A Hybrid Sampling Approach for Imbalanced Binary and Multi-Class Data Using Clustering Analysis

ABDUL SATTAR PALLI\textsuperscript{1,2}, JAFREEZAL JAFAAR\textsuperscript{1,3}, (Senior Member, IEEE), MANZOO AHMED HASHMANI\textsuperscript{1,4}, (Senior Member, IEEE), HEITOR MURILO GOMES\textsuperscript{5,6}, AND ABDUL REHMAN GILAL\textsuperscript{1,7}

\textsuperscript{1}Department of Computer and Information Sciences, Universiti Teknologi PETRONAS (UTP), Seri Iskandar, Perak 32610, Malaysia  
\textsuperscript{2}Anti-Narcotics Force, Ministry of Narcotics Control, Islamabad 46000, Pakistan  
\textsuperscript{3}Centre for Research in Data Science, UTP, Seri Iskandar, Perak 32610, Malaysia  
\textsuperscript{4}High-Performance Cloud Computing Centre (HPC3), UTP, Seri Iskandar, Perak 32610, Malaysia  
\textsuperscript{5}AI Institute, University of Waikato Wellington, Hamilton 3240, New Zealand  
\textsuperscript{6}School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6012, New Zealand  
\textsuperscript{7}Department of Computer Science, Sukkur IBA University, Sukkur, Sindh 65200, Pakistan

Corresponding author: Abdul Sattar Palli (abdulsattarpalli@gmail.com)

This work was supported by URIF under Grant 015LB-053.

\textbf{ABSTRACT} Unequal data distribution among different classes usually cause a class imbalance problem. Due to the class imbalance, the classification models become biased toward the majority class and misclassify the minority class. Class imbalance issue becomes more complex when it occurs in multi-class data. The most common method to handle the class imbalance is data resampling that involves either over-sampling minority class instances or under-sampling majority class instances. In the case of under-sampling, there is a chance of losing some crucial information, whereas over-sampling can cause an overfitting problem. Therefore, we propose a novel Cluster-based Hybrid Sampling for Imbalance Data (CBHSID) approach to address these issues. The CBHSID calculates the mean of the data observations based on the number of classes. It uses the calculated mean as a threshold value to segregate majority and minority classes. CBHSID applies affinity propagation cluster analysis to each class to create sub-clusters and calculates the distance of each data item of sub-cluster using centroid mean. CBHSID removes data observations that are away from the center of sub-cluster during under-sampling. On the other hand, during the over-sampling, it generates synthetic samples using data observations near to the center of sub-cluster. We compared CBHSID with a few state-of-the-art data balancing methods on 12 binary and 4 multi-class benchmark datasets. Based on Geometric-Mean (G-Mean), Recall, and F1-score, our method outperformed the other compared methods on 14 datasets out of 16. Results also revealed that CBHSID is suitable for addressing class imbalance issues in both binary and multi-class classifications. In the current state, we have only validated CBHSID on stationary data streams. Consequently, CBHSID can further be tested on non-stationary data streams in online learning environments.

\textbf{INDEX TERMS} Class imbalance, classification, clustering analysis, binary class, multi-class.

\section{I. INTRODUCTION}
Learning from imbalanced data has remained the area of interest in the research community over the last many years. The health systems [1], [2], [3], [4], fraud detection systems [5], anomaly detection [6], and fault diagnosis [7], [8] are common examples of applications with unequal data distribution, which leads to the class imbalance problem [9]. In any data set, the class with a higher number of data samples is the majority class and the one with fewer data samples is
known as a minority class [10]. The classifier trained on such data becomes biased towards the majority class [11]. For example, in financial systems where abnormal data samples are rare, most of the data samples are normal. The classification model trained on such data will produce 99.9% accuracy and considers only 0.1% of illegal transactions. As a result, it misclassifies most of the illegal transactions. Such classifiers could not be effective in real-world financial systems. Hence, it is hard for traditional classifiers to classify the imbalanced data accurately [12]. Over the years, many approaches have been proposed to address the class imbalance issue. The core objective of these approaches is to improve the performance of the classifier for the minority class. These approaches are generally divided into four categories: Data-level approaches; Algorithm-level approaches; Ensemble approaches, and Hybrid approaches [13], [14]. The Data-level approach addresses the class imbalance problem by data resampling. Data resampling can be achieved by either increasing the minority class samples known as over-sampling or decreasing the majority class samples known as under-sampling. Oversampling causes the issue of overfitting, and eliminating data during under-sampling may lead to the loss of useful information [15], [16].

The algorithm-level approaches are focused on producing an intelligent mechanism for data sampling. They apply weights to minority class samples to modify the learning algorithm to give more importance to the minority class samples. The main limitation of these approaches is that it is hard to calculate the weights of the minority class samples for most real-life applications [17]. On the other hand, Ensemble-level imbalance handling has become another major category of methods to handle class imbalance by combining multiple classifiers like SMOTEBoost [18] and AdaBoost.NC [19].

The hybrid approaches are a combination of data-level and algorithm-level approaches. These approaches try to adopt the strengths of both approaches such as [20], [21], and [22]. The existing hybrid approaches use the traditional concepts of data-level and algorithm-level approaches. They mostly rely on oversampling the minority class samples and under-sample the noised samples of majority class [23]. Instead of oversampling minority samples and removing noise only, some approaches use cluster analysis to identify the most valuable data samples from the majority class and then apply data under-sampling such as [17] and [24]. The cluster-based approaches help in keeping the useful information of the majority class and only under-sample the data which is less important. In general, the existing data balancing approaches are unable to get the more relevant information for resampling except cluster-based approaches. Whereas the existing cluster-based approaches either under-sample the majority class or over-sample the minority class. Hence these approaches may again lead to the overfitting problem.

Therefore, to overcome the limitations of the existing approaches, we propose the Cluster-based Hybrid Sampling approach for Imbalance Data (CBHSID). In CBHSID, we first calculate the mean (threshold) of the data observations concerning the number of classes. We use this threshold to separate the minority and majority classes of the underlying dataset. The same threshold is also considered the limit of samples in each class after under-sampling and oversampling. Any class with a number of samples less than a threshold is a minority class. In contrast, the class with the number of instances greater than the threshold is considered a majority class. After that, we divide each class into groups or sub-clusters with similar data points using affinity propagation [25]. Next, we calculate the number of data observations (k) to add or remove from each sub-cluster based on the size of the sub-cluster. We further calculated the distance of each item from its sub-clusters centroid. We kept the data observations closer to the centroid during under-sampling and discarded the rest for every sub-cluster of the majority class. In over-sampling, data observations near the centroid were used to generate k synthetic samples for each sub-cluster to raise the examples in the minority class. Hence every class (minority and majority) will have an almost identical number of instances, nearly equal to the threshold.

The rest of the paper is organized as follows. Section 2 discusses the existing approaches proposed to address the class imbalance issue. Section 3 describes the proposed method. Section 4 presents the experimental results and analysis; finally, we conclude our paper in section 5.

