *Mach. Learn.*, vol. 110, no. 2, pp. 279–301, Feb. 2021.
[17] 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.
[18] D. Dablain, B. Krawczyk, and N. V. Chawla, "DeepSMOTE: Fusing deep learning and SMOTE for imbalanced data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 9, pp. 6390–6404, Sep. 2022.
[19] Y. Zhang, B. Kang, B. Hooi, S. Yan, and J. Feng, "Deep long-tailed learning: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 9, pp. 10795–10816, Sep. 2023.
[20] D. A. Dablain, C. Bellinger, B. Krawczyk, and N. V. Chawla, "Efficient augmentation for imbalanced deep learning," in *Proc. IEEE 39th Int. Conf. Data Eng. (ICDE)*, Apr. 2023, pp. 1433–1446.
[21] R. Sauber-Cole, T. M. Khoshgoftaar, and J. M. Johnson, "GANs for class-imbalanced data: A meta-analysis of GitHub projects," in *Proc. IEEE 34th Int. Conf. Tools Artif. Intell. (ICTAI)*, Oct. 2022, pp. 1419–1424.
[22] 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.
[23] H. Beom Lee, H. Lee, D. Na, S. Kim, M. Park, E. Yang, and S. Ju Hwang, "Learning to balance: Bayesian meta-learning for imbalanced and out-of-distribution tasks," 2019, *arXiv:1905.12917*.
[24] M. Ochal, M. Patacchiola, J. Vazquez, A. Storkey, and S. Wang, "Few-shot learning with class imbalance," *IEEE Trans. Artif. Intell.*, vol. 4, no. 5, pp. 1348–1358, Oct. 2023.
[25] A. J. Ferreira and M. A. Figueiredo, *Boosting Algorithms: A Review of Methods, Theory, and Applications*. Cham, Switzerland: Springer, 2012, pp. 35–85.
[26] H. Binder, O. Gefeller, M. Schmid, and A. Mayr, "The evolution of boosting algorithms," *Methods Inf. Med.*, vol. 53, no. 6, pp. 419–427, Jan. 2014.
[27] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th Int. Conf. Int. Conf. Mach. Learn.*, Jul. 1996, pp. 148–156.
[28] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *J.-Jpn. Soc. Artif. Intell.*, vol. 14, nos. 771–780, p. 1612, 1999.
[29] R. Meir and G. Rätsch, *An Introduction to Boosting and Leveraging*. Berlin, Germany: Springer, 2003, pp. 118–183.
[30] 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.
[31] 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.
[32] 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.
[33] S. Ayesha, M. K. Hanif, and R. Talib, "Overview and comparative study of dimensionality reduction techniques for high dimensional data," *Inf. Fusion*, vol. 59, pp. 44–58, Jul. 2020.
[34] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2916–2929, Dec. 2013.
[35] C. O. S. Sorzano, J. Vargas, and A. P. Montano, "A survey of dimensionality reduction techniques," 2014, *arXiv:1403.2877*.
[36] M. J. Zaki and W. Meira, *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2014.
[37] S. Liao, J. Chen, Y. Wang, Q. Qiu, and B. Yuan, "Embedding compression with isotropic iterative quantization," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, vol. 34, no. 5, pp. 8336–8343.
[38] 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. Mult. Valued Log. Soft Comput.*, vol. 17, pp. 255–287, Jan. 2011.
[39] K. Nakai, 1991, "Yeast," UCI Machine Learning Repository, doi: 10.24432/C5KG68.
[40] M. Bohanec, 1988, "Car evaluation," UCI Machine Learning Repository, 10.24432/C5JP48.
[41] S. Aeberhard and M. Forina, 1992, "Wine," UCI Machine Learning Repository, doi: 10.24432/C5PC7J.
[42] M. Charytanowicz, J. Niewczas, P. Kulczycki, P. Kowalski, and S. Lukasik, 2010, "Seeds," UCI Machine Learning Repository, doi: 10.24432/C5H30K.
[43] B. German, 1987, "Glass identification," UCI Machine Learning Repository, doi: 10.24432/C5WW2P.
[44] B. Ramana and N. Venkateswarlu, 2022, "ILPD (Indian liver patient dataset)," UCI Machine Learning Repository, doi: 10.24432/C5D02C.
[45] 2020, "Early stage diabetes risk prediction," UCI Machine Learning Repository, doi: 10.24432/C5VG8H.
[46] T. Sejnowski and R. Gorman, 1988, "Connectionist bench (Sonar, mines vs. rocks)," UCI Machine Learning Repository, doi: 10.24432/C5T01Q.
[47] W. Wolberg, O. Mangasarian, N. Street, and W. Street, 1993, "Breast cancer Wisconsin (Diagnostic)," UCI Machine Learning Repository, doi: 10.24432/C5DW2B.
[48] I.-C. Yeh, 2009, "Default of credit card clients," UCI Machine Learning Repository, doi: 10.24432/C55S3H.
[49] V. Realinho, M. Vieira Martins, J. Machado, and L. Baptista, 2021, "Predict students' dropout and academic success," UCI Machine Learning Repository, doi: 10.24432/C5MC89.
[50] A. Sharma, P. K. Singh, and R. Chandra, "SMOTified-GAN for class imbalanced pattern classification problems," *IEEE Access*, vol. 10, pp. 30655–30665, 2022.
[51] 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, Jan. 2003.
[52] 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 Explorations Newslett.*, vol. 6, no. 1, pp. 20–29, Jun. 2004.
[53] 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.*, Jan. 2005, pp. 878–887.
[54] N. V. Chawla, A. Lazarevic, L. Hall, and K. W. Bowyer, "SMOTEBoost: Improving prediction of the minority class in boosting," in *Proc. Eur. Conf. Princ. Data Mining Knowl. Discovery*. Cham, Switzerland: Springer, Jan. 2003, pp. 107–119.



SEEMA YADAV is currently pursuing the Ph.D. degree in computer engineering with the Veermata Jijabai Technological Institute (VJTI), Mumbai, India. She is also 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 with the 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 with the Veermata Jijabai Technological Institute (VJTI), Mumbai, Maharashtra, India. He has made significant contributions as both a Developer and a Mentor at Google Summer of Code, participating, in 2023, and mentoring at INCF, in 2024. 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. He has also interned at Citi Bank India, where he built scalable web applications to improve operational efficiency.



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 ACM and the Computer Society of India.



GIRISH P. BHOLE (Member, IEEE) received the Ph.D. degree in engineering from the 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, India. His research interests include machine learning, data science, the IoT, and wireless networks. He has extensive experience in setting up and managing the data center.