Improved Adaptive Semi-Unsupervised Weighted Oversampling using Sparsity Factor for Imbalanced Datasets

Haseeb Ali[1], Mohd Najib Mohd Salleh[2*]
Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia
Batu Pahat, Malaysia

Kashif Hussain[3]
Institute of Fundamental and Frontier Sciences
University of Electronic Science and Technology of China
Chengdu, China

***Abstract*—With the incredible surge in data volumes, problems associated with data analysis have been increasingly complicated. In data mining algorithms, imbalanced data is a profound problem in machine learning paradigm. It appears due to desperate nature of data in which, one class with a large number of instances presents the majority class, while the other class with only a few instances is known as minority class. The classifier model biases towards the majority class and neglects the minority class which may happen to be the most essential class; resulting into costly misclassification error of minority class in real-world scenarios. Imbalanced data problem is significantly overcome by using re-sampling techniques, in which oversampling techniques are proven to be more effective than undersampling. This study proposes an Improved Adaptive Semi Unsupervised Weighted Oversampling (IA-SUWO) technique with sparsity factor, which efficiently solves between-the-class and within-the-class imbalances problem. Along with avoiding over-generalization, overfitting problems and removing noise from the data, this technique enhances the number of synthetic instances in the minority sub-clusters appropriately. A comprehensive experimental setup is used to evaluate the performance of the proposed approach. The comparative analysis reveals that the IA-SUWO performs better than the existing baseline oversampling techniques.**

***Keywords*—Data mining; imbalanced data; minority; majority; oversampling**

I. INTRODUCTION

At present, data mining tasks involve large amount of data which is complex and embedded with noise; hence techniques used for information extraction are needed to be efficient and effective decisions making [1][2]. In data mining, classification is the most commonly performed data analysis task in real-world applications including medical, engineering, and business [3][4][5]; i.e., cancer prediction [6], face detection [7], software fault detection [8][9], bankruptcy prediction [10][11], fraud detection [12]. Majority of classification algorithms consider that the given dataset has the proportional instances among the classes, and these algorithms are not intelligent enough to detect the inappropriate distribution of instances. However, in many practical applications, it is often found that one class known as majority class contains the number of instances to a great extent – intensively dominating the other class which has only a small number of instances; known as minority class. This phenomena is considered as imbalanced data problem [13][14]. Due to imbalanced distribution of instances among the classes, the classifiers show biased behavior to present accuracy according to the majority class while neglecting the essential minority class [15][16]. For example, phishing email dataset actually presents the imbalanced data set in which for each 1 million emails, only 30 emails present the phishing email. For this scenario, the classifier may show unconscious bias towards majority class. If the classifier neglects the minority class, it will result into higher misclassification cost [17][18].

Imbalanced data also have other aspects like desperate distribution of data in the feature space in imbalanced datasets, and these datasets usually have some problematic characteristics as the overlapping of data instances, presence of noise, small disjuncts, and small sized instances [19][20][21]. Also, another kind of data imbalances, which is present with in the class is known as within-class imbalanced problem that result in the performance loss [22][23]. Therefore, it is very difficult for trivial classifiers to predict minority class correctly. In this context, the analysis or prediction accuracy of any classifier for the minority class becomes significantly critical in real world domain which encounter the imbalanced data problem such as, affected vs. non-affected cases in several diseases predictions, non-bankruptcy vs. bankruptcy in bankruptcy predictions, and fraud detection in credit card (fraudulent vs. non fraudulent cases) [1][24][25].

Importance of imbalanced data problem is owned by worldwide researchers hence they proposed many exceptional approaches to tackle this problem. These contributions made for imbalanced data problems can be divided into three categories: data level approach, algorithmic approach, and cost sensitive approach [14][26]. Data level approaches work as the preprocessing of the data before learning process by using resampling techniques which are independent of the classifier [17]. Algorithmic approaches embrace new algorithms or modify the existing ones for imbalanced data problem [27]. Last one is the cost sensitive approach which minimizes the total cost of errors in data level or algorithmic level approach [15]. Data level approaches, also known as external level methods, are more effective in handling the class imbalance problem, as these approaches perform preprocessing of data, whereby data is modified before the learning process [28].

*Corresponding Author..

These methods manipulate the data externally by balancing the distribution of samples among the classes [17][15].

Data level approaches either use the oversampling methods in which artificial data is deliberately generated in the minority class, or the undersampling methods where data is eliminated from the majority class; in order to balance the distribution of minority class. However, removing the data from the majority class may also eliminate the potential data with it, which can be used for the learning process [14]. On the other hand, oversampling methods are preferred more by the researcher's community because there is no risk of losing any useful data. Despite this, in oversampling methods, generating exact replication of data may create overfitting in the data, where selection of incorrect instances in the data for oversampling may generate new instances that might fall in the incorrect region – giving rise to overlapping with other instances that belong to other class, causing overfitting and over-lapping of samples and deteriorating the performance of classifiers [20]. To overcome this, various methods have been proposed in the related literature. The effective approaches are clustering-based approaches in which the input space is partitioned into the clusters then the oversampling technique is applied [29]. In some clustering-based approaches, majority and minority classes are clustered separately; possibly when minority class instances shows small disjoints, so it needs to make several minority sub-clusters [30]. The number of instances vary differently in minority sub-clusters, and it raises within-class imbalance problem because it is necessary to oversample all minority sub-clusters; otherwise the classifier biases towards the oversampled class [23].

In this paper, an improved oversampling technique is proposed, so-called Improved Adaptive Semi-Unsupervised Weighted Oversampling (IA-SUWO) with sparsity factor for class imbalance problem. IA-SUWO clusters the minority class instances and assigns higher weights to the minority instances which are closer to majority instances, in order to manage hard-to-learn minority instances. Secondly, it also considers every minority sub-cluster for oversampling along with small concepts which are far from the majority clusters and ignored by several other techniques (Fig. 1). The proposed approach avoids over-lapping between the synthetic minority instances and majority instances by using the semi-unsupervised clustering approach that significantly avoids majority class clusters to come in-between two minority clusters that need to be merged. It assigns weights to minority instances for oversampling according to average Euclidean distance of minority instances from the instances of majority class, in addition to decrease more chances of over-generalization. Moreover, it also assigns weights according to the sparsity factor of the minority instances in each sub-cluster to enhance the learnability of the classifier for the minority class instances which are sparse apart. The IA-SUWO technique identifies the sub-clusters misclassification error and assigns sizes of sub-cluster appropriately based on their complexity in being misclassified. In order to validate the proposed IA-SUWO technique, a comprehensive experimental setup is performed. Three publicly available datasets are used to evaluate the proposed method after the classification process on the two classifiers. Precision, F-measure, and ROC are used as the performance measures. Outcome of this whole experimental setup is compared to state of the art with four other existing techniques.

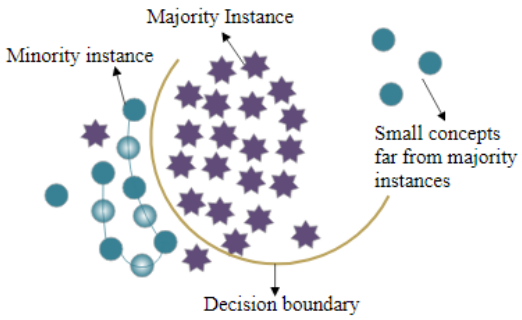

Fig. 1. Small Concepts of the Minority Class that are Far from the Majority Class are neglected and not oversampled.

The remaining paper is organized as follows: Section II presents the related works from recent literature. The detail of the proposed technique is given in Section III, while the research methodology adopted for this study is explained in Section IV. Section V provides the experimental results for analysis and discussion. The study is duly concluded in Section VI which also highlights potential future directions pertaining to relative research line.