II. RELATED WORK

The class imbalance is a widespread problem in most applications such as social media mining [26], risk management [27], and anomaly detection [6]. These applications face a situation where data in one class (minority) is skewed or underrepresented than the other (majority class), termed a class imbalance [28]. For such applications, the classification models misclassify the minority class samples. Hence, different approaches have been proposed to address the issue to improve the classifier’s performance for the minority class. These approaches are broadly divided into four categories: Data-level strategies; Algorithm-level strategies; Ensemble strategies, and Hybrid strategies [13], [29]. There is another data balancing approach which uses any of these four approaches but it first converts data into clusters using cluster analysis and then applies data resampling. We briefly discuss each approach type in following subsections.

A. DATA-LEVEL APPROACHES

The most common and very simple approach to address the class imbalance is a data-level approach that resamples the actual data. Resampling is achieved by over-sampling, under-sampling, or combining both [30], [31]. Random Over-sampling (ROS) [32] and Random Under-sampling (RUS) [33] are the two standard versions of data resampling. Such methods randomly select samples from minority or majority classes to increase or decrease a particular class. Most of the time, ROS leads to an overfitting problem [16], [29]. In RUS, eliminating elements from the majority class sometimes causes the loss of valuable information [10].
B. ALGORITHM-LEVEL APPROACHES
Innovative algorithmic approaches like SMOTE [34] come into play to overcome the limitations of data-level approaches. It draws a line between k nearest neighbors of minority samples and generates synthetic data on these lines. Algorithm-level approaches are more focused on modifying the learning algorithm in a way that gives more importance to the minority samples, such as MWMOTE [35] and Adaptive Synthetic Sampling (ADASYN) [36]. These approaches apply weights or costs to misclassified data. In other words, higher weights are applied to minority class examples, whereas majority class examples are assigned lower weights. The learning algorithm is adjusted accordingly (the learning algorithm aims to minimize the global cost associated with misclassification) so that the algorithm becomes biased towards the minority class [37]. The main limitation of this approach is that it is hard to set actual values of the cost matrix for most real-life applications [17].

C. HYBRID APPROACHES
A hybrid method SMOTE-Tomek [23], applies both oversampling and under-sampling approaches. It uses SMOTE to generate synthetic data for minority classes and Tomek [38] to remove the ambiguous and noisy data from the decision boundary. We can say that SMOTE-Tomek mainly relies on over-sampling. It does not really under-sample the majority class but only removes the noised data. Another hybrid data sampling approach introduced in the same paper combines SMOTE with Edited Nearest Neighbor (ENN) [39], known as SMOTEENN. The SMOTE was used for over-sampling, whereas ENN was applied for under-sampling. The ENN finds the k nearest neighbors of the observation and returns the class with more samples among the nearest neighbor, i.e., the minority class of that small group. It discards the observation and its neighbors if the observation class and its neighbor’s majority class are not the same. The SMOTEENN may struggle in reducing the imbalance ratio because it removes the data observation and its neighbors when the majority of the neighbors and the observation itself belong to two different classes. There could be a situation where observation belongs to the negative (majority) class, and nearest neighbors belong to the positive (minority) class. In such cases, removing observation and its nearest neighbors may further increase the class imbalance because it removes minority class samples in the majority of only among most immediate neighbors.

D. ENSEMBLE APPROACHES
The ensemble approach is another famous method to deal with the class imbalance problem. In the ensemble learning approach, multiple classifiers are trained on training data. Their output is aggregated to get the final result. Bagging [40] and boosting [41] are well-known ensemble methods to deal with class imbalance. The bagging or Bootstrap Aggregation minimizes the prediction variance. It creates a separate subset of the training data for each model so that every model is trained on a different copy of data (but is a subset of the primary dataset). It uses a complete copy of the dataset to test the models and get the final decision after the aggregation of outputs of all models. The random forest [42] is the most common example of the bagging approach. Using multiple models instead of a single model reduced the overfitting problem. As the final output is the average of all models’ results, one cannot achieve a precise value for the model. In boosting, on the other hand, multiple classifiers are sequentially created. The data samples that the initial model misclassifies are assigned higher weights. A new classifier is built, which will be trained on the updated data. The process is repeated till a certain level of accuracy is achieved. The AdaBoost [43] and Gradient Boosting [44] are common examples of boosting approaches.

Combining different approaches such as data-level and algorithm-level gives an improved version of the technique known as the hybrid approach. Some intelligent approaches, such as Kubat and Matwin [45], offered an intelligence-based under-sampling one-sided selection (OSS) approach, which selectively removes the majority instances that are either redundant or borderline majority examples. Overall, all the above-discussed methods are unable to get the more relevant information for resampling.

E. CLUSTER-BASED APPROACHES
The cluster-based imbalanced handling approaches aim to keep the more relevant information of a majority class and discard the rest or even create multiple clusters of majority class and combine each sub-cluster with minority class to create multiple balanced datasets [46]. Yen and Lee [17] presented a cluster-based under-sampling algorithm (CLUS) where training data with homogeneous characteristics is organized into groups and then the number of majority class samples in each cluster downsized. Two cluster-based majority class under-sampling approaches were proposed in [24]. Where k clusters are created from the majority class and the ratio of each sub-cluster class is calculated to the total number of samples of the majority class. In the first approach, certain numbers of majority class samples are selected from each sub-cluster. Whereas in the second approach they selected the majority class samples which are near to the centroid of the sub-cluster. At last, they merge the selected majority class samples with minority class samples to create a training set. The fast-CBUS [47] is another cluster-based under-sampling approach that creates k clusters of the minority class. It then defines the boundary of each sub-cluster based on the distance of the foremost minority item of that sub-cluster. The observations of a majority class that lies outside of this boundary are discarded during under-sampling. This step is repeated for each sub-cluster. In case the majority class observations are less in numbers than the minority class in any sub-cluster they randomly select majority observations to balance the sub-cluster. This approach may face trouble with datasets where minority class samples are sparse and majority class samples are abundant within the boundary of minority
Another cluster-based approach for under-sampling was proposed by Tsai et al. in [48], where they apply cluster analysis on majority class and convert it into small groups or clusters. They use the instance selection method to select a certain number of samples from each group (cluster). As a result, they develop a reduced majority class and then combine it with minority class to form a training set. Their proposed approach was tested only on binary class datasets. In [49], Zhu et al. applied a neighborhood-based clustering algorithm to divide minority class samples into three different groups based on their position: inland minority, borderline minority, and trapped minority. They further apply various over-sampling strategies to generate synthetic samples for each group of minority samples.

An adaptive weighted over-sampling approach is proposed in a more recent work [50] using density peaks clustering. In this study, clustering is applied to minority class samples. The sub-clusters with sparse data (less dense sub-clusters), which are closer to the borderline of the majority class, are assigned higher weights to achieve a high probability of selection for generating a new synthetic sample. They also apply a heuristic filter to minority samples. The existing cluster-based data sampling approaches mostly under-sample the majority class. Applying such methods in applications where the imbalance ratio is too high and contains very rare minority class samples may discard most of the data from the majority class to achieve a balanced training set. As a result, useful information can be lost.

Recently, [51] proposes a clustering strategy based on the Red Deer Algorithm (RDA) to address the class imbalance in kidney rejection data. In order to improve the accuracy of their final prediction models, the authors devised a three-stage clustering-based undersampling strategy to discard samples from the majority class. The study was limited to only kidney rejection dataset, also removing samples from majority class may cause to lose the important information.

