Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Rapid Density Calculation for Efficient Undersampling in Large Overlapping Datasets

Sima Mayabadi
University of Tehran
Hamid Saadatfar
University of Birjand
Emrah Arslan
KTO Karatay University
Bulbula Kumeda
Jinka University
Javad Hassannataj Joloudari
javad.hasannataj@gmail.com

University of Birjand

Article

Keywords: Large imbalanced data, Density, Undersampling, Overlapping, Noise

Posted Date: May 29th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9700380/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Rapid Density Calculation for Efficient Undersampling in Large Overlapping Datasets

Sima Mayabadi[1], Hamid Saadatfar[2], Emrah Arslan[3], Bulbula Kumeda[4], Javad Hassannataj Joloudari[2,5,6,*]

[1]Information Technology Engineering, University of Tehran, Tehran, Iran
[2]Department of Computer Engineering, Faculty of Electrical and Computer Engineering, University of Birjand, Birjand, Iran
[3] Department of Computer Engineering, Faculty of Engineering, KTO Karatay University, Konya, Türkiye
[4] Department of Computer Science, Jinka University, Jinka Ethiopia
[5]Department of Computer Engineering, Bab.C., Islamic Azad University, Babol, Iran
[6]Department of Computer Engineering, Technical and Vocational University (TVU), Tehran, Iran

Corresponding Author[*]: javad.hasannataj@gmail.com

Abstract

In the real world, we are faced with many unbalanced large data sets where one class accounts for a large amount of data compared to another class. In most cases, the learning of the minority class is very important. But learning the minority class in an unbalanced data set is not possible with good quality by traditional data mining methods. Data-level solutions that can be divided into three categories, i.e., oversampling, undersampling, and hybrid, are one of the most popular approaches to deal with the lack of balance. In this paper, a fast approach for calculating the density of an instance and, thus, a fast density-based algorithm for balancing large imbalanced data are proposed. The proposed algorithm is based on the undersampling technique and removes samples in dense areas. In addition to eliminating overlapped instances, this algorithm also deletes noise. The experimental results on 22 datasets, which included large ones, show that the proposed high-speed algorithm significantly reduces the time required to balance the data compared to other recent well-known related methods. While using different classifiers, it is also found that the resulting dataset is of high quality in comparison with the output of the other methods.

Keywords: Large imbalanced data, Density, Undersampling, Overlapping, Noise.

1- Introduction

In recent years, the analysis, processing, and learning of imbalanced datasets have been very popular among researchers [1-6]. Imbalanced datasets are divided into majority (negative) and minority (positive) classes. The majority and minority classes have the highest and lowest number of instances in the dataset, respectively [7-9]. The imbalance issue disrupts the learning process; it also makes the learning classifiers a tendency to learn the majority class.

In the real world, most of the available and extracted data from applications are imbalanced. Imbalanced datasets can be seen in malignant cancer [10], wireless networks [11], drug discovery [12], medical diagnosis [13-15], and cell phones [16]. If traditional algorithms are run on imbalanced datasets, the resulting model does not correctly predict the minority class. These models have a high false negative rate in most cases, while minority-class learning is more valuable than majority-class learning. For example, in the diagnosis of rare diseases, if a patient mistakenly predicts a healthy person, the cost of this mistake is much heavier than when a healthy person is diagnosed as sick.

Approaches to dealing with imbalance problems in datasets can be divided into three general categories: data level, algorithm level, and cost sensitive [4, 17]. The first group tries to balance by increasing and decreasing the number of minority and majority class data, respectively [18]. The methods presented in the second category modify the existing traditional algorithms according to the characteristics of the imbalanced dataset to be suitable for their learning [19]. The last category charges a different fee for learning each class according to their value [13, 17, 20]. The first category has attracted more attention from researchers, because they do not need to change the machine learning algorithm and, after preprocessing, all the learning algorithms can be applied to preprocessed datasets. Data level can be implemented using three approaches: oversampling, undersampling, and hybrid.

Oversampling or undersampling methods, especially if they are based on density calculation, although they have better accuracy, are time-consuming, and have a high computational cost. On the other hand, today, due to the advent of the Internet and the use of social networks and smart objects, the volume of data generated every day has greatly increased which leads to the emergence of large imbalanced data. To solve the problem of performing heavy calculations on large data, various methods have been proposed, including the use of parallel and distributed systems, Pruning algorithm calculations, or data reduction [21, 22].

In this paper, a fast density-based method for properly balancing large data is presented. To speed up the algorithm, many of the calculations required to find the density for a sample have been pruned. This reduction of calculations did not damage the accuracy of density calculation and the method still works based on the exact value of density. Furthermore, the proposed method is able to remove and eliminate noise and overlapping data. It also creates a uniform distribution of data in the feature space and solves class distribution problems while trying to maintain the structure of classes maximally. Contributions to this work are as follows:

1- Proposing a fast and accurate approach for data density calculation
2- Proposing a density-based algorithm, using the undersampling technique, which in addition to eliminating overlap, tries to balance and eliminate noise data
3- Comprehensively evaluating the proposed methods on a wide range of datasets that differ in terms of volume as well as imbalanced rate and number of features in comparison with other recent related

The structure of this paper is as follows: Section 2 examines the background of the work done in the field related to the proposed method. Section 3 describes the proposed approach for fast calculation of the density and the proposed algorithm for reaching data balance. Section 4 describes how to implement the proposed methods and the experimental conditions. Section 5 shows the results of experiments performed to evaluate the proposed methods. Finally, Section 6 provides the general conclusion of this paper.