II. Related Work

As discussed earlier in this paper, the data level approaches are effective in balancing the data distributions. Resampling methods used for the preprocessing of data can be categorized into two major types: oversampling and undersampling. This current research is intended to focus on the oversampling techniques (Fig. 2) for imbalanced data problem, therefore a brief overview of existing oversampling techniques, proposed in recent literature, is presented.

Oversampling methods are further categorized into random and informed methods [14]. Random Oversampling (ROS) is the pioneer and simplest technique used for oversampling. This technique randomly generates synthetic instances until the desired ratio. However, in spite of its ease in implementation and its simplicity, it encounters a major drawback that it generates exact replication of the original minority instances, which often results in over fitting [26] [31]. Addressing this problem, Nitesh Chawla proposed first informative method proposed for imbalanced data problem in 2002, named as Synthetic Minority Oversampling Technique (SMOTE) [26]. In which synthetic minority instances are generated by linear interpolation of two neighboring instances. SMOTE generates minority instances between randomly selected minority instance and its nearest neighbor. SMOTE produces new minority instances with the same primary number of original instances. However, some of these generated new instances fall into the incorrect region and overlapped with the instances of the other class result into over generalization [12] (Fig. 2(i)).

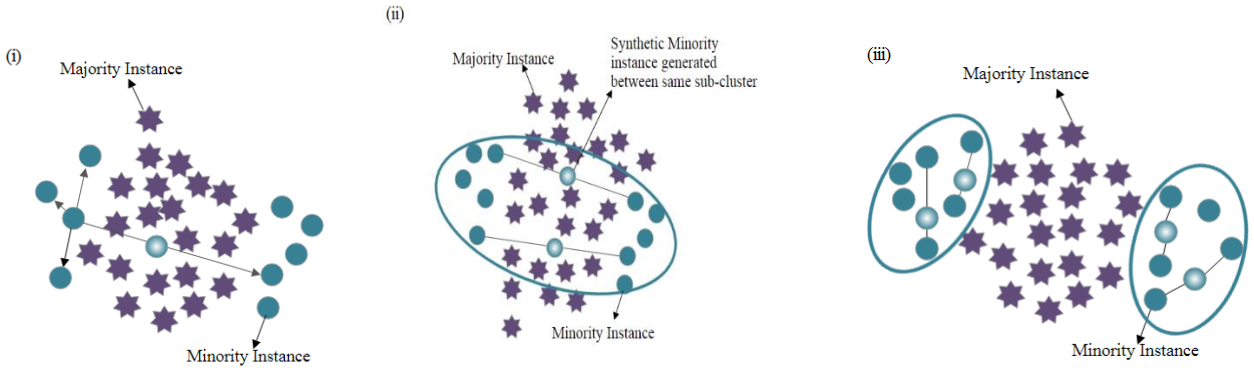

Fig. 2. Presenting the different Oversampling Scenarios in which, (i) Synthetic Instances Generated between Selected Instance and its 4 Nearest Neighbors; (ii) Synthetic Instances are generated between Selected Instance and its Nearest Neighbors from the Same Cluster; (iii) Synthetic Instances are generated between Selected Instance and its 4 Nearest Neighbors that belong to the Same Cluster, which Avoids Overlapping.

Informed methods identify the most effective areas for the oversampling like, Safe-level SMOTE [32] is a modified SMOTE algorithm which identifies the safe level value for each minority instance and applies weight degree. It oversamples in only safe areas by differentiating between noisy and safe instances. There are some other approaches which focus on the class regions for oversampling, as Borderline-SMOTE [33] which determines borderline among the two classes then generates synthetic minority instances near the decision boundary. This technique chooses the targeted instances which are close to the borderline. Considering class regions, Cluster-SMOTE [34] is another approach which focuses on the certain regions for oversampling. In this regard, it partitions the data into clusters. This technique first uses the *k*-means to cluster minority class then it employs SMOTE within those naturally occurring minority clusters. Moreover, unlike SMOTE which produces new instances equal in number to original instances. Techniques like ADASYN [35] and its variation KernalADASYN [36] are proposed with objective to choose hard-to-learn instances. By employing the weight assigning approach, it assigns more weights to these instances for oversampling. The basic reason behind this approach is to priorities the minority instances to avoid their misclassification and to enhance the number of synthetic instances to be generated [37].

Majority Weighted Minority Oversampling technique (MWMOTE) [37] uses clustering approach, which partitions the majority and minority clusters separately and then assign weights to minority instances according to the Euclidean distance of the minority instances from majority instances. However, it neglects the small disjoints which are far from the majority instances (Fig. 1). Presence of small disjoints of minority class results in within-class imbalance problem. It is necessary to oversample all the minority sub-cluster; otherwise, the classifier biases towards the oversampled ones. Adaptive Semi Unsupervised Weighted Oversampling (A-SUWO) [23] which uses semi-unsupervised clustering approach to cluster the minority class instances to avoid over-generalization. Secondly, in order to oversample all the minority sub-cluster including those small concepts that are far from the majority class, this technique measures the misclassification error rate to determine the complexity of each minority sub-cluster in being misclassified. Later, it assigns larger size to those sub-clusters which have higher misclassification error rate. Enhanced Minority Oversampling Technique (EMOTE) [38] enhances the minority class distribution by generating new instances in their neighborhood in order to improve the classifier performance. It effectively improves the classification results by tuning the wrongly classified instances into correctly classified instances by using its proposed oversampling approach. For imbalance learning, an Evolutionary Cluster-Based Synthetic Oversampling Ensemble (ECO-Ensemble) method [39] creates an ensemble by combining an evolutionary algorithm (EA) with a new clustering based synthetic data generation method. In this method, regions for oversampling of minority instances are identified by using the clustering approach based on the modern ideas. The EA benefits in lowering the overall computational cost and in optimization of parameters for data generation method. Self-Organizing Map-based Oversampling (SOMO) [29] is another technique that uses self-organizing map to convert input data into two dimensional space. It generates synthetic instances into effective areas. It also uses SMOTE to generate synthetic minority instances into clusters which are found in the lower dimensional space. SOMO alleviates between class and within-class imbalances problem. *k*-means SMOTE [17] is another technique which uses density factor for the data generation. After clustering the input data and finding sparsity factor among all clusters, it generates synthetic samples by using SMOTE according to the weight assigned based on the density of the clusters.

A Radial-Based Oversampling (RBO) [19] tackles the noisy imbalanced data classification problem. This method generates minority data instances into the rightful regions according to their imbalance ratio calculated by radial based functions, also removes noise from the data effectively. A robust oversampling technique, proposed in [40], for imbalanced data learning which uses Gaussian Mixture Model (GMM) to balance the distribution of instances in both classes. This method considers high dimensional feature space for generating synthetic instances and GMM determines and filters out outlier instances from the minority class. An exclusive technique for ordinal regression imbalanced problem also uses oversampling approach for data generation in minority class based on the weights assigned. This synthetic minority oversampling for ordinal regression (SMOR) considers

generation direction for each candidate [41]. Another oversampling technique for imbalances problem in ordinal regression is proposed which is adaptive structure based. It enhances the data generation process in minority class after exploration of immature and complex minority instances [42].