In order to pre-process unbalanced data, the [52] study presents Reduced Noise-SMOTE (RN-SMOTE). To begin, RN-SMOTE uses SMOTE to over-sample the training data, leading to the introduction of noisy oversampled synthetic examples in the minority class. Then, they used DBSCAN to concentrate and filter the noise. The next step is to merge the original data with the pristine synthetic instances. After the dataset has been rebalanced using SMOTE, RN-SMOTE feeds it into the underlying classifier. The RN-SMOTE was limited to only binary class imbalance problem.

Inspired by the cluster-based approaches, we propose an approach called Cluster-based Hybrid Sampling for Imbalanced Data to address the class imbalance issue. The existing cluster-based data sampling approaches only apply either over-sampling or under-sampling, but our proposed approach is a combination of new over-sampling and under-sampling methods, unlike SMOTEENN and SMOTETomek, which combine two existing approaches. The following section discusses the proposed approach in detail.

III. PROPOSED APPROACH

The Cluster-based Hybrid Sampling for Imbalance Data (CBHSID) first calculates the threshold using (1), which is further used to separate the minority and majority classes of the underlying dataset. The $Z_D$ is the size of the dataset D, and $C_D$ is the number of classes in D. This threshold ($\mu$) is then considered the limit of samples in each class after under-sampling and over-sampling. Any class with less number of samples than a threshold is considered a minority class (MI), whereas the class with a higher number of instances than the threshold is considered a majority class (MJ).

Each class is divided into groups of data examples with similar properties called as sub-clusters using affinity propagation (AP) [25]. For under-sampling of the majority class, data points from each sub-cluster of the majority class that are away from the sub-cluster centroid are removed to get observations close to the centroid. In the case of over-sampling the minority class, we use the data observations from sub-clusters of the minority class which are closer to the centroid to generate synthetic data. The core advantage of our approach is that it will not increase the training complexity of the classifier because it balances the training set in a way that the size of the training set remains the same. As CBHSID is a hybrid approach, the number of observations it removes from the majority class during under-sampling is equal to the number of synthetic observations it adds to the minority class while over-sampling. The flow diagram of the proposed approach is shown in Figure 1.

$$\mu = \frac{Z_D}{C_D}$$ (1)

A. CLUSTER ANALYSIS

Cluster analysis helps in identifying the most valuable data in the class. Existing well know data balancing approaches like ROS and RUS randomly select data for adding or removal purposes. The AP was chosen as the clustering approach to divide each class into multiple sub-clusters (we say k sub-clusters). The AP does not require the value of k (number of clusters to generate). AP is applied to each class which provides a certain number of sub-clusters based on data similarity; it also provides the centroid for each sub-cluster. Next, we under-sample the majority class sub-clusters and oversample the minority class sub-clusters to balance the data. The CBHSID removes observations that are away from the centroid in under-sampling and generates synthetic data using observations closer to the centroid to over-sample the data. To identify how far (or how close) an observation is from its centroid, it calculates the distance of each observation of the sub-cluster from its centroid. The distance is nothing but the difference in means. To get the distance of observation, it first calculate the mean of every observation available in the sub-cluster and stores in a new feature called Mean. Let's say, if there are m observations in a sub-cluster with n features,
it calculates the mean of every observation using (2) given below:

$$\text{Mean}_{X_r} = \frac{1}{n} \sum_{f=1}^{n} X_{r,f} \quad \text{Where} \quad r \in \{1, 2, \ldots, m\} \quad (2)$$

The $X_r$ represents the $r^{th}$ observation and $X_{r,f}$ represents the $f^{th}$ feature of the $r^{th}$ observation, $n$ represents the total number of features in an observation $X_r$, and $\text{Mean}_{X_r}$ is the calculated mean of all features of the observation $X_r$. Using this mean, it calculates the absolute difference as the distance of each observation from the centroid of the sub-cluster using (3) given below:

$$\text{Distance}_{X_r} = \text{abs}(\text{Mean}_{\text{centroid}} - \text{Mean}_{X_r}) \quad (3)$$

Equation (3) subtracts the mean of every observation of a sub-cluster from the mean of the $X_{\text{centroid}}$ and returns absolute difference. The $X_{\text{centroid}}$ represents the observation with a centroid (center of the sub-cluster). This difference is considered as a distance of observation $X_r$ from the centroid. The observation with a smaller distance value is considered the nearest observations to the centroid. In contrast, the observation with higher distance value is considered the farthest observation from the centroid.

Generally, class imbalance handling approaches either increase the size of training data in case of over-sampling or decrease due to under-sampling and losing some helpful information. The advantage of our proposed method is that it keeps the valuable information and the overall size of the training data remains almost the same as the original one as we apply both under sample and over-sample approaches. The details of novel under-sampling and over-sampling techniques are discussed in following sub-sections, and then summarised in Algorithm 1, Algorithm 2, and Algorithm 3.

**B. CLUSTER-BASED UNDER SAMPLING**

Suppose the class is identified as the majority class (MJ). In that case, we have to remove a certain number of observations from the majority class, i.e., $N_{\text{REMOVE MJ}}$, to reduce the size of the majority class ($N_{\text{MJ}}$) to the threshold level; this is achieved using (4) given below:

$$N_{\text{REMOVE MJ}} = \text{abs}(\mu - N_{\text{MJ}}) \quad (4)$$

After applying clustering analysis as discussed in section III, the majority class is converted into multiple sub-clusters; The $N_{\text{REMOVE MJ}}$ is distributed among different sub-clusters based on their size; In other words, the $N_{\text{REMOVE MJ}}$ is equal to the sum of the observations removed from each sub-cluster of MJ. We propose a novel formula to calculate the number of observations to remove from every sub-cluster of the majority class MJ using the (5) given below:

$$N_{\text{REMOVE SCi MJ}} = \text{abs}\left(\text{round}\left(\frac{N_{\text{SCi MJ}} \cdot \mu}{N_{\text{MJ}}} \right) - N_{\text{SCi MJ}}\right) \quad (5)$$

The SCi is the $i^{th}$ sub-cluster (SC), so SCi$\text{MJ}$ represents the $i^{th}$ sub-cluster of majority class MJ. Whereas N$_{\text{SCi MJ}}$ represents the number of observations in SCi of MJ. Hence, the $N_{\text{REMOVE SCi MJ}}$ represents the number of observations to remove from $i^{th}$ sub-cluster of the majority class. The $N_{\text{REMOVE SCi MJ}}$ observations far away from the centroid (having a higher distance from the centroid) are removed from each sub-cluster of the majority class. This is achieved using the calculated distance of each observation. We arrange these distances in descending order (observations with higher distance on top), and then the top $N_{\text{REMOVE SCi MJ}}$ observations are removed. In other words, the observations that are far away from the centroid in terms of mean difference are removed. As a result, we get data observations near the centroid for each sub-cluster. These remaining observations of each sub-cluster are merged to form an under-sampled majority class with the size of $\mu$ approximately.