2- Related Work

There has been intense research interest in the application of machine learning techniques to counter the effects of imbalance in large datasets characterized by significant overlapping between the classes. Class imbalance poses a significant challenge in the process of modeling since the approach may end up being biased in the direction of the majority class when there exists overlapping in the dataset. Effective undersampling techniques have therefore become a significant research approach in order to counter the effects of imbalance.

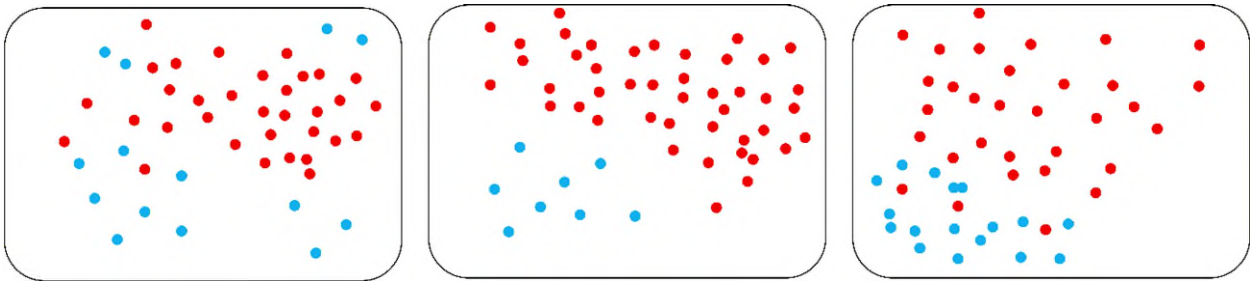
Principally, in the case of efficient undersampling in large overlapping datasets, machine learning approaches to class imbalance issues have drawn a lot of attention. RUS, Tomek Link, and MGRU are among the techniques covered in this section. The proposed approach stands in comparison with the advantages and disadvantages of these methods.

2.1-Characteristics of class distribution

Class distribution makes up one of the key factors that influence the performance of the classifier and machine learning process, as discussed in numerous research works, such as [6, 23]. As shown in Figure 1, it consists of three characteristics.

1- **Small Disjuncts:** This situation arises if the data points are scattered into separate areas of the space, and there is little overlap between the respective areas. Classic learning algorithms often tend to incorrectly classify the small disjuncts as noise. This is particularly harmful in a system that depends upon a few key sub-classes.
2- **Small Sample Size:** When some classes have only a few examples, the learning process is hampered. Scarce data can reduce the model's generalization capability.
3- **Class Overlapping:** In some cases, instances from different classes share similar values of features; it is hard for the classifier to accurately tell them apart.

The works under this section emphasize the importance of data distribution characteristics (overlap, noise, boundary instances) and propose methods that aim to preserve or clean the class structure rather than just balancing class counts. Class overlap has been identified as one of the factors that is most detrimental to the process of learning, compared to imbalance learning, as observed in various research [24-28].



a) Small disjuncts **b) Small sample size** **c) Class overlapping**

Fig. 1. Characteristics of Class Distribution

This paper [29] addresses multi-class imbalance in the presence of label noise, small disjuncts, and overlapping class distributions. The method identifies “safe regions” for oversampling and performs

simultaneous cleaning to remove noisy and borderline instances, thereby preserving the intrinsic class structure while balancing classes. The authors show robustness across multiple multi-class benchmark datasets.

EvoSampling is a hybrid method that uses evolutionary programming with a "granular ball" approach to generate diverse, high-quality minority instances, followed by a multi-granularity undersampling to remove noise and overlapping majority samples [30]. Because it aims for high data quality, not just balance, it is well-suited for datasets with complex class distributions, overlap, and noise. Experimental results across 20 imbalanced datasets show consistent improvements over standard sampling methods.

SMOTE was extensively used in the literature for handling class imbalance by injecting synthetic samples of the minority class through a linear combination of the minority class samples along with their K-nearest neighbors. Though successful in effectively boosting the minority class instances, the approach cannot ensure the generated samples represent the class distribution in the original problem effectively, resulting in potential overlap in the regions for the majority class. Recent work in [31] introduced worthwhile theoretical insight for the popular oversampling approach of SMOTE by analyzing the probability distribution for the generated samples in comparison to the true class-conditional distributions. This recent work effectively sheds insight into the inadequacies in the use of the popular class oversampling approach in maintaining class structural properties for the minority class.

The iHHO SMOTE method deals explicitly with outliers and noise before oversampling. It applies a clustering-based outlier detection (e.g., DBSCAN) to remove noisy minority samples, then oversamples the cleaned minority set [32]. This ensures that synthetic samples are generated from a "clean" minority distribution, reducing the risk of overlap or distorted class conditional distributions. The reported results show very strong classification performance (e.g., high AUC and G mean).

The research work in [33] presented Optimized Borderline Minority Instance (OBMI), a technique aimed at easing the class imbalance problem by focusing on borderline minority instances, which are more critical for classifier performance. This method employs a two-stage Tomek link procedure: first, Tomek links between nearest-neighbor instances of opposite classes are identified to select minority instances for oversampling, and new synthetic samples are generated via linear interpolation between these instances and their minority neighbors; second, the removal of overlapping or noisy instances from both classes by again applying Tomek links reduces class overlap introduced during oversampling. Conducted tests in several benchmark datasets depict that the OBM technique is effective in improving minority class representation with noise control.

2.2- Imbalanced dataset

These studies in this section focus on dealing with class imbalance (minority vs majority) using oversampling, undersampling, or hybrid sampling, typically on moderately sized / standard datasets without necessarily focusing on large-scale or high-volume data.