III. Improved Adaptive Semi-Unsupervised Weighted Oversampling (IA-SUWO)

This research proposes an improved oversampling technique, so-called IA-SUWO. The proposed technique significantly enhances the learnability of the classifier and improves its accuracy by using sparsity factor for assigning weights to overcome the limitation of the conventional A-SUWO. Standard A-SUWO assigns weights to instances for oversampling according to their Euclidean distance from majority samples (Fig. 3). Here, weights are assigned to those instances which are closer to majority instances or in front of majority clusters. However, it neglects the instances that are far from majority instances or located behind in the minority sub-clusters. Minority instances contain potential data about minority class or could be neglected by the classifier. Therefore, if weights are assigned according to sparsity of each minority cluster, it can assign weights appropriately. Sparsity factor finds the sparse minority instances in each cluster by measuring its density. Synthetic instances are generated according to both approaches of assigning weights to significantly alleviate the between-the class and within-class imbalance problem.

A. Sparsity Factor for Oversampling in IA-SUWO

Sparsity factor in a cluster can be defined as the measure of sparse instances that are found to be scattered in a particular cluster or where density of instances in the cluster is low. This research improves the conventional A-SUWO by assigning weights to minority instances according to sparsity factor for oversampling. Those minority instances present in the minority clusters which are sparse apart and far from the majority instances assumed as incompetent but certainly they are not. Hence, after assigning weights based on their Euclidean distance from majority instances, the proposed model looks forward for the density of each cluster to measure the sparsity factor. Density of each cluster is measured by dividing the number of minority instances present in each sub-cluster by the average distance between them raised to the power of features count *m*. Density is inversely proportional to the sparsity; therefore, sparsity factor is measured as the inverse of density. After measuring sparsity for each filtered minority cluster, sum of all the sparsity measures is taken, which is then transformed into the sampling weights. These sampling weights are assigned to instances of all minority filtered cluster which determine the expedient distribution of instances to be generated in each cluster. In this way, instances which present lower density measure attain more weights for oversampling. It is noted that these sampling weights are based on the comparative analysis of density measure of each cluster, which determines the cluster density as compared to others on average. It considers only the distance between the minority instances while measuring the density of minority clusters.

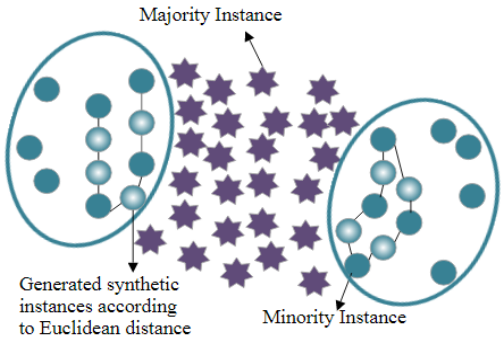

Fig. 3. Synthetic Instances are Generated According to the Euclidean Distance from Majority Instances.

B. Semi-unsupervised Clustering

Before clustering, this technique finds the noisy instances and removes them from the dataset based on the method proposed by [33]. This is done in order to identify if any instance is noisy, and also to determine its *NN* nearest neighbors. If its neighbors belong to another class, this instance is declared to be noise hence removed from the dataset. After removing the noise from dataset, it uses hierarchical clustering to cluster the majority class first, that results into *m* majority sub-clusters $Cmaj_{i=1,\dots,m}$. Later, all the minority sub-clusters are allocated to minority sub-clusters $Cmin_{j=1,\dots,n}$ based on the semi unsupervised hierarchical clustering proposed by [23] to avoid overlapped generated minority instances. This algorithm is based on the complete-linkage agglomerative hierarchical clustering, but it checks overlapping in each iteration between the two nominated minority sub-clusters say $Cmin_a$ and $Cmin_b$ selected for merging [43]; because this algorithm is modified as it uses information about majority class sub-clusters. It is therefore, the proposed approach is not fully unsupervised but semi-unsupervised. Here, the two closest minority sub-clusters are not allowed to be merged if any majority class sub-cluster exists between them. While in the case when there is not any majority sub-cluster between these two minority sub-clusters, and their distance is also less than the pre-defined threshold, they are allowed to be merged.

This approach avoids overlapping of synthetic minority instances with majority instances significantly by avoiding majority cluster to come in between. The semi-unsupervised hierarchical clustering algorithm has the following steps:

- Each minority instance is assigned to an individual sub-cluster which results into *n* sub-clusters of minority class of size B = $Cmin_{j=1,\dots,n}$.
- Two minority sub-clusters that are identified can be $Cmin_a$ and $Cmin_b$ for merging with lowest Euclidean distance л.
- Determine the majority sub-clusters $C_{maj_i} \in A$, (where A belongs to a set of majority class) which have Euclidean distance between $Cmin_a$ and $Cmin_b$ lesser than π.

- Majority sub-cluster exists between $Cmin_a$ and $Cmin_b$ if this $A \neq \emptyset$; in this case these minority clusters should not be merged. In order to avoid these sub-clusters to be considered for merging again, distance between $Cmin_a$ and $Cmin_b$ is set to a large number.
- Otherwise, these minority sub-clusters are merged and become a new sub-cluster $Cmin_c$.
- These steps are again repeated for every newly created sub-cluster $Cmin_c$, and these steps are repeated unless the Euclidean distance between any two closest sub-clusters of minority class becomes lesser than a threshold *T*. Finally, in this way *n* Minority sub-clusters are formed.

In order to find a better estimated value for *T*, this algorithm measures Euclidean distance d_{med} among all the *h* instances of majority and minority class by using Eq. (1).

$$T = d_{avg} \times c_{thres} \quad (1)$$

where c_{thres} is a user-defined constant parameter and its optimum value depends on the dataset. The greater value of c_{thres} results into larger cluster size which can result in increasing the chance of over-lapping, while the smaller value for c_{thres} may result in small cluster sizes which can cause overfitting or less diverse synthetic samples generation. Optimum value for c_{thres}, is in the range of [0.7, 2.0]. It depends on the dataset, for examples, for wine dataset the best c_{thres} value is 1.0.

C. Adaptively Sub-Cluster Sizing

Mostly, all sub-clusters present in dataset have similar sizes after the oversampling by using the existing clustering-based techniques. However, the sub-clusters that have higher chances of misclassification like those small concepts that are far from the majority class, need larger sizes and more oversampling. Though, some sub-clusters have lower chance of misclassification and do not need much oversampling. For this scenario, A-SUWO resets the sizes of all sub-clusters according to their misclassification rate, which means the misclassification of the related instances results into two achievements: firstly, larger size is assigned to sub-cluster which is prone to more miss-classified, and secondly, it balances the ratio of samples in both classes. This is done by using cross validation in Linear Discriminant Analysis (LDA), which calculates the complexity or misclassification rate of every sub-cluster. For the classification, each of the *n* minority sub-cluster is partitioned into *k* similar sized divisions Fig. 4.

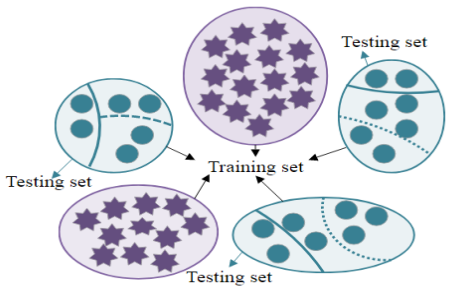

Fig. 4. Split Minority Sub-Clusters into Testing and Training Partitions.

All majority instances from all sub-clusters and *k-1* divisions from each minority sub-cluster is used for training purpose, whereas all majority sub-clusters and one division that is remained in each minority sub-cluster are used for testing purpose. LDA is a simple classifier and it needs no parameters to be tuned, it runs *k* times for cross validation and measures the misclassification error rate. The misclassification error ε_{jk} is calculated in each fold *f*, for each minority sub-cluster *j* for the number of instances that are wrongly classified as the majority instance while testing. This misclassification error ε_{jk} is divided by total number of instances present in each sub-cluster R_j to calculate error rate ε_{jk}. By taking average of all error rates, this result into average error rate $\bar{\varepsilon}_{jk}$ of all minority sub-clusters for all folds.