In Figure 2, we illustrate how the under-sampling part of our proposed approach works. Figure 2(a) illustrates the actual majority class. After applying cluster analysis we get sub-clusters of the class which is illustrated in Figure 2(b). In Figure 2(c), the lines from the center of the sub-cluster to the observation represent the distance of that observation from the centroid. The observations with samll distances are represented with blue-colored lines and observations with higher distance value are represented with red-colored lines. As CBHSID removes observations which are away from the centroid and keeps observations near to the centroid based on the distance value, hence we illustrated observations which
Algorithm 1: Main Function

Input: imbalanced training set $D$, number of classes $C_D$, size of training set, $Z_D$
Output: balanced training set $D'$

1. $\mu = \frac{Z_D}{k}$ // Calculate threshold
2. for $i = 1$ to $C_D$ Do
3. if (size($D[i]$) $\geq \mu$)
4. \hspace{1cm} $N_{MJ} =$ size($D[i]$) // $D[i]$ contains observation related to class $i$
5. \hspace{1cm} $N_{\text{REMOVE}}_{MJ} = \text{abs}(\mu - N_{MJ})$ // size of class $i$ which is majority class $MJ$
6. \hspace{1cm} $D' =$ call undersampling($D[i], \mu$) // number of observations to remove from $MJ$
7. else
8. \hspace{1cm} $N_{MI} =$ size($D[i]$) // size of class $i$ which is minority class $MI$
9. \hspace{1cm} $N_{\text{ADD}}_{MI} = \text{abs}(\mu - N_{MI})$ // number of observations to remove from $MI$
10. \hspace{1cm} $D' =$ call oversampling($D[i], N_{\text{ADD}}_{MI}$) // update $D'$ with the oversampled majority class
11. End if
12. end for

Algorithm 2: Undersampling Function

Input: majority class $MJ$, threshold $\mu$
Output: under sampled majority class $MJ'$

1. $N_{MJ} =$ size($MJ$)
2. $MI_{clusters} = \text{AP}(MJ)$ // $MJ = \text{clusters}$
3. $k = \text{count}(MI_{clusters})$ // number of clusters of $MJ$
4. for $i = 1$ to $k$ Do
5. \hspace{1cm} $N_{SCi}MI =$ size($MI_{clusters}[i]$) // number of observations in $i$th sub cluster
6. \hspace{1cm} $SCi = MI_{clusters}[i]$ // $SCi$ contains observations of $i$th sub cluster
7. \hspace{1cm} $X_{centroid} = \text{AP.centroid}(MI_{clusters}[i])$ // $X_{centroid}$ is center of the sub cluster
8. \hspace{1cm} $N_{\text{REMOVE}}_{SCiMI} =$ abs ($\text{round} \left( \frac{N_{SCiMI} - N_{MJ}}{N_{MI}} \right)$) // number of observations to remove from $i$th sub cluster, Eq. 5
9. for $r = 1$ to $N_{SCiMI}$ Do
10. \hspace{1cm} $Mean_{X_i} = \frac{1}{n} \sum_{j=1}^{n} X_{rj}$ // calculate the distance for every observation of $SCi$
11. \hspace{1cm} $Distance_{X_i} = \text{abs} (\text{Mean}_{X_{centroid}} - \text{Mean}_{X_i})$ // Eq. 2
12. end for
13. sort $SCi$ by distance in descending order
14. remove top $N_{\text{REMOVE}}_{SCiMI}$ from $SCi$ and update & update $MJ$
15. end for
16. Return $MI'$ // returns the under sampled majority class

Algorithm 3: Oversampling Function

Input: minority class $MI$, threshold $\mu$
Output: over sampled minority class $MI'$

1. $N_{MI} =$ size($MI$)
2. $MI_{clusters} = \text{AP}(MI)$ // $MI = \text{clusters}$
3. $k = \text{count}(MI_{clusters})$ // number of clusters of $MJ$
4. for $i = 1$ to $k$ Do
5. \hspace{1cm} $N_{SCiMI} =$ size($MI_{clusters}[i]$) // number of observations in $i$th sub cluster
6. \hspace{1cm} $SCi = MI_{clusters}[i]$ // $SCi$ contains observations of $i$th sub cluster
7. \hspace{1cm} $X_{centroid} = \text{AP.centroid}(MI_{clusters}[i])$ // $X_{centroid}$ is center of the sub cluster
8. \hspace{1cm} $N_{ADD}_{SCiMI} =$ abs ($\text{round} \left( \frac{N_{SCiMI} - N_{MI}}{N_{MI}} \right)$) // number of observations to remove from $i$th sub cluster, Eq. 6
9. for $r = 1$ to $N_{SCiMI}$ Do
10. \hspace{1cm} $Mean_{X_i} = \frac{1}{n} \sum_{j=1}^{n} X_{rj}$ // calculate the distance for every observation of $SCi$
11. \hspace{1cm} $Distance_{X_i} = \text{abs} (\text{Mean}_{X_{centroid}} - \text{Mean}_{X_i})$ // Eq. 2
12. end for
13. sort $SCi$ by distance in ascending order
14. Generate $N_{ADD}_{SCiMI}$ synthetic observations using Eq. 8 and update the $MI$
15. end for
16. Return $MI'$ // returns the over sampled minority class
are near to the centroid within dotted-circle. Figure 2(d) shows the majority class after under-sampling.

C. CLUSTER-BASED OVER SAMPLING

For over-sampling, we need to add a certain number of samples in the minority class (MI), denoted as \( N_{\text{ADD}}^{\text{MI}} \), to increase the size of the minority class to the threshold level. We use the symbol \( N_{\text{MI}} \) to represent the size of the minority class. The \( N_{\text{ADD}}^{\text{MI}} \) is calculated using (6) given below:

\[
N_{\text{ADD}}^{\text{MI}} = \text{abs}(\mu - N_{\text{MI}})
\]  

(6)

After applying clustering analysis to the minority class, as discussed in section III, the minority class illustrated in Figure 3(a) is converted into different sub-clusters, as illustrated in Figure 3(b). We further calculate the number of observations to add to each sub-cluster defined as \( N_{\text{ADD}}^{\text{SCiMI}} \) based on the size of the sub-cluster or number of observations in ith sub-cluster \( N_{\text{SCI}} \) of minority class MI using (7) given below:

\[
N_{\text{ADD}}^{\text{SCiMI}} = \text{abs}\left(\text{round}\left(\frac{N_{\text{SCI}}^{\text{MI}} \times \mu}{N_{\text{MI}}} \right) - N_{\text{SCI}}^{\text{MI}}\right)
\]

(7)

For over-sampling, we repeat the same method to calculate the distance of each observation from the centroid. This time we arrange these distances in ascending order (data observations with lesser distance on top), and then the top \( N_{\text{ADD}}^{\text{SCiMI}} \) observations are selected for generating synthetic data. These selected observations are the data observations that are near the centroid. For generating a synthetic data observation with \( n \) features, we use all \( n \) features of the nearest observation one by one. First feature of nearest observation is used to generate first feature of the synthetic observation, second feature of nearest observation is used to generate second feature of the synthetic observation, and so on.