Several approaches have been proposed for dealing with imbalanced data, which can broadly be classified into three categories, namely data-level, algorithm-level, and cost-sensitive learning, as discussed in [4, 17]. This paper focuses on data-level solutions, which include under-sampling and over-sampling, and introduces the concept of their combination, which has recently come into focus.

Data level approaches can be divided into two main techniques of undersampling and oversampling. Other categories of research in this area use a combination of these two techniques to achieve balance, known as hybrids. In the undersampling technique, the majority of class samples are deleted until the balance is reached. The idea is to delete the data samples that have a bad or negligible effect on the learning process. Oversampling increases samples in the minority class by generating new samples or copying the existing ones. If this technique is used for balancing a highly imbalanced dataset, the overfitting chance will increase, as well as the data volume, which makes the dataset difficult to manage, control, and store [4].

2.3- Undersampling Techniques

In undersampling, the majority class instances are removed until the majority and minority classes are equal. The basic assumption underlying the undersampling technique is that the removed data are less significant to the learning process. The simplest undersampling technique is the Random Undersampling method, wherein the majority class data are removed without any discrimination [7, 18]. The major drawback of the simple undersampling technique is that it removes significant data points.

Other approaches like Tomek Links (TL) [34] and Edited Nearest Neighbors (ENN) [35] focus on lessening the overlap, besides having an equal number of instances for each class. Nevertheless, the use of TL has some issues regarding the capability to lessen the overlap, as discussed previously. In an attempt to remedy the above-mentioned problems, an innovative strategy called Multi-Granularity Relabeled Undersampling (MGRU) was presented. The proposed strategy tries to overcome the TL method's problems concerning balance and overlap.

Tomek Link works by selecting and removing cases from the majority class that are closest to those of the minority class. The authors in [36] assessed the efficacy of the method in clarifying class boundaries through the removal of instances from the majority class that are nearest to instances in the minority class from models of large overlapping datasets. They noted that the approach does not directly address the issue of class imbalance, but the data they presented demonstrated significantly improved model accuracy. Sharpening decision boundaries can be achieved by excluding the majority class instances, but doing so may also result in the omission of some crucial information that is necessary for appropriate model training.

SMOTE is combined with Tomek Link undersampling for imbalanced fault detection data in electrical machinery [37]. The authors claim that the hybrid technique produces the best classification results, and this is particularly notable when it comes to identifying the minority (fault) class, which clearly shows the efficiency of hybrid resampling.

This [38] study compares several sampling strategies, such as Tomek Links undersampling, Borderline SMOTE oversampling, and hybrid SMOTE+clustering+undersampling on a highly imbalanced synthetic fraud dataset, PaySim. Hybrid sampling techniques result in better performance on F1 score and ROC-AUC compared to pure oversampling or undersampling; this again stresses the benefit of method combination within imbalanced classification tasks.

The result of recent developments in efficient undersampling in large overlapping datasets is the Multi-Gradient Reweighting Unit (MGRU), a method that increases the model's resilience by giving the training samples varying weights based on their gradients throughout the learning process. In [39], the authors consider the Tomek Link algorithm to have disadvantages in resolving data overlap, so they have proposed another algorithm named the multi-granularity relabeled undersampling algorithm (MGRU). The authors

used the method to prove that this technology could strengthen the learning of informative samples, here meanwhile detracting the influence from corrupted data. The findings from their study confirmed that the MGRU can remarkably enhance the analytical accuracy when compared with the traditional way, especially in the case of multi-source data. Furthermore, another aspect emphasized by the study is the difficulty of adjusting hyperparameters when it comes to MGRU. The authors propose that technical methods could ease such complexity, especially when working on large-scale projects.

Random Under-Sampling (RUS) offers a way to handle class imbalance by reducing the size of the majority class. Research conducted by [40] supports this approach; combining RUS with ensemble learning boosts the recall of the minority class significantly. Nevertheless, the overall classification performance remains largely unaffected. Even so, it doesn't make a significant impact on how well classifications are done. In the end, the performance of classification stays almost the same. On the flip side, RUS has its drawbacks; it can remove important data points, leading to data loss. This might negatively impact improvements in making models more generalizable. The study shows that RUS can simplify datasets, but it also raises the question of whether there is enough data to make accurate predictions.

Further researches were conducted to study advanced undersampling algorithms, and some of the algorithms that were investigated include DBSCAN and graph-based undersampling algorithms, but such algorithms are often associated with high processing costs [41, 42]. Some other researchers [24, 43-45] have employed the KNN algorithm to perform the undersampling process. In [45], four algorithms based on the nearest neighbors concept were presented. Among them, the NB-Rec algorithm complements the previous three algorithms and removes overlap. In [46], the KNN algorithm has been used to solve the imbalance in software defect prediction data. The N-US algorithm, in addition to enhancing the learning accuracy of the minority class, aims to avoid excessive deletion of data and information of the majority class. The K-means algorithm has also been used in some papers [47, 48] to achieve balance. The kmUnder algorithm [48] considers the number of clusters in the majority class as much as the number of minority class samples. Cluster centers are presented as final samples.

2.4-Oversampling Techniques

Unlike the undersampling approach, oversampling methods seek to increase the representation of the minority class. In the Random Oversampling method, new instances are created for the minority class by copying the existing instances, which can increase the risk of overfitting. The Synthetic Minority Oversampling Technique (SMOTE) [49], which creates new instances inside the feature space, increasing the number of the minority class, has gained significant attention as an innovative solution for the problem.

This [50] paper provides a broad empirical comparison of multiple oversampling techniques, including SMOTE, SVM SMOTE, Borderline SMOTE, K-means SMOTE, and ADASYN across many datasets. It assesses various oversampling techniques and their impact on classifier performance, and it is a good source to consider when researching the advantages/disadvantages of typical oversampling techniques.