Using the following equation the standardized average error rate $\hat{\varepsilon}_j$ is calculated by standardizing $\bar{\varepsilon}_{jk}$.

$$\hat{\varepsilon}_j = \frac{\bar{\varepsilon}_j}{\sum_{j=1}^{n} \bar{\varepsilon}_j} \quad (2)$$

In order to get the final sizes of any two minority sub-clusters say L_1 and L_2 should have equivalent ratio to their average error rates $\hat{\varepsilon}_{L_1}$ and $\hat{\varepsilon}_{L_2}$ as Eq. (3):

$$\frac{S_{L_1}}{S_{L_2}} = \frac{\hat{\varepsilon}_{L_1}}{\hat{\varepsilon}_{L_2}} \forall\, L_1, L_2 \in \{1, \dots, n\} \quad (3)$$

Here, for the final sizes of minority sub-clusters L_1 and L_2, S_{L_1} and S_{L_2} are the final sizes after oversampling respectively. While, for the L_1 and L_2, the standardized average error rates are $\hat{\varepsilon}_{L_1}$ and $\hat{\varepsilon}_{L_2}$, respectively.

D. Synthetic Instance Generation

Before A-SUWO, there were many existing techniques which generate synthetic instances in safe areas, borderline, and in those clusters that are near the majority class. Likewise, in the MWMOTE, there might be some sub-clusters that are distant from the instances of majority class and neglected completely, which means they are not oversampled. This causes within-class imbalance problem and classifier becomes biased towards oversampled ones. Some of the techniques that generate synthetic instances between the candidate instance and its nearest neighbor which belongs to another sub-cluster, produce the overlapping of generated instance with majority instances (Fig. 2(i)). Similarly, generating the instances between candidate instance and its nearest neighbor from the same sub-cluster but far from it, also results into overlapping (Fig. 2(ii)). A-SUWO overcomes these problems by generating the synthetic instances between the primary instance and its nearest neighbor within the same sub-cluster to avoid overlapping of instances (Fig. 2(iii)). Secondly, it oversamples all the minority sub-clusters (smaller ones) that are far from the majority instance which reduce the within-class imbalance problem. However, A-SUWO generates instance using Euclidean distance of minority instances from the majority instances and assigns weights to those instances of minority sub-clusters that are nearer to majority instances. This causes the vacant space and ignorance of those instances that are far away from the majority instances or at the back within the same sub-cluster, even if they carry the important information about the minority class. IA-SUWO improves this method by

using the sparsity factor and assigns weights adaptively to all instances within each individual minority sub-cluster that results into the appropriate number of instances generated within each sub-cluster and oversamples every sub-cluster to its required extent (Fig. 5).

For oversampling, probability distribution of instances is derived from the weight assign to instances of the minority class. This weight is assigned according to two approaches: according to the Euclidean distance from majority samples and sparsity factor. At first, weights are assigned according to the Euclidean distance from the majority instances. Purpose of assigning weights according to the Euclidean distance is that, the minority instances which are found to be more nearer to decision boundary or majority instances have higher chances of being misclassified. Therefore, in order to assign weight to minority instances in the minority sub-cluster C_{min}, find the k nearest neighbor for the h^{th} minority instance x_{jh}, according to its Euclidean distance from the majority instance $y_{jh(v)}$, and measure this distance $d(x_{jh}, y_{jh(v)})$, where this $v = 1 \dots, k$ which implies the indices of the nearest neighbors. This distance $d(x_{jh}, y_{jh(v)})$ is normalized by dividing it by the total number of features D to make it robust to the datasets used with several number of features, see Eq. (4):

$$\hat{d}(x_{jh}, y_{jh(v)}) = \frac{d(x_{jh}, y_{jh(v)})}{D} \quad (4)$$

Then, define the closeness factor as $\Gamma(x_{jh}, y_{jh(v)})$ between x_{jh} and $y_{jh(v)}$ using Eq. (5):

$$\Gamma(x_{jh}, y_{jh(v)}) = f_i\left(\frac{1}{\hat{d}(x_{jh}, y_{jh(v)})}\right) \quad (5)$$

where f_j is defined as a cutoff function, which is used for avoiding $\frac{1}{\hat{d}(x_{jh}, y_{jh(v)})}$ to become extremely large in the situation when two instances x_{jh} and $y_{jh(v)}$ in a sub-cluster C_j become quietly close to each other. Hence, f_j can be expressed as Eq. (6):

$$f_j(x) = \begin{cases} x \; if, x \le TH_i \\ TH_j, otherwise \end{cases} \quad (6)$$

Here, the largest value f_j (x) can achieve TH_j, whereas TH_j is automatically determined for every sub-cluster C_j. This is gained by measuring the Euclidean distance of all minority instances x_{jh} in every sub-cluster to their nearest majority instance $y_{jh(1)}$ and after this identify $f\left(\frac{1}{\hat{d}(x_{jh}, y_{jh(1)})}\right)$, TH_j is set as average of $f\left(\frac{1}{\hat{d}(x_{jh}, y_{jh(1)})}\right)$, using Eq. (7):

$$TH_j = \sum_{j=1}^{R_j} f\left(\frac{1}{\hat{d}(x_{jh}, y_{jh(1)})}\right) \quad (7)$$

where R_j present the number of instances in sub-cluster C_j. As aforementioned TH_j is determined automatically, but this is really critical, because the weighting algorithm runs separately for each sub-cluster and a specific threshold is required to each sub-cluster.

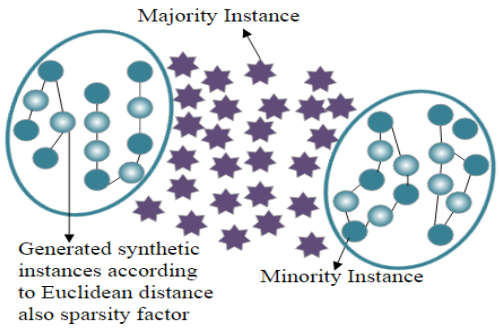

Fig. 5. Synthetic Instances are generated According to Euclidean Distance and Sparsity Factor.

In order to assign higher weights to the instances that are closer to the majority instances, Eq. (5) present the reciprocal of $\Gamma(x_{jh}, y_{jh(v)})$. At last, the weights $W(x_{jh})$ are decided according to the Euclidean distance of minority instance (x_{ij}) from the all nearest neighbors, using Eq. (8):

$$W(x_{jh}) = \sum_{v=1}^{k} \Gamma(x_{jh}, y_{jh(v)}) \quad (8)$$

This weight is transformed into the probability distribution for oversampling along with the weights assigned according to the proposed approach that is based on the sparsity factor. As the sparsity factor is proposed in this paper for assigning weights to the minority instances. For determining the sparsity factor, the proposed approach measures the density of each minority sub-cluster, because inverting the density results in the sparsity measure. Therefore, density is measured as we first determine the mean Euclidean distance among all the minority instances, which is measures by finding the distance between each minority instance present in that sub-cluster. This distance is used for measuring the density as Eq. (9):

$$density(f) = \frac{minority\ count(f)}{average\ minoirty\ distance(f)^m} \quad (9)$$

where density is inversely proportional to the sparsity hence, inverse of density will be equal to the sparsity factor that can be measured as Eq. (9):

$$sparsity(f) = \frac{1}{density(f)} \quad (10)$$

After measuring sparsity for each filtered minority cluster, we can take sum of all the sparsity measures and this sparsity sum is then transformed into the sampling weights as Eq. (11):

$$Sparsity\ sum = \sum_{f \in filteredcluster} sparisty\ factor(f) \quad (11)$$

As the sparsity sum is calculated which is then used in weight formula as Eq. (12):

$$W(f) = \frac{sparsity\ factor(f)}{sparsity\ sum} \quad (12)$$

Consequently, these sampling weights are assigned to instances of all minority filtered cluster which determine the expedient distribution of instances to be generated in each cluster. In this way, instances which present lower density measure attain more weights for oversampling. It is noted that these sampling weights are based on the comparative analysis of density measure of each cluster, which determines the

density of a cluster as compared to others on average. It considers only the distance between the minority instances while measures the density of minority clusters.