To generate a synthetic data, we first take the absolute difference of the first two consecutive feature values, i.e., \( \text{abs}(F_i - F_{i+1}) \) of the nearest observation to generate the synthetic value. \( F_i \) represents the ith feature; we start with \( i = 0 \) to get the value of the first feature. We multiply it with a randomly selected feature from observation with a centroid value, i.e., \( X_{\text{centroid}} \). Again we multiply the absolute difference with the mean value of the centroid observation. Finally, we sum these values and add the actual value of \( F_i \) to get the synthetic value for the first feature of the synthetic observation, which is represented as \( F_i' \).

We iterate this step by increasing the value of \( i \) by one till \( i \) reaches to \( n + 1 \) (the mean column) to generate \( n \) synthetic feature values to complete one synthetic observation (a complete row with \( n \) features). The method is repeated.
till we generate the required number of observations to add in a specific sub-cluster i.e. $N_{ADD, SCiMI}$, as illustrated in Figure 3(c). In case the $N_{ADD, SCiMI}$ (number of samples to add in ith sub-cluster) is equal to or greater than the $N_{SciMI}$ (total number of samples in ith sub-cluster), then all the actual observations are used to generate the synthetic observations (synthetic data) or even one actual observation can be used to generate multiple observations. Selecting random values from the centroid observation helps us generate unique synthetic observations even if we generate multiple synthetic observations using a single actual observation. Eq. 8 shows the complete statement for calculating the synthetic value for a feature $Fi$. The $F_i$ and $F_{i+1}$ are two consecutive features of the nearest observation and $X_{centroid}$ is the central observation. The random ($X_{centroid}$) gives feature value from the central observation randomly, whereas Mean ($X_{center}$) gives the mean value of central observation.

\[
Fi' = \text{abs}(F_i - F_{i+1}) \times \text{random}(X_{centroid}) + \text{abs}(F_i - F_{i+1}) \times \text{Mean}(X_{centroid}) + F_i \quad (8)
\]

IV. EXPERIMENTAL RESULTS AND DISCUSSION

A. EXPERIMENT SETTING

1) BASE CLASSIFIER

The most well-known and straightforward classification algorithm, Support Vector Machine (SVM), is considered the base classifier for our experiments, the same has been widely used for the classification tasks as explained by [53]. The SVM is initially formulated for binary class problems [54]. However, by setting up different data arrangements, nowadays SVM is being used for multiclass classification too such as [55]. In this study for binary class data simple version of SVM, whereas for multi-class data the multi-class version of SVM is used with one-vs-rest classification setting which is considered less complex than one-vs-one.

2) PERFORMANCE METRICS

The classification performance is measured on the geometric mean (G-Mean), Recall, and F1 score metrics, mostly used in the literature when data is imbalanced [56]. The G-Mean is the geometric mean of the True Positive Rate (TPR) and True Negative Rate (TNR). A good classifier should have high TPR and TNR values which result in achieving high G-Mean. The recall is the proportion of the correctly classified positive class observations to the total observation in the positive class. The positive class is in the minority in most applications such as health systems [1]. Therefore, the algorithms which produce high Recall values are considered more helpful in such applications. The F1 score is another valuable metric to measure model performance when data is imbalanced. It is the harmonic mean of TPR and TNR; instead of considering only positive class, it considers all classes (positive as well as negative).

3) DATASETS

Experiments of the current study are performed on twelve binary and four multi-class datasets, which are selected from the University of California Irvine (UCI) and KEEL data repositories. The datasets are chosen based on the class imbalance ratio. The imbalance ratio is calculated as the number of data samples in the majority class over the number of data samples in the minority class for both binary and multi-class datasets. The imbalance ratio of the selected datasets ranges from 1.38 to 41.4. If the value of the imbalance ratio is closer to 1 means data is less imbalance and high imbalance otherwise. The selected datasets are considered as benchmark and widely used in the literature as summarized by [29]. The summary of the datasets used in this study is mentioned in Table 1.

The proposed approach is compared with well-known and commonly used state-of-the-art data balancing techniques that include data-level: ROS, RUS, SMOTE; hybrid methods: SMOTETomek and SMOTEENN and a cluster-based approach ClusterCentroid (ClusterC), the summary of these approaches is given in Table 2. The experiments are performed using imblearn and sklearn Python libraries. All approaches are used with default parameters. The train test split method is used to generate different training sets and testing sets each time with 80:20 ratio [57]. For every dataset, 80% of the observations are used for training and 20% are used for testing of the classifier. The experiments are run five times and the average of results is considered as the final result.

| Sr. | Dataset       | No# of Features | No# of Classes | Imbalance ratio |
|-----|---------------|-----------------|----------------|-----------------|
| 1   | Liver         | 6               | 2              | 1.38            |
| 2   | Pima          | 8               | 2              | 1.87            |
| 3   | Cancer        | 10              | 2              | 1.89            |
| 4   | Heart failure | 12              | 2              | 2.13            |
| 5   | Yeast1        | 8               | 2              | 2.46            |
| 6   | Haberman      | 3               | 2              | 2.78            |
| 7   | Vehicle1      | 18              | 2              | 2.9             |
| 8   | Glass6        | 9               | 2              | 6.38            |
| 9   | Yeast3        | 8               | 2              | 8.1             |
| 10  | Yeast4        | 8               | 2              | 28.1            |
| 11  | Yeast5        | 8               | 2              | 32.73           |
| 12  | Yeast6        | 8               | 2              | 41.4            |
| 13  | Penbased      | 16              | 10             | 1.95            |
| 14  | Contraceptive | 9               | 3              | 5.6             |
| 15  | Dermatology   | 34              | 6              | 36.94           |
| 16  | Thyroid       | 21              | 3              | 39.17           |

| Sr. | TECHNIQUES NAME | METHOD              |
|-----|-----------------|---------------------|
| 1   | RANDOM OVER SAMPLING | OVERSAMPLING       |
| 2   | RANDOM UNDER SAMPLING | UNDERSAMPLING       |
| 3   | SMOTE            | OVERSAMPLING        |
| 4   | SMOTETomek       | HYBRID SAMPLING     |
| 5   | SMOTEENN         | HYBRID SAMPLING     |
| 6   | CLUSTERCENTROID  | CLUSTER BASED OVERSAMPLING |

TABLE 1. Summary of the datasets.

TABLE 2. Summary of state-of-the-art techniques.
B. EXPERIMENT RESULTS AND DISCUSSION
The performance of the proposed approach is measured on three different performance metrics. We discuss each of those below.

Table 3 shows the G-Mean results obtained using six state-of-the-art data balancing approaches and our proposed CBHSID approach on 16 datasets (12 binary class and 4 multi-class). The results show that our proposed algorithm outperformed the other approaches on most of the datasets. It completely outperformed all the compared methods on 10 out of 12 binary class datasets. Out of 4 multi-class datasets, our approach outperformed on 3 datasets (pen-based, contraceptive, and Thyroid). It can be observed that the datasets on which CBHSID performed better than approaches compared, the range of their imbalance ratio is vast, i.e., CBHSID can perform well on datasets with low imbalance as well as high imbalance.