In oversampling techniques, the size of the dataset increases dramatically. Random oversampling (ROS) increases the size of a minority class by randomly copying minority class samples. The synthetic minority oversampling technique (SMOTE) [49], which uses the nearest neighbor approach, is the most well-known and widely used method in this regard. Two developments of this method that have focused on overlapping are Borderline-SMOTE [51] and SMOTE-IPF [52]. In [53], an SMOTE-based oversampling method is presented, which tries to improve its performance and stability by reducing the randomness of the SMOTE

method in creating new data. Another improvement of the SMOTE algorithm is presented in [54]. Intending to improve the SMOTE algorithm, the authors have proposed the NaNSMOTE method that uses a random difference between the original selected sample and one of its natural neighbors to generate new data. In [55], an algorithm using SMOTE and K-means to increase the volume of minority classes is proposed. The proposed algorithm in the paper [56] also uses the knn as a common method to generate synthetic data.

2.5-Large Imbalanced Data

The focus of some of these research works is scalability, data quality, and computational efficiency, which in turn relate to handling large or complex imbalanced datasets. EvoSampling, as described above in [30] while focusing on data quality, also emphasizes efficient hybrid sampling via evolutionary programming and multi-granularity undersampling, making it suitable for larger or more complex datasets where standard oversampling/undersampling may fail.

Most of the research in this area has used parallel programming and employed high-performance computing systems to process large imbalanced datasets. The Spark [57] and map-reduce parallel programming environments have attracted the most attention. Researchers in [58] have implemented several techniques, especially oversampling and cost-sensitive ones, using MapReduce and evaluating their effectiveness. In [59], a development of SMOTE using MapReduce was presented and evaluated by using a random forest classifier. In [60], parallel algorithms based on the map-reduce model are presented for undersampling and oversampling in large imbalanced data. However, high-performance systems are very expensive and not always available. Compared to these methods, the algorithm proposed in this research is not based on the use of a distributed or parallel system. In fact, an attempt has been made to provide a fast method for calculating the density of samples and removing data based on it by reducing additional calculations (while maintaining accuracy) so that large imbalanced data can be handled with the help of a normal computing machine.

The work in [61] used a method that iteratively combines a safe level variant of SMOTE (generating synthetic minority samples in “safe” regions) with SVM-based undersampling to remove distant majority points, thus balancing classes while cleaning boundary noise. This hybrid, adaptive method seeks to preserve meaningful data structure and is more scalable than naive sampling.

In this [62] study on extremely imbalanced credit card transaction data, the authors apply a hybrid method combining majority class undersampling (via Tomek Links), clustering (BIRCH) to identify minority sub-regions, and then Borderline SMOTE to oversample minority instances. The approach strikes a balance between computational viability and sampling quality in order to handle the large volume, skewed datasets common in real-world fraud detection.

In 2025, the authors in [63] took a deep dive into a meta-analysis focusing on different sampling methods like RUS, SMOTE, and Tomek Link. They looked into how various approaches manage class imbalances. Random Under Sampling (RUS) may prove useful, but perhaps other alternatives could preserve more vital information. Perhaps other alternatives could preserve the vital information needed for the success of the process in the field of classification. Even though RUS offers some strengths on its own merits, perhaps other alternatives might help in achieving improved results through the preservation of vital information. Other methods might preserve the important information needed for better performance. They proposed a complex sampling strategy to deal with the problem of class imbalance in order to preserve the vital information of the dominant class.

The work in [64] proposed an approach addressing the problem of record linking in massive class-imbalanced datasets by combining both oversampling and undersampling methods. This proposed approach works effectively in addressing the challenge of class imbalance in massive datasets by emphasizing the increase of the minority class while downplaying the majority class. Sensitivity analysis was performed with the approach by varying the splitting percentage for the train and test datasets. Despite its strengths in addressing the issue of class imbalance in massive datasets effectively, it lacks explicit methods addressing the local class boundary in the approach. Additionally, it may incur higher computational costs for very massive datasets. Also, the approach may incur class imbalance during the process of synthesizing instances.

The balancing methods proposed in [65, 66] have used deep learning and parallel programming for large data. Researchers in [45] proposed a new undersampling method to remove overlapping data samples. They claimed the proposed method applies to large imbalanced data. In [67], a convolutional rebalancing network is presented for the classification of large imbalanced agricultural data. This network consists of a combination of a convolutional rebalancing module, an image augmentation module, and a feature fusion module to classify images related to rice pests. In the article [68], an oversampling technique called Fast-SMOTE is presented for large data. This technique is based on the SMOTE method, but like many developments of SMOTE, it does not use the k-nearest neighbor method. Although this algorithm has a high speed in balancing, because it works based on the oversampling technique, it is not suitable for high-volume data, and it increases the dataset processing time for learning.

The evaluation of the existing techniques for undersampling identifies the strengths and weaknesses of these techniques in countering the imbalance problem in large and overlapping datasets. The Multi-Gradient Reweighting Unit (MGRU) technique has shown versatility in handling the imbalance problem but lacks efficiency in hyperparameter settings. The Tomek Link (TL) technique helps to distinctly define the boundaries for classification by efficiently removing the overlapping majority instances, but suffers from the problem of losing pertinent information. Random Under-Sampling (RUS) technique helps to efficiently decrease the overlapping dataset, but faces the risk of losing the vital information present in the dataset. Even though the methods have made their contributions in effectively handling the problem of imbalance in the dataset, the proposed algorithm makes a breakthrough in providing a more complex framework for information retention in the dataset, supported by the efficiency of computation. Contrary to the existing algorithms for imbalance problems in the dataset, the new algorithm neither incorporates parallel processing concepts nor makes multiple iterations to estimate the density of the sample, but efficiently identifies the unnecessary majority instances in the dataset. The technique takes advantage of the density of the sample in the dataset to produce a higher-quality balanced dataset without wasting more computation in producing the same. Additionally, the algorithm efficiently reduces the amount of majority class information in the dataset, which has a significant effect on the computation of the dataset in the context of a learning platform. Finally, Table 1 provides a comprehensive comparison of the discussed methods, summarizing their core mechanisms, overlap handling strategies, strengths, and limitations.