To get the probability distribution $P(X_{jh})$, we sum that the both weights are as Eq. (13):

$$W(X) = W(f) + W(jh) \tag{13}$$

At last, these weights are transformed in the probability distribution $P(X_{jh})$ of synthetic samples by using Eq. (14):

$$P(x_{jh}) = \frac{W(X)}{\sum_{h=1}^{r_j} W(x_{jh})} \tag{14}$$

Finally, in order to obtain the each cluster size up to S_j, each sub-cluster C_j, $j = 1, \dots, n$ is oversampled according to the probability distribution of weights assigned by both approaches to minority instances. For this purpose, an instance 'a' is selected randomly from the probability distribution in any particular sub-cluster, then instance 'b' from its nearest neighbor is selected which belongs to that same sub-cluster and a new instance 'c' is generated between these selected instances *a*, and *b* like as Eq. (15):

$$c = \beta a + (1 - \beta)b \tag{15}$$

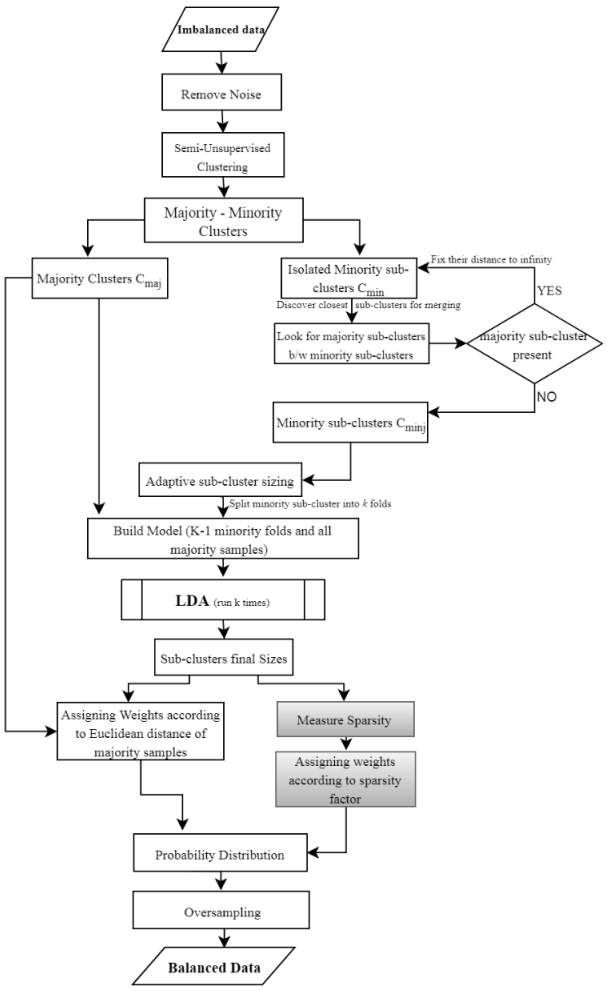

Fig. 6. Flow Diagram of the Proposed Method.

where, β is a random number between [0, 1]. Synthetic instances are generated in each minority sub-cluster, these instances are generated among the instances which are closer to the majority instances only. Proposed technique is summarized in the flow diagram shown by Fig. 6.

IV. Research Methodology

At first, for developing the proposed model, datasets are used for the experimental purpose to evaluate the performance of the proposed model. For performance evaluation of the proposed oversampling technique, some standard benchmark datasets are used that contain binary or multi-classes in which, number of samples in the classes varied exceptionally. Those datasets which have more than two classes are converted into binary-class dataset, labeling the smallest class as minority class and a bigger one as majority class. These datasets are examined for selection on the basis of the number of features, the total number of samples in both classes, number of minority samples, number of majority samples and their imbalance ratio.

IA-SUWO Algorithm

1. Semi-unsupervised clustering
   i. Noisy samples eliminated from the dataset.
   ii. Evaluate *T* by using Eq. (1).
   iii. Make majority class *m* sub-clusters $Cmaj_{i=1,\dots,m}$.
   iv. Each minority sample is assigned to an isolated sub-cluster.
   v. Discover the closest sub-clusters $Cmin_a$ and $Cmin_b$.
   vi. Look for any majority sub-cluster if overlapping between $Cmin_a$ and $Cmin_b$.
   vii. If any majority sub-clusters exist, set the distance equal to infinity and go to step (1.v). Otherwise, combine $Cmin_a$ and $Cmin_b$ and form one new sub-cluster $Cmin_c$.
   viii. Repeat steps 1.v to 1.vii until Euclidean distance among the closest sub-cluster is less than *T* threshold.
2. Adaptive sub-cluster sizing
   i. Split randomly every sub-cluster of minority class into *k* divisions.
   ii. Use *k*-1 divisions from each minority sub-cluster as a training set, and construct a model in addition to all majority instances.
   iii. Use one portion which is remained in each minority sub-cluster for testing of the model.
   iv. Evaluate Average Minority Standardized Error Rate $\hat{\varepsilon}_j$.
   v. Repeat *k* times, steps (2.ii) to (2.iv).
   vi. Identify the final sizes S_j for all sub-clusters $Cmin_{j=1,\dots n}$ by using Eq's. (2) and (3).
3. Assigning weights for synthetic samples generation within each minority sub-cluster.
   (a) By determining the Euclidean distance from majority samples for each sub-cluster $j = 1,2, \dots, n$:
      i. Find nearest neighbors *NN* surrounded by majority samples, for all the minority samples X_{jh} in sub-cluster $Cmin_j$.
      ii. Evaluate $W(X_{jh})$ by estimating TH_j, for each minority sample in $Cmin_j$ by using Eqs. (4) to (8).
   (b) By using the sparsity factor, for each cluster, compute sampling weights based on each sub-clusters density.
      i. For each filtered cluster *f*, find average distance (f) ← mean (Euclidean distances (*f*)) between the minority samples.
      ii. Determine the density measure by using Eq. (9).
      iii. Get the measure of sparsity by inverting the density measure using Eq. (10).
      iv. Determine the sampling weight of each sub-cluster as the sparsity factor of sub-cluster divided by the sum of all sub-cluster sparsity measures using Eqs. (11) and (12).
      v. Convert these weights for probability distribution $P(X_{jh})$ by Eqs. (13) and (14).
4. Oversample minority instances: First initialize the $O = I$, then for each sub-cluster $j = 1,2, \dots, n$:
   i. Select a minority instance 'a' in sub-cluster j from $P(X_{jh})$ for oversampling.
   ii. Randomly select one of its nearest neighbor *NN* 'b' from the same sub-cluster.
   iii. Produce a new synthetic instance *'c'* using Eq. (15) between *'a'* and *'b'*, and add 'c' to set O.
   iv. Repeat steps from (4.i) to (4.iv) until the sub-cluster size reaches S_j.

A. Datasets

This research used three main datasets for the experimental setup namely IRIS, WINE, and GLASS. These three datasets are commonly used by every researcher for the evaluation of their proposed approaches for imbalanced data problem. There are many benchmark datasets present in UCI directory, but these datasets are more simple and convenient to evaluate the performance of proposed model, as well as, better judgement of their impact on the dataset and learning process. Brief description of these datasets is as follows.