Table 4 shows the performance in terms of Recall. The results show that our proposed algorithm outperformed the other approaches used in the experiments on 12 datasets out of 16. However, on Yeast5, Yeast6, Dermatology, and Thyroid datasets which are highly imbalanced, SMOTE performed better than CBHSID. Table 5 shows the performance of compared approaches in terms of F1 score. The results show that our proposed algorithm outperformed the other approaches used in the experiments on 12 datasets out of 16.

The pairwise comparison was performed between CBHSID and other selected state-of-the-art methods to validate the performance of CBHSID further. Table 6 shows the comparison results of CBHSID with other methods for 16 datasets on three evaluation metrics. A similar kind of comparison is performed in [58]. When CBHSID performed better than or equal to the compared method, we represent it with ‘1’ and ‘0’ otherwise. Columns from 1 to 16 represent

| Dataset   | ROS   | RUS   | SMOTE | SMOTEENN | SMOTEtomek | Cluster-C | CBHSID |
|-----------|-------|-------|-------|-----------|------------|-----------|--------|
| River     | 0.6520±0.046 | 0.6234±0.048 | 0.6378±0.042 | 0.6116±0.045 | 0.6524±0.046 | 0.6176±0.098 | 0.6668±0.037 |
| Pima      | 0.5920±0.002 | 0.7740±0.003 | 0.7310±0.002 | 0.7016±0.002 | 0.7153±0.010 | 0.7378±0.023 | 0.7558±0.032 |
| Cancer    | 0.9584±0.014 | 0.9570±0.017 | 0.9556±0.015 | 0.9600±0.016 | 0.9570±0.015 | 0.9544±0.021 | 0.9612±0.015 |
| Heart failure | 0.7636±0.066 | 0.7600±0.008 | 0.7866±0.052 | 0.7830±0.043 | 0.7656±0.081 | 0.7668±0.02 | 0.8032±0.034 |
| Yeast1    | 0.8214±0.029 | 0.7072±0.023 | 0.7110±0.020 | 0.6664±0.018 | 0.7106±0.030 | 0.6332±0.014 | 0.7408±0.021 |
| Haberman  | 0.6966±0.074 | 0.6712±0.068 | 0.667±0.062 | 0.6644±0.060 | 0.6870±0.075 | 0.5742±0.084 | 0.7872±0.042 |
| Vehicle1  | 0.7624±0.029 | 0.7190±0.036 | 0.7612±0.020 | 0.7070±0.023 | 0.7578±0.026 | 0.7576±0.035 | 0.7764±0.023 |
| Glass6    | 0.9582±0.048 | 0.9582±0.026 | 0.9582±0.038 | 0.9440±0.031 | 0.9484±0.027 | 0.9674±0.013 | 0.9720±0.020 |
| Yeast3    | 0.9104±0.012 | 0.9306±0.016 | 0.9278±0.015 | 0.9098±0.009 | 0.9218±0.011 | 0.9144±0.015 | 0.9346±0.019 |
| Yeast4    | 0.8702±0.015 | 0.8504±0.029 | 0.8870±0.017 | 0.8546±0.024 | 0.8826±0.019 | 0.849±0.032 | 0.8876±0.006 |
| Yeast5    | 0.9644±0.004 | 0.9218±0.009 | 0.9664±0.005 | 0.9342±0.017 | 0.9472±0.017 | 0.9306±0.004 | 0.9392±0.002 |
| Yeast6    | 0.9280±0.015 | 0.8924±0.040 | 0.9494±0.013 | 0.9218±0.011 | 0.9290±0.018 | 0.9096±0.013 | 0.9400±0.012 |
| Penbased  | 0.9736±0.011 | 0.9718±0.012 | 0.9710±0.013 | 0.9574±0.010 | 0.9726±0.016 | 0.9726±0.009 | 0.9754±0.013 |
| Contraceptive | 0.5478±0.031 | 0.5482±0.045 | 0.5566±0.034 | 0.4896±0.048 | 0.5492±0.045 | 0.525±0.031 | 0.5656±0.025 |
| Dermatology | 0.9864±0.019 | 0.9674±0.025 | 0.9864±0.019 | 0.9836±0.022 | 0.9756±0.015 | 0.9592±0.017 | 0.9836±0.022 |
| Thyroid   | 0.9072±0.022 | 0.4792±0.102 | 0.9334±0.014 | 0.9002±0.019 | 0.9280±0.019 | 0.3528±0.054 | 0.7518±0.047 |
the datasets in the same sequence as in Table 1, and column 'SUM' represents the total number of datasets on which CBHSID performed better than the compared method. Regarding G-Mean (Table 6a), CBHSID is better than ROS, SMOTE on 13 datasets and worst on only 3 datasets. In contrast, CBHSID is better than RUS, SMOTEENN, and SMOTE Tomek on 14 datasets and worse on only 2 datasets. In Recall (Table 6b), CBHSID performed better than ROS, SMOTE, SMOTEENN, and SMOTE Tomek on 13, 12, 15, and 14 datasets, respectively, whereas it completely outperformed RUS and Cluster-C on all 16 datasets. Generally, CBHSID outperformed all other methods on most of the datasets for all three evaluation metrics. The performance of algorithms varies on different metrics, and the same can be seen in our results. CBHSID outperformed all other methods in G-Mean on Thyroid dataset (column 16) but did not produce good results in terms of Recall and F1 score for the same dataset. Similarly, on Glass6 (column 7) and yeast3...
and RUS over the same metric. In contrast, there is no significant change in the performance of CBHSID, ROS, and Cluster-C over the G-mean metric. However, there is better performance than SMOTE, SMOTEENN, SMOTETomek, compared methods where CBHSID produced significantly better results than RUS, SMOTEENN, SMOTETomek, and Cluster-C for all three metrics. We performed statistical tests discussed in the following sub-section to verify whether the CBHSID performed significantly better than other methods.

1) STATISTICAL ANALYSIS

Results of all algorithms are statistically analyzed to identify the performance significance. We used the Friedman test [59] to compare the mean ranks of all the algorithms investigated across all experimental datasets. The Friedman test scores all of the compared algorithms independently on each dataset. The best algorithm receives a score of 1, the second-best receives a score of 2, and so on. In the case of a tie, we assign the average rank value.

The mean rank is the average value of the ranks of the algorithm across all datasets. As a result, a lower mean rank signifies higher performance. Suppose the null hypothesis, which states that all algorithms perform equally, is rejected. In that case, the Bonferroni-Dunn post-hoc test [60] is used to determine which algorithms are statistically distinct. Furthermore, we use the non-parametric Wilcoxon signed-ranks test [61] to analyze the performance difference between the two algorithms more precisely. The Wilcoxon test ranks the performance differences of two algorithms for each dataset. It compares the rankings while disregarding the signs for positive and negative differences, unlike the Friedman test, which is performed directly on the mean ranks of numerous algorithms.

Table 7 shows the mean ranks in G-mean, Recall, and F1score for the compared algorithms across all the experimental datasets. From Table 7, we can see that CBHSID exhibits the best mean ranks in terms of all metrics. It reveals that CBHSID produced the best results compared to other class imbalance handling techniques. The same table also presents Friedman test results which reject the null hypothesis for all three metrics G-mean, Recall, and F1score over alpha 0.05.