Table 1. Comparison of related sampling methods in terms of mechanism, overlap handling, strengths, and limitations.

Study	Method / Variant	Method Type	Core Mechanism / Idea	Overlap Handling	Strengths	Limitations

Pereira et al. (2020) [36]	**Tomek Link (TL)**	Undersampling	Removes the majority of instances closest to the minority	Moderate – clarifies boundary but may remove informative data	Simple; enhances clarity of the decision boundary	Cannot correct the imbalance by itself; might eliminate useful samples
Dai, J. et. al. (2022) [31]	**MGRU**	Gradient-based Undersampling	Reweights samples using gradient magnitude & relabeling	Strong – suppresses corrupted/overlapping samples	Robust to multi-source/complex data.	Difficult hyperparameter tuning, computationally heavy.
Khan et al. (2024) [4]	**Random Undersampling (RUS) + Ensemble**	Undersampling	Random removal of the majority + ensemble boosting	Weak – not designed to handle overlap	Fast; improves minority recall; reduces dataset size	High information loss risk; overall accuracy unchanged
Various [24, 43–46]	**KNN-based Undersampling**	Undersampling	Uses nearest neighbors to remove the majority while retaining structure	Moderate to strong, depending on variant	Good boundary cleanup; preserves structure	Sensitive to K selection; expensive on large datasets
Various [41, 42]	**DBSCAN & Graph-based Undersampling**	Clustering/Graph Undersampling	Density-based cluster pruning of majority samples	Strong – identifies noisy clusters	Effective in overlap removal	Time-consuming; not scalable
Koziarski et al. (2020) [29]	**Cleaned Oversampling (Koziarski et al.)**	Cleaned Oversampling	Identifies "safe regions" + removes noisy/borderline instances while oversampling	Strong explicit removal of noise & borderline cases	Preserves class structure; robust to noise; suitable for multi-class	Higher computational overhead; complexity increases with noise
Pei, Wenbin, et al. (2024) [30]	**EvoSampling**	Hybrid Evolutionary Sampling	Granular-ball evolutionary generation + multi-granularity undersampling	Strong – removes overlap via multi-stage cleaning	High-quality synthetic samples; strong performance across datasets	Complex pipeline; evolutionary search can be expensive
Raslan, K. SH.	**iHHO-SMOTe**	Hybrid (Cleaning +	Outlier removal	Strong – cleans minority noise	High AUC & G-mean;	Clustering costs high for

Et. al. (2025) [32]		Oversampling)	(DBSCAN) → oversampling cleaned minority subset	before generating samples	reduces SMOTE noise	large datasets; sensitive to DBSCAN parameters
Elreedy et al. (2024) [31]	**SMOTE Distribution Analysis (Elreedy et al.)**	Theoretical Analysis	Analyzes SMOTE distribution vs true distribution	Weak – identifies but does not solve the overlap	Deep insight into SMOTE's structural distortion	Diagnostic only; no practical sampling algorithm
Chawla, Nitesh V., et al. (2002) [49]	**SMOTE**	Oversampling	Synthetic samples generated via K-NN interpolation	Weak – may increase overlap	Simple, widely used	Cannot maintain the original class distribution
Han, H. et. al. (2005) [51]	**Borderline-SMOTE**	Oversampling	Generates samples around the borderline minority	Moderate	Focuses on a difficult region	Still susceptible to noise
Sáez, José A., et al. (2011) [42]	**SMOTE-IPF**	Hybrid Oversampling	SMOTE + iterative filtering	Strong	Reduces noisy samples	Multi-stage & slower
Li, Junnan, et al. (2021) [54]	**NaNSMOTE**	Oversampling	Uses natural neighbors to guide synthetic generation	Moderate	Less random, more stable	Still increases the dataset size
Douzas, G. et. al. (2017) [55]	**SMOTE + K-means**	Hybrid Oversampling	Generate samples per cluster	Moderate	Better cluster representatio n	Costly clustering
Mujahi d et al. (2024) [50]	**SMOTE Variants Benchmarki ng**	Oversampling Benchmarking	Compares SMOTE variants (SVM-SMOTE, K-means SMOTE, ADASYN, etc.)	Variable	Comprehensi ve evaluation	Shows tradeoffs but offers no new method

3-The Proposed method

The use of density-based methods helps well identify the depressions of the classes while standardizing the density and maintaining the shape of the classes. Given this point and in order to achieve higher quality in learning, we select a density-based method for balancing and cleaning. However density-based methods are time-consuming, making them very difficult to use for large data.

First, in this paper, an effective approach is proposed for calculating the data density with an acceptable speed, and, then, using this method, an algorithm is proposed to deal with a large imbalanced data problem. In order to calculate the density of a sample, the number of its neighbors in a certain neighborhood radius should be determined. For example, in Figure 2, assuming the neighborhood radius is *r*, the number of neighbors around sample *A* that belong to the circle class is equal to 4.