IRIS: This dataset is created by R. A. Fisher, a famous dataset in pattern recognition and classification. There are 150 instances with 4 attributes and with 3 classifications in this dataset. The classification of Iris dataset involves the classification of data like petal length, petal width, sepal length, and sepal width into three classes of species: Iris Versicolor, Iris Sentosa, and Iris Verginica.

This dataset is transformed into the binary class dataset in which one class contains 50 instances that is minority class, while the second class contains 100 instances which present the majority class. For this experiment, total 100 instances are randomly taken from minority and majority instances in the whole dataset for training purpose and 50 random instances for testing. After applying the proposed technique and other standard oversampling techniques on this dataset, the performance of these techniques are evaluated.

WINE: This dataset is created by C. Blake used for examining the classifier performance. It is used for comparison of classifiers with high dimensional settings. It contains a total of 178 instances and 13 attributes which have been classified into three main classes, where each attribute is continuous. In class distribution, the number of instances in the three classes are 59, 71, and 48, respectively. For this experimental setup, this dataset is transformed into the binary class dataset in which one class (minority) has 71 instances and the second class (majority) contains 107 instances. The proposed technique and other oversampling techniques are applied to this dataset then the performance of these techniques are evaluated.

GLASS: This dataset is collected by B. German on the fragments of glass confronted in the forensic work. The study of the classification of glass types drives in the criminological investigation. At the scene of the crime, the glass left can be used as evidence. This dataset is consisted of 9 attributes and 6 classes. There are 163 instances of window glass including 4 subclasses, and 51 instances of Non-window glass with 3 subclasses. Thus, there are a total of 7 classes of a glass. The input attributes of this dataset include Refractive Index, Aluminum, Barium, Calcium, Iron, Magnesium, Potassium, Silicon and Sodium. For this experimental setup, this dataset is transformed into the binary class dataset in which one class (minority) has 51 instances and the other class (majority) contains 163 instances. The proposed technique and other oversampling techniques are applied to this dataset. The description of these datasets is summarized in Table I.

B. Experimental Setup

In this experimental process, the proposed IA-SUWO technique and four other oversampling techniques are applied to these datasets which output the new oversampled datasets. This research is implemented on MATLAB r2015a, on a workstation with 64 bit operating system, 4 GB RAM and 2.6 GHz CPU. Since, every technique generates different number of synthetic minority samples in each dataset, therefore 4-fold stratified cross-validation is used for determining the mean and standard deviation of the oversampling methods performance measures. To reduce the randomness effect of the results, the average of results taken by repeating each experiment three times. The selected datasets are remedied for the imbalanced data problem after being oversampled.

The outcome of the proposed technique and contribution of this research is validated by comparing its oversampling results with the standard technique. It is evaluated by measuring the extent of synthetic samples generated and the reduced imbalanced ratio achieved by both techniques on this dataset. However, to evaluate the performance of each oversampling technique and the correctness of the synthetic samples generated by oversampling techniques, it is necessary to find the classification accuracy of these oversampled datasets. For the classification of these datasets, this research used four classifiers known as; Naïve Bayer, K-Nearest Neighbor (KNN), Logistic regression, and Neural Network. These classifiers are trained and tested on these datasets using 70-30% and 50-50% training and testing ratios, and also classified by 10-Fold cross-validation. These results are measured by the performance evaluations metrics, Precision, F-measure, and ROC; which are compared with standard A-SUWO and other oversampling techniques namely, SMOTE, Borderline-SMOTE, and Safe Level-SMOTE.

C. Performance Measures

Precision, F-measure, and ROC are the performance measures chosen by this research for the evaluation of proposed IA-SUWO. As these are the common measures used by research community for the imbalanced data domain. Precision measures the exactness of the classifier that the number of samples of minority class, which are labeled as positive are actually positive.

TABLE. I. DATASET DESCRIPTION

No.	Dataset	Minority Class	Majority Class	No. of feature	No. of Instances	No. of Minority samples	No. of Majority Samples	Imbalanced Ratio
1	Iris	1	All others	4	150	50	100	1:2.0
2	Wine	-1	All others	13	178	71	107	1:1.83
3	Glass	-1	All others	9	214	51	163	1:3.20

F-measure is calculated by precision and recall, where recall measure the completeness of the classifier. And, the relative importance between precision and recall is adjusted by F-measure. Receiver Operating Characteristics (ROC) is an important factor for evaluating the classification model performance, as it determines that how much the model is capable of distinguishing between the classes.

$$Precision = \frac{TP}{TP+FP} \quad (16)$$

$$Recall = \frac{TP}{TP+FN} \quad (17)$$

$$F_{measure} = \frac{(1+\beta^2)*Recall*Precision}{\beta^2*Recall*Precision} \quad (18)$$

Receiver Operating Characteristics (ROC) is also an important evaluation matrix for determining the classification model performance, as it tells that how much the model is capable of distinguishing between the classes. ROC is acquired by scheming the true positive rate over false positive rate.

$$TPR = \frac{TP}{TP+FN} \; FPR = \frac{FP}{FP+TN} \quad (19)$$

V. Results and Discussion

Performance of proposed technique for the validation of contributed part is evaluated by measuring the extent of oversampling, which means the number of synthetic samples generated and the reduced imbalances ratio achieved by the standard and improved technique on the selected datasets. These results are shown in Table II and discussed as below.

Table II shows the results before and after the oversampling technique applied on all datasets. These results demonstrate that the IA-SUWO generated more synthetic samples than the standard technique using the sparsity factor. The difference in the number of samples generated after oversampling by both techniques indicates that these samples are correctly generated in minority clusters, where the sample are sparse apart and the density was low, as these clusters required more oversampling. The exactness of the generated samples can be evaluated by the classification of the datasets. IA-SUWO generated optimum number of synthetic samples among the minority sub-cluster, as the model did not generate samples randomly or incredibly large number of samples which can cause overfitting which significantly deteriorates the performance of the classifier. The imbalanced ratio given in Table II, after the oversampling, is more reduced by the purposed model than the standard technique. The ideal and best value for imbalanced ratio is 1:1 for both classes, and the value closer to this ratio respectively and the maximum value of imbalanced ratio should not be more than standard technique.

Table III presents the mean results of classification obtained from two classifiers according to 70-30%, 50-50% testing and training ratios, and 10-fold cross-validation for the proposed methods and other oversampling techniques. The best measure is presented in bold. IA-SUWO obtained best results according to at least one measure in all the datasets. Table IV Neural Network classifier shows the best results most of all the measures for both ratios of testing and training; also 10-fold cross validation for proposed IA-SUWO. For Iris and Wine datasets, IA-SUWO presented the best results in all measures by using 50-50% testing and training ratios. For Glass, IA-SUWO achieved best results in all measure using 70-30% testing and training ratios. KNN classifier gave the best results for IA-SUWO using 70-30% testing and training ratio on the Iris dataset in all measures and 10-fold cross validation on the glass dataset.

IA-SUWO presented overall good results on all the datasets. These results declare that the IA-SUWO generated useful synthetic instances by using the sparsity factor which determined the space where the oversampling is needed the most that can be helpful for the learnability of the minority class. In this way, it also enhanced the reliability on the classification results by using any classifier on the datasets which are oversampled by using IA-SUWO.