We proceed with the Bonferroni-Dunn post-hoc test. The critical difference (CD) in Bonferroni-Dunn test is 2.014 (α = 0.05). The results of CD diagrams are given in Figure 4, where the horizontal line in the CD diagram connects the methods with no significant performance difference. The methods which are not connected with the same line shows the significant performance difference. We are comparing CBHSID with other methods, so we discuss the performance of CBHSID mainly.

Figure 4(a) shows a significant change in performance of compared methods where CBHSID produced significantly better results than SMOTE, SMOTEENN, SMOTETomek, and Cluster-C over the G-mean metric. In contrast, there is no significant change in the performance of CBHSID, ROS, and RUS over the same metric.

| METHODS     | G-MEAN | RECALL  | F1SCORE |
|-------------|--------|---------|---------|
| ROS         | 3.81   | 3.22    | 2.91    |
| RUS         | 3.88   | 5.34    | 5.34    |
| SMOTE       | 4.34   | 3.03    | 2.94    |
| SMOTEENN    | 4.66   | 5.47    | 5.97    |
| SMOTETomek  | 4.56   | 3.78    | 3.97    |
| Cluster-C   | 4.84   | 5.50    | 5.28    |
| CBHSID      | 1.91   | 1.66    | 1.59    |

Table 7. Mean ranking and Friedman test results on G-mean, Recall, and F1 score.

Friedman Test

\[
\chi^2_{(\text{G-Mean})} = 20.774, p=0.002 (\text{Reject } H_0) \\
\chi^2_{(\text{Recall})} = 45.969, p=0.000 (\text{Reject } H_0) \\
\chi^2_{(\text{F1score})} = 53.232, p=0.000 (\text{Reject } H_0)
\]

Best mean ranks are highlighted in **bold**. The value \( \alpha = 0.05 \).

Figure 4(b) shows a significant change in performance of compared methods where CBHSID produced significantly better results than SMOTETomek, Cluster-C, RUS, and SMOTEENN over Recall metric. There is no significant change in the performance of CBHSID, ROS, and SMOTE over the same metric, hence are connected with same horizontal line.

Figure 4(c) shows that again there is a significant change in performance of compared methods where CBHSID produced significantly better results than RUS, SMOTEENN, SMOTETomek, and Cluster-C over the F1 score metric. Whereas there is no significant change among the performance of CBHSID, ROS, and SMOTE over the same metric. Figures 4(b) and 4(c) show a statistically significant difference in the performance of CBHSID versus RUS, CBHSID versus SMOTEENN, and SMOTETomek for recall and f1score metrics. At the same time, no significant difference was found among performances of CBHSID versus ROS and CBHSID versus SMOTE.

Table 8 shows Wilcoxon signed ranks test results for the pairwise comparison of CBHSID versus RUS, SMOTE, SMOTEENN, SMOTETomek, and Cluster-C for all three metrics. The R+ stands for the sum of ranks for the datasets on which CBHSID outperformed the other one. Whereas R− denotes the sum of ranks for the opposite and I stands for identical results. The symbols “**” and “*” represent that the considered algorithm is significantly better than the other at a significance level of 0.01 and 0.05, respectively. Since 16 datasets are used in this experiment, the critical values of the Wilcoxon test for \( \alpha = 0.01 \) and \( \alpha = 0.05 \) are 19 and 29, respectively. Table 8(a) shows that CBHSID performed significantly better than ROS on Gmean and F1score metrics for \( \alpha = 0.05 \), which is the opposite of the Bonferroni-Dunn test, which showed no significant change. However, for Recall, it shows no significant change, which is similar to the results of the Bonferroni-Dunn test. Table 8(b) shows that CBHSID performed significantly better than RUS on all three metrics.
On the Gmean metric for $\alpha = 0.05$, whereas on Recall and F1score metrics, CBHSID is better for both $\alpha = 0.05$ and $\alpha = 0.1$. These results are also slightly different from the Bonferroni-Dunn test, which showed no significant change between performances of CBHSID and RUS on Gmean.

CBHSID vs. SMOTE showed significant results on Gmean for $\alpha = 0.05$, and non-significance on Recall and F1score in Table 8(c), which is the same as achieved through the Bonferroni-Dunn test. Similarly, CBHSID performed significantly better than SMOTEENN and SMOTETomek on all three performance metrics for both $\alpha = 0.05$. For $\alpha = 0.1$ on Gmean, CBHSID is better than SMOTEENN, whereas, on F1score, CBHSID is better than both SMOTEENN and SMOTETomek. Lastly, CBHSID remained significantly better than Cluster-C on all three metrics, with $\alpha = 0.05$ on Gmean and with $\alpha = 0.01$ on Recall and F1score. Overall, CBHSID produced significantly better results than the other methods for most of the datasets on Gmean, Recall, and F1score performance measures metrics.

V. CONCLUSION
This paper introduced a novel cluster-based Hybrid Sampling approach for Imbalance Data to address the class imbalance.
problem for both binary and multi-class classification problems. In our approach, we first calculate the mean value, the average number of samples in each class. The same is further used as the threshold for data balancing. Next, we identify minority and majority classes based on the calculated threshold. The cluster analysis is applied to each class to convert it into sub-clusters. We calculate the distance of each data item from the centroid of the sub-cluster to identify the observations closer to the centroid. We achieve this by calculating the mean of every observation and selecting the observations whose mean is closer to the mean of the centroid observation. In case of over-sampling, we discard the rest of the data samples and keep only the data samples closer to the centroid. Whereas in the case of under-sampling, we generate synthetic data samples from the data observations near the center. The number of data observations to add (oversampling) or remove (undersampling) in any sub-cluster is calculated based on the size of that sub-cluster. We compared our proposed approach with six state-of-the-art data balancing methods on 12 binary and 4 multi-class datasets. The results of G-Mean, Recall, and F1 score show that our method outperformed the other compared methods on most of the datasets. On G-Mean metric, CBHSID is better than ROS, SMOTE on 13 datasets and CBHSID is better than RUS, SMOTEENN, and SMOTETomek on 14 datasets. On Recall metric, CBHSID performed better than ROS, SMOTE, SMOTEENN, and SMOTETomek on 13,12,15, and 14 datasets, respectively. Whereas it completely outperformed RUS and Cluster-C on all 16 datasets. In the same way, on F1score metric, CBHSID performed better than ROS, SMOTE, SMOTEENN, and SMOTETomek on 13,12,15, and 15 datasets, respectively, whereas it again completely outperformed RUS and Cluster-C on all 16 datasets. CBHSID produced best mean average rank among all the compared methods. The results were verified on the Friedman test, CBHSID produced best mean average rank among all the compared methods. It ranked first on all three performance metrics with 1.91 on Gmean, 1.66 on recall and 1.59 on f1 score. Bonferroni-Dunn test, and Wilcoxon signed ranks post-hoc tests, showing that CBHSID produced significantly better results than other methods. However on Yeast5, Yeast6, Dermatology and Thyroid datasets which are highly imbalanced, SMOTE performed better than CBHSID on Recall and F1 score performance metrics.