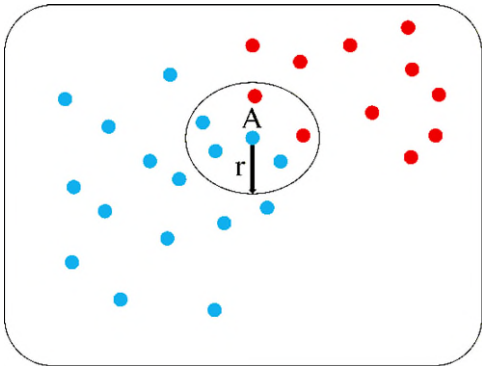


Fig. 2. The density of sample A

3.1-The Fast density calculation method

In this paper, the Euclidean distance (Equation (1)) is used to calculate the distance between the samples. To calculate the density of a sample, its distance to all other samples must be calculated which is very costly. To solve this problem, we propose the fast density calculation method (FDCM). In this method, it first selects a sample (d_{rand}) randomly and calculates its distances to all the other samples in the dataset. Then, based on the neighborhood radius (r) and the calculated distances, the data are divided into t subsets with *r(1)* to *r(t)* names. The value of t is calculated from Equation (2). Table 2 defines these subsets in more detail.

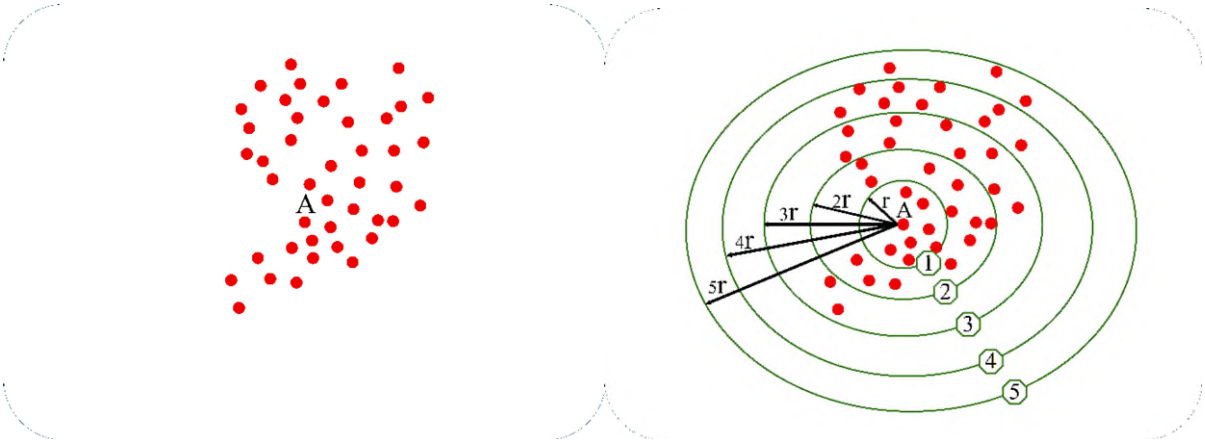
Table2 . The definitions of subsets r(1) to r(t)

Name	Definition
r(1)	samples that their distance from sample d_{rand} is between 0 and r.
r(2)	samples that their distance from sample d_{rand} is between r and 2*r.
r(3)	samples that their distance from sample d_{rand} is between 2*r and 3*r.
…	
r(t)	samples that their distance from sample d_{rand} is between t*r and (t+1)*r.

$$D_{\text{euclidean}}(\mathrm{x},\mathrm{y}) = \sqrt{\sum_{i=1}^{p}(x_i - y_i)^2} \quad \textbf{(1)}$$

$$t = \lfloor (\, max\ (distance\ d_rand\ \ to\ all\ data\ of\ dataset)/r\ ) \rfloor \quad \textbf{(2)}$$

Figure 3 shows the result of performing these steps of the proposed algorithm. Part A shows the initial dataset. Sample *A* is randomly selected. Distance *A* to other samples is calculated. Then, the value of *t* is calculated (*t* in this figure is equal to 5). Afterwards, it forms subsets *r(1)* to *r(5)*. In Figure 3, part b shows the samples in each subset.



(A) **(B)**

Fig 3. Dividing data into subsets in the FDCM approach

According to Figure 4, samples *A* and *B* are the border samples from both sides in subset *r(h)*. To determine the density and thus the label of these two samples, in addition to the *r(h)* subset, the samples of *r(h-1)* and *r(h+1)* subsets are required. Therefore, to determine the data density of the samples that belongs to set *r(h)*, it is just necessary to consider the samples in three subsets: *r(h)*, *r(h-1),* and *r(h+1)*.

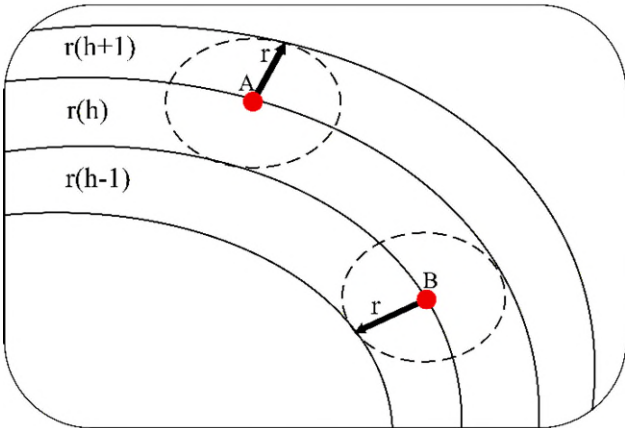


Fig 4. Subsets required to determine the density and label of subset r(h) samples

Proof: *r(h)* is defined as a set of data samples whose distance from the sample d_{rand} (d_{rand} is modeled as (x_{rand}, y_{rand}) in 2D space) is between $r \times h$ and $r \times (h-1)$. Equation 3 illustrates this definition:

$$r(h) = \left\{(x,y) | r \times (h-1) \leq \sqrt{(x - x_{rand})^2 + (y - y_{rand})^2} \leq r \times h\right\} \quad \textbf{(3)}$$

As you can see in Figure 4 and based on Equation 3, two samples $A(x_a, y_a)$ and $B(x_b, y_b)$ from *r(h)* have the maximum and minimum distance to sample d_{rand} in this subset, respectively. Equation 4 specifies these distances:

$$\begin{aligned} dis(A, d_{rand}) &= \sqrt{(x_a - x_{rand})^2 + (y_a - y_{rand})^2} = r \times h \\ dis(B, d_{rand}) &= \sqrt{(x_b - x_{rand})^2 + (y_b - y_{rand})^2} = r \times (h-1) \end{aligned} \quad \textbf{(4)}$$

On the other hand, to calculate the density for *A* and *B*, data samples that are less than r away from them must be available. Equation 5 shows the datasets (S_a and S_b) needed for calculating the density of these two samples:

$$\begin{aligned} S_a &= \left\{(x,y) | \sqrt{(x - x_a)^2 + (y - y_a)^2} \leq r\right\} \\ S_b &= \left\{(x,y) | \sqrt{(x - x_b)^2 + (y - y_b)^2} \leq r\right\} \end{aligned} \quad \textbf{(5)}$$

Based on the definitions of sets S_a and S_b (presented in Equation 5) and Equation 4, The upper and lower limits for the distances between the data of these two sets and sample d_{rand} can be written by Equation 6, respectively:

$$\begin{aligned} r \times (h-1) &\leq \sqrt{(x - x_{rand})^2 + (y - y_{rand})^2} \leq r \times (h+1) \\ r \times (h-2) &\leq \sqrt{(x - x_{rand})^2 + (y - y_{rand})^2} \leq r \times h) \end{aligned} \quad \textbf{(6)}$$

Based on the definition of set *r(h)* (Equation 3), sets *r(h-1)* and *r(h+1)* can be defined similarly. According to these definitions, Equation 7 can be written about the union of these sets:

$$r(h) \cup r(h+1) = \{(x.y) | r*(h-1) \leq \sqrt{(x-x_{rand})^2 + (y-y_{rand})^2} \leq r*(h+1)\}$$
$$r(h) \cup r(h-1) = \{(x.y) | r*(h-2) \leq \sqrt{(x-x_{rand})^2 + (y-y_{rand})^2} \leq r*h\} \quad \textbf{(7)}$$

As a result, the union of all three sets will be equal to:

$$r(h) \cup r(h-1) \cup r(h+1) = \{(x.y) | r*(h-2) \leq (x-x_{rand}) + (y-y_{rand}) \leq r*(h+1)\} \quad \textbf{(8)}$$

According to Equations (6) and (8), three subsets *r(h)*, *r(h-1),* and *r(h + 1)* are just needed to calculate the density for data samples in subset *r(h)*. As a result, based on this division of data into smaller subsets, it is no longer necessary to calculate the distance to all other samples to obtain the density for each data point. This idea can lead to a significant reduction in computational overhead especially in the case of large datasets.

3.2-DBUSB algorithm

Density-based algorithms usually need two input parameters, neighborhood radius (*r*) and the minimum number of neighbors within that radius (*minpoint*). Density-based algorithms usually divide data samples into three categories based on these two parameters: core, border, and noise samples.

Sample *A* belongs to the core category if the number of its neighbors within the neighborhood radius (*ngh(A,r)*) is greater than or equal to *minpoint* value: $|ngh\ (A,r)| >= minpoint$. Sample *B* belongs to the border category if the number of neighbors in the neighborhood radius (*ngh(B,r)*) is less than the *minpoint* value and, in the neighborhood of sample *B,* a sample such as *C* exists which belongs to the core category: $|ngh(B,r)| < minpoint$ and $\exists\ C\ \in ngh(B,r) | \ | ngh\ (C,r)\ | >= minpoint$. Sample *D* is noise if it does belong to neither border, nor core categories.

The density-based undersampling for large imbalanced data algorithm (DBUSB) is proposed to achieve balance at high speed while maintaining the classes' structure maximally. In addition, this algorithm eliminates overlap and deletes noise in minority and majority classes. This algorithm uses the proposed FDCM approach to speed up preprocessing.

First, the DBUSB algorithm uses Algorithm 1 in order to detect the high-dense majority class samples and remove the number of samples needed to balance from these areas. In Algorithm 1, the proposed method FDCM is used to calculate the density. Then, using algorithm 2, it removes overlap and deletes noise in the minority class.