TABLE. II. Comparative Oversampling Results for Improved and Standard A-SUWO

Dataset	Technique	#. of majority instances	#. of minority instances before oversampling	Imbalanced Ratio	#. of minority instances after oversampling	Imbalanced Ratio
IRIS	IA-SUWO A-SUWO	100 100	50 50	1:2.0 1:2.0	**85** 80	**1:1.17** 1:1.25
WINE	IA-SUWO A-SUWO	107 107	71 71	1:1.50 1:1.50	**96** 90	**1:1.11** 1:1.18
GLASS	IA-SUWO A-SUWO	163 163	51 51	1:3.2 1:3.2	**125** 120	**1:1.30** 1:1.35

TABLE. III. CLASSIFICATION RESULTS FOR OVERSAMPLING TECHNIQUES ON 3 DATASETS USING KNN

Dataset	Testing-Training	Measure	SMOTE	Borderline-SMOTE	Safe-level SMOTE	A-SUWO	IA-SUWO
IRIS	70-30%	Precision	0.898	0.738	0.832	0.911	**0.934**
		F-measure	0.877	0.668	0.824	0.908	**0.923**
		ROC	0.879	0.895	0.898	0.929	**0.941**
	50-50%	Precision	0.891	0.549	0.882	**0.958**	0.923
		F-measure	0.954	0.545	0.877	**0.967**	0.908
		ROC	**0.976**	0.623	0.893	0.974	0.934
	10-Fold	Precision	0.901	0.708	0.921	**0.971**	0.963
		F-measure	0.898	0.706	0.932	**0.969**	0.962
		ROC	0.976	0.740	0.957	0.970	**0.986**
WINE	70-30%	Precision	**0.986**	0.978	0.958	0.977	**0.987**
		F-measure	0.986	0.977	0.954	0.976	0.974
		ROC	0.982	0.980	0.960	0.975	**0.998**
	50-50%	Precision	0.972	0.944	0.930	**0.973**	0.957
		F-measure	**0.972**	0.944	0.930	0.971	0.915
		ROC	0.973	0.945	0.932	0.970	**0.989**
	10-Fold	Precision	0.978	0.974	**0.986**	0.979	0.979
		F-measure	0.977	0.972	0.986	0.978	**0.989**
		ROC	0.980	0.971	**0.989**	0.975	0.978
GLASS	70-30%	Precision	0.971	0.966	0.965	0.975	**0.983**
		F-measure	0.969	0.945	**0.985**	0.984	0.963
		ROC	0.969	0.962	0.985	0.984	**0.988**
	50-50%	Precision	0.966	0.959	0.966	**0.981**	0.937
		F-measure	0.963	0.959	0.963	0.981	**0.984**
		ROC	0.965	**0.991**	0.966	0.971	0.977
	10-Fold	Precision	0.954	0.959	0.959	0.924	**0.986**
		F-measure	0.954	0.959	0.959	0.956	**0.982**
		ROC	0.954	0.971	0.966	0.967	**0.985**

TABLE. IV. CLASSIFICATION RESULTS FOR OVERSAMPLING TECHNIQUES ON 3 DATASETS USING NEURAL NETWORK

Dataset	**Testing-Training**	**Measure**	**SMOTE**	**Borderline-SMOTE**	**Safe-level SMOTE**	**A-SUWO**	**IA-SUWO**
IRIS	70-30%	Precision	0.970	0.901	0.910	0.934	**0.989**
		F-measure	0.965	0.899	0.905	0.923	**0.981**
		ROC	0.985	0.909	0.921	0.992	0.991
	50-50%	Precision	0.971	0.904	0.965	0.971	**0.986**
		F-measure	0.970	0.901	0.966	0.969	**0.979**
		ROC	0.995	0.926	0.971	0.991	**0.998**
	10-Fold	Precision	0.955	0.945	0.947	0.954	**0.956**
		F-measure	**0.955**	0.940	0.947	0.954	0.946
		ROC	0.977	0.963	0.980	**0.996**	**0.996**
WINE	70-30%	Precision	0.977	**0.982**	0.958	0.974	0.977
		F-measure	0.977	0.975	0.954	0.975	**0.978**
		ROC	**0.998**	0.989	0.996	0.991	0.993
	50-50%	Precision	0.973	0.973	0.958	0.942	**0.986**
		F-measure	0.972	0.972	0.958	0.942	**0.986**
		ROC	0.997	0.994	0.992	0.997	**0.998**
	10-Fold	Precision	0.986	0.966	0.986	0.978	**0.987**
		F-measure	0.986	0.965	0.968	0.978	**0.988**
		ROC	**0.998**	0.995	0.995	0.997	0.991
GLASS	70-30%	Precision	0.969	0.969	0.940	0.975	**0.988**
		F-measure	0.969	0.969	0.938	0.974	**0.978**
		ROC	0.972	0.978	0.981	0.987	**0.995**
	50-50%	Precision	0.943	**0.988**	0.924	0.971	0.987
		F-measure	0.935	0.980	0.917	0.973	**0.994**
		ROC	0.956	0.983	0.971	**0.994**	0.990
	10-Fold	Precision	0.943	0.965	0.949	0.977	**0.979**
		F-measure	0.935	0.963	0.969	0.967	0.967
		ROC	0.966	0.966	0.968	0.969	**0.973**

VI. Conclusion

In preprocessing of data by using any resampling technique, oversampling techniques are being preferred mostly by the authors because it does not encounter the risk of losing any potential data. However, by oversampling of data, the synthetic samples over-fitted or can cause over-generalization, but this problem is avoided by IA-SUWO. The standard A-SUWO is lacking the ability to generate more appropriate strength of synthetic samples, while the results were taken after the whole experimental process showed that IA-SUWO outperformed standard technique by using sparsity factor.

Sparsity factor actually measures the density of all clusters, and any of cluster having low density results in high sparsity of the samples, which means that more synthetic samples need to be generated in that cluster. In this way, it increases the ability of any classifier to classify more accurately the minority class instances. The results of oversampling on these datasets clarify the oversampling rates and amount of imbalanced ratio reduced by improved and standard A-SUWO. The results show that the IA-SUWO gave better results in terms of reducing imbalances ratio by using sparsity factor, which appropriately generated more samples than standard A-SUWO; within the minority clusters where the samples are sparse apart and required more oversampling, these generated synthetic instances employed a positive role in the classification process.

Acknowledgment

The authors would like to thank Research Management Center (RMC), Universiti Tun Hussein Onn Malaysia (UTHM) for funding this research, Vote No. H334, and research fund, E15501, UTHM.

References

[1] F. Tsai, W. C. Lin, Y. H. Hu, and G. T. Yao, “Under-sampling class imbalanced datasets by combining clustering analysis and instance selection,” Inf. Sci. (Ny)., vol. 477, pp. 47–54, 2019.

[2] S. Wang and X. Yao, “Multiclass Imbalance Problems : Analysis and Potential Solutions,” IEEE Trans. Syst. Man. Cybern., vol. 42, no. 4, pp. 1119–1130, 2012.

[3] H. Yu, J. Ni, and J. Zhao, “ACOSampling: An ant colony optimization-based undersampling method for classifying imbalanced DNA microarray data,” Neurocomputing, vol. 101, pp. 309–318, 2013.

[4] L. Song, D. Li, X. Zeng, Y. Wu, L. Guo, and Q. Zou, “nDNA-prot: Identification of DNA-binding proteins based on unbalanced classification,” BMC Bioinformatics, vol. 15, no. 1, pp. 1–10, 2014.

[5] M. Zulqarnain, R. Ghazali, S. H. Khaleefah, and A. Rehan, “An Improved the Performance of GRU Model based on Batch Normalization for Sentence Classification,” Int. J. Comput. Sci. Netw. Secur., vol. 19, no. 9, pp. 176–185, 2019.

[6] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, “Machine learning applications in cancer prognosis and prediction,” Comput. Struct. Biotechnol. J., vol. 13, pp. 8–17, 2015.