The performance of CBHSID can be improved further by:

i) Using different distance metrics in cluster analysis while calculating the distance of data item from the centroid of the sub-cluster.

ii) Apply feature selection and generate synthetic data using most relevant features.

iii) While under-sampling the majority class, after applying cluster analysis, the data items are removed from each sub-cluster based on its size. There could be a possibility that some sub-clusters may have very few data items and removing data items from those sub-clusters may loss its representation from the actual class. So, there should be a mechanism to overcome such situation.

iv) In current study, we only considered data samples which are closer to the centroid for over-sampling and under-sampling, because, we believe that there is very less possibility of presence of noisy data near the centroid. However, if in case noisy data is somehow present near the centroid that could lead to generate noisy data during resampling.

v) Another interesting situation to look into could be a class imbalance between sub-clusters and test our approach based on the complexity.

In current study, our proposed approach was not tested on noisy data which may affect the performance of the CBHSID. In our future work, we are going to look into the matters discussed above to see their effect on performance of CBHSID. We also plan to test CBHSID on non-stationary data streams in online learning environment.

ACKNOWLEDGMENT
The authors would like to thank Dr. Syed Muslim Jameel (Sir Syed University, Karachi, Pakistan) and Mr. Asad Mehmood (IBM software house, Islamabad, Pakistan) for their technical support during the implementation of the proposed algorithm.

REFERENCES

[1] M. A. Mazurowski, P. A. Habas, J. M. Zarada, J. Y. Lo, J. A. Baker, and G. D. Tourassi, “Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance,” Neural Netw., vol. 21, nos. 2–3, pp. 427–436, 2008.

[2] S. Fotouhi, S. Asadi, and M. W. Kattan, “A comprehensive data level analysis for cancer diagnosis on imbalanced data,” J. Biomed. Informat., vol. 90, Feb. 2019, Art. no. 103089, doi: 10.1016/j.jbi.2018.12.003.

[3] K. De Angeli, S. Gao, I. Danciu, E. B. Durbin, X.-C. Wu, A. Stroup, J. Doherty, S. Schwartz, C. Wiggins, M. Damesyn, L. Coyle, L. Penberthy, G. D. Tourassi, and H.-J. Yoon, “Class imbalance in out-of-distribution datasets: Improving the robustness of the TextCNN for the classification of rare cancer types,” J. Biomed. Informat., vol. 125, Jan. 2022, Art. no. 103957, doi: 10.1016/j.jbi.2021.103957.

[4] G. Du, J. Zhang, S. Li, and C. Li, “Learning from class-imbalance and heterogeneous data for 30-day hospital readmission,” Neurocomputing, vol. 420, pp. 27–35, Jan. 2021, doi: 10.1016/j.neucom.2020.08.064.

[5] W. Wei, J. Li, L. Cao, Y. Ou, and J. Chen, “Effective detection of sophisticated online banking fraud on extremely imbalanced data,” World Wide Web, vol. 16, no. 4, pp. 449–475, 2013.

[6] J. Meseguer, V. Puig, and T. Escobet, “Fault diagnosis using a timed discrete-event approach based on interval observers: Application to sewer networks,” IEEE Trans. Man, Cybern. A, Syst. Humans, vol. 40, no. 5, pp. 900–916, Sep. 2010.

[7] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, “Deep learning and its applications to machine health monitoring,” Mech. Syst. Signal Process., vol. 115, pp. 213–237, Jan. 2019, doi: 10.1016/j.ymssp.2018.05.050.

[8] F. Zhou, S. Yang, H. Fujita, D. Chen, and C. Chen, “Deep learning fault diagnosis method based on global optimization GAN for unbalanced data,” Knowl. Based Syst., vol. 187, Jan. 2020, Art. no. 104837.

[9] A. Oussous, F. Z. Benjelloun, A. A. Lahcen, and S. Belfikh, “Big data technologies: A survey,” J. King Saud Univ.-Comput. Inf. Sci., vol. 30, no. 4, pp. 431–448, Oct. 2018.

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

[11] B. Zhu, B. Baesens, and S. K. van den Broucke, “An empirical comparison of techniques for the class imbalance problem in churn prediction,” Inf. Sci., vol. 408, pp. 84–99, Oct. 2017.
[59] M. Friedman, “A comparison of alternative tests of significance for the problem of m rankings,” Ann. Math. Statist., vol. 11, no. 1, pp. 86–92, 1940.
[60] J. Demšar, “Statistical comparisons of classifiers over multiple data sets,” J. Mach. Learn. Res., vol. 7, pp. 1–30, Dec. 2006.
[61] A. Ghasemi and S. Zahediasl, “Normality tests for statistical analysis: A guide for non-statisticians,” Int. J. Endocrinol. Metabolism, vol. 10, no. 2, p. 486, 2012.

ABDUL SATTAR PALLI received the M.Sc. degree in information technology from SZABIST, Islamabad, Pakistan. He is currently pursuing the Ph.D. degree with Universiti Teknologi PETRONAS, Malaysia. He also works as an Assistant Director (Information Technology) at the Department of Anti-Narcotics Force, Ministry of Narcotics Control, Pakistan. His research interests include artificial intelligence, machine learning, and data stream mining.

JAFREEZAL JAFAAR (Senior Member, IEEE) received the B.Sc. degree in computer science from Universiti Teknologi Malaysia, in 1998, the M.App.Sc. degree in IT from RMIT University, Australia, in 2002, and the Ph.D. degree from the University of Edinburgh, U.K., in 2009. He is currently an Associate Professor and the Dean at the Faculty of Science and Information Technology, Universiti Teknologi PETRONAS (UTP), Malaysia. He joined the Department of Computer and Information Sciences, UTP, in 1999, where he was the Former Head, from 2012 to 2016. He is also the Head of the Center for Research in Data Science (CERDAS), which focus on implementation of AI and machine learning in Oil and Gas (O&G) Industry. He is also active in conducting professional training in AI and big data analytic for public and industry. His main research interests include AI, machine learning, and data analytics, with over 100 technical publications. Based on his experience and expertise, he had become a member of the Academy Science Malaysia SIG in machine learning, the Executive Committee for IEEE CS Malaysia Chapter (2016–2018), and the Executive Committee for MyAIS (2017–2019).

MANZOOR AHMED HASHMANI (Senior Member, IEEE) received the Ph.D. degree in communication networks from the Nara Institute of Science and Technology, Japan. He is currently an Associate Professor with the Department of Computer and Information Sciences, Universiti Teknologi PETRONAS. He is also a member of the High Performance Cloud Computing Centre and the Centre for Research in Data Science. His research interests include computer communications (networks), artificial neural networks, and artificial intelligence.

HEITOR MURILO GOMES received the Ph.D. degree in computer science from Pontifícia Universidade Católica do Paraná, Brazil, in 2017. He is a machine learning researcher and a data scientist. He is currently a Lecturer (an Assistant Professor) in AI at the Victoria University of Wellington (New Zealand). His research interests include machine learning, data stream mining, big data analytics, and artificial intelligence.

ABDUL REHMAN GILAL received the Ph.D. degree in information technology from Universiti Teknologi PETRONAS (UTP), Malaysia. He has been mainly researching in the field of empirical software engineering and cybersecurity for creating the methods and techniques for efficient and secure software development. Based on his research publication track record, he has contributed in the areas of human factor in software development, complex networks, databases and data mining, programming, and cloud computing.

***