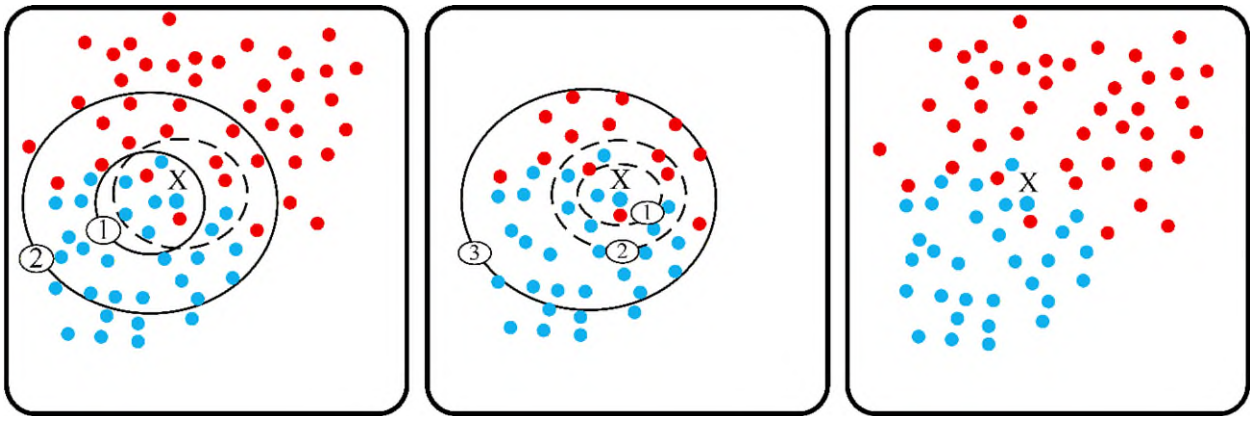
The samples are labeled as noise, border, core, and indelible core by algorithm 1. This algorithm randomly selects a sample (like *x*) and calculates its distance to the samples of the majority class (*line 2*). Subsets $D_{sub}^{(i)}$ includes a number of data from the majority class that we want to calculate their density in this round and it calculate using Equation 9 and $D_{Threesub}^{(i)}$ includes all the data from the necessary majority class to accurately calculate the density of the data of subset $D_{sub}$are made according to FDCM method, it is also calculate by Equation 10.

$$D_{sub}^{(i)} = \{y\ \in D \mid r.(i-1) \leq dist(y, d_{rand}) \leq r.i\} \quad \textbf{(9)}$$

$$D_{Threesub}^{(i)} = \{ z \in D \mid r.(i-2) < dist(z, d_{rand}) \leq r.(i+1) \} \quad \textbf{(10)}$$

For each sample of D_{sub} that is not yet labeled, it calculates the number of neighbors around it. If the number of neighbors is less than *min_majoirty*, it is identified as a border; if all of its neighbors are border, the sample and its neighbors are labeled as noise (line 10). If the number of neighbors is greater than *min_majoirty*, algorithm 1 examines the label of neighbors and places the samples that are labeled as core in the *neighbors_removable* subset (lines 11 to 13). Then, it randomly deletes the samples of the *neighbors_removable* subset until reaching balance (line 14); Subsequently, the central sample and its surviving neighbors are designated as 'indelible core', ensuring their preservation in the final dataset.

According to the values of *r_majority* and *min_majority* parameters (which are described in detail in Section 4.2), algorithm 2 labels minority class samples just as noise or core, because this algorithm only wants to remove the data that is identified as noise from the minority class. Algorithm 2 detects noise samples based on density again to have higher accuracy. In order to have a higher speed, it also uses the proposed algorithm FDCM. When Algorithm 1 is completely performed, algorithm 2 randomly selects a sample (x) and calculates its distance to the samples of both classes (*distance2* Matrix in line 4). The distance of this sample to the minority class data is stored in a matrix named *distance1*. Algorithm 2 creates subsets $D_{sub}^{(i)}$ and $D_{Threesub}^{(i)}$ based on the proposed FDCM method using the data of these two matrices, as in Algorithm 1, so that the density of samples can be calculated at a higher speed (lines 7 and 8). For each sample belonging to the $D_{sub}^{(i)}$ subset that hasn't visited yet, algorithm 2 calculates its distance to all the data of $D_{Threesub}^{(i)}$ and according to the conditions written in line 12 of the algorithm, the set of *neighbors(X)* (which is equal to the blue samples in circle 4 of figure 5) and the set *D_overlap* (which is equal to the red samples in circle 3 of figure 5) are created. The radius of the circle is 3 equal to r_minority /2 because we don't want to lose too much of the majority class data as overlap and do want to keep the shape of the majority class with maximally. In Figure 5, it is assumed that *min_minority* is equal to 3. Because in this example, the number of neighbors for *X* data is more than 3, *X* is labeled as core, and *D_overlap* data is removed. But if the number of *X*'s neighbors was less than 3, it would get a noise label and would be removed at the end, and we would have nothing to do with the *D_overlap* data.



(1) **(2)** **(3)**

Fig 5. Red squares represent the majority class and blue circles represent the minority class

Algorithm 1: undersampling with FDCM approach

Input: D_majority, min_majority, r_ majority
Output: DR_ majority

1. select x $\in$ D_ majority randomly
2. calculate distance x to D_ majority
3. calculate t using the formula (2)
4. for i=1:t
5. D_sub^i = {y | distance(y) <= r_ majority *i and distance(y) >= r_ majority *(i-1))}
6. D_threesubs^i= {z | distance(z) <= r_ majority *(i+1) and distance(z) >= r_ majority *(i-2))}
7. for each X $\in$ D_sub^i that X doesn't have a label
8. calculate distance X to D_threesubs^i
9. neighbors (X) = y $\in$ D_threesubs^i that distance X to y <= r_ majority
10. if (num (neighbors (X)) < min_majority)
    a. if (all of the neighbors(X) == border)
       X and neighbors(X) are noise
    b. else
       X is border
11. if (num (neighbors (X)>= min_majority)
    X is core
12. neighbors_removable = []
13. for each z $\in$ neighbors (X) that isn't indelible core
    if (z is core)
    a. neighbors _removable = neighbors _removable U z
    if (z doesn't have labeled)
    b. neighbors (z) = {y $\in$ D_threesubs^i that distance z to y <= r_ majority}
    c. if (num(neighbors(z)) < min_ majority)
       i. z is border
    d. if (num(neighbors(z)) >= min_ majority)
       i. z is core
       ii. neighbors _removable = neighbors _removable U z
    num_remove = (num(neighbors(X)) - (min_ majority)
14. for i=1 to num_remove
    a. Select a data from neighbors _