[7] S. Zafeiriou, C. Zhang, and Z. Zhang, “A survey on face detection in the wild: Past, present and future,” Comput. Vis. Image Underst., vol. 138, pp. 1–24, 2015.

[8] R. Malhotra, “A systematic review of machine learning techniques for software fault prediction,” Appl. Soft Comput. J., vol. 27, pp. 504–518, 2015.

[9] M. Reza, S. Miri, and R. Javidan, “A Hybrid Data Mining Approach for Intrusion Detection on Imbalanced NSL-KDD Dataset,” Int. J. Adv. Comput. Sci. Appl., vol. 7, no. 6, pp. 20–25, 2016.

[10] A. Abu-Srhan, B. Alhammad, S. Al, and R. Al-Sayyed, “Visualization and Analysis in Bank Direct Marketing Prediction,” Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 7, pp. 651–657, 2019.

[11] W. Y. Lin, Y. H. Hu, and C. F. Tsai, “Machine learning in financial crisis prediction: A survey,” IEEE Trans. Syst. Man Cybern. Part C Appl. Rev., vol. 42, no. 4, pp. 421–436, 2012.

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

[13] R. Longadge, S. S. Dongre, and L. Malik, “Class imbalance problem in data mining: review,” Int. J. Comput. Sci. Netw., vol. 2, no. 1, pp. 83–87, 2013.

[14] N. V Chawla, N. Japkowicz, and P. Drive, “Editorial : Special Issue on Learning from Imbalanced Data Sets,” Sigkdd Explor., vol. 6, no. 1, pp. 2000–2004, 2004.

[15] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, “A comparative study of data sampling and cost sensitive learning,” Proc. - IEEE Int. Conf. Data Min. Work. ICDM Work. 2008, pp. 46–52, 2008.

[16] Q. Zou, S. Xie, Z. Lin, M. Wu, and Y. Ju, “Finding the Best Classification Threshold in Imbalanced Classification,” Big Data Res., vol. 5, pp. 2–8, 2016.

[17] G. Douzas, F. Bacao, and F. Last, “Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE,” Inf. Sci. (Ny)., vol. 465, pp. 1–20, 2018.

[18] B. Krawczyk, M. Woźniak, and G. Schaefer, “Cost-sensitive decision tree ensembles for effective imbalanced classification,” Appl. Soft Comput. J., vol. 14, no. PART C, pp. 554–562, 2014.

[19] M. Koziarski, B. Krawczyk, and M. Woźniak, “Radial-Based oversampling for noisy imbalanced data classification,” Neurocomputing, no. 2019, 2019.

[20] M. Galar, A. Fern, E. Barrenechea, and H. Bustince, “A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches,” IEEE Trans. Syst. MAN, Cybern. C Appl. Rev., vol. 42, no. 4, pp. 463–484, 2012.

[21] Q. Kang, X. S. Chen, S. S. Li, and M. C. Zhou, “A Noise-Filtered Under-Sampling Scheme for Imbalanced Classification,” IEEE Trans. Cybern., vol. 47, no. 12, pp. 4263–4274, 2017.

[22] S. Alshomrani, A. Bawakid, S. O. Shim, A. Fernández, and F. Herrera, “A proposal for evolutionary fuzzy systems using feature weighting: Dealing with overlapping in imbalanced datasets,” Knowledge-Based Syst., vol. 73, pp. 1–17, 2015.

[23] I. Nekooeimehr and S. K. Lai-Yuen, “Adaptive semi-unsupervised weighted oversampling (A-SUWO) for imbalanced datasets,” Expert Syst. Appl., vol. 46, pp. 405–416, 2016.

[24] A. Agrawal, H. L. Viktor, and E. Paquet, “SCUT: Multi-Class Imbalanced Data Classification using SMOTE and Cluster-based Undersampling,” 2015 7th Int. Jt. Conf. Knowl. Discov. Knowl. Eng. Knowl. Manag., vol. 01, pp. 226–234, 2015.

[25] A. Majid, S. Ali, M. Iqbal, and N. Kausar, “Prediction of human breast and colon cancers from imbalanced data using nearest neighbor and support vector machines,” Comput. Methods Programs Biomed., vol. 113, no. 3, pp. 792–808, 2014.

[26] 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, 2002.

[27] B. Zadrozny and C. Elkan, “Learning and making decisions when costs and probabilities are both unknown,” Proc. seventh ACM SIGKDD Int. Conf. Knowl. Discov. data Min. - KDD ’01, pp. 204–213, 2001.

[28] W. C. Lin, C. F. Tsai, Y. H. Hu, and J. S. Jhang, “Clustering-based undersampling in class-imbalanced data,” Inf. Sci. (Ny)., vol. 409–410, pp. 17–26, 2017.

[29] G. Douzas and F. Bacao, “Self-Organizing Map Oversampling (SOMO) for imbalanced data set learning,” Expert Syst. Appl., vol. 82, pp. 40–52, 2017.

[30] S. Ite and S. Ite, “Class imbalances versus small disjuncts,” Sigkdd Explor., vol. 6, no. 1, pp. 40–49, 2004.

[31] J. Liu, X. Zhou, D. Li, X. Li, Z. Dong, and S. Wang, Advanced Data Mining and Applications. 2005.

[32] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "Safe-Level-SMOTE : Safe-Level-Synthetic Minority Over-Sampling TEchnique for Handling the Class Imbalanced Problem," Adv. Knowl. Discov. Data Mining, PAKDD 2009, pp. 475–482, 2009.

[33] H. Han, W. Wang, and B. Mao, "Borderline-SMOTE : A New Over-Sampling Method in," Springer-Verlag Berlin Heidelb., pp. 878–879, 2005.

[34] A. S. Nickerson, N. Japkowicz, and E. Milios, "Using Unsupervised Learning to Guide Resampling in Imbalanced Data Sets," Proc. Eighth Int. Work. AI Statitsics, no. 2001, p. 5, 2001.

[35] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," Proc. Int. Jt. Conf. Neural Networks, pp. 1322–1328, 2008.

[36] B. Tang and H. He, "KernelADASYN: Kernel based adaptive synthetic data generation for imbalanced learning," 2015 IEEE Congr. Evol. Comput. CEC 2015 - Proc., pp. 664–671, 2015.

[37] S. Barua, 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, 2014.

[38] S. Babu and N. R. Ananthanarayanan, "EMOTE: Enhanced Minority Oversampling TEchnique," J. Intell. Fuzzy Syst., vol. 33, no. 1, pp. 67–78, 2017.

[39] P. Lim, C. K. Goh, and K. C. Tan, "Evolutionary Cluster-Based Synthetic Oversampling Ensemble (ECO-Ensemble) for Imbalance Learning," IEEE Trans. Cybern., vol. 47, no. 9, pp. 2850–2861, 2017.

[40] Z. Tianlun and Y. Xi, "G-SMOTE: A GMM-BASED Synthetic Minority Oversampling Technique for Imbalanced Learning," arxiv1810.10363v1 , a Prepr., 2018.

[41] T. Zhu, Y. Lin, Y. Liu, W. Zhang, and J. Zhang, "Minority oversampling for imbalanced ordinal regression," Knowledge-Based Syst., vol. 166, pp. 140–155, 2019.

[42] C. Science, D. Dhanalakshmi, A. S. Vijendran, and A. Info, "Adaptive Data Structure Based Oversampling Algorithm for Ordinal Classification," Indones. J. Electr. Eng. Comput. Sci. Vol., vol. 12, no. 3, pp. 1063–1070, 2018.

[43] E. M. Voorhees, "Implementing agglomerative hierarchic clustering algorithms for use in document retrieval," Inf. Process. Manag., vol. 22, no. 6, pp. 465–476, 1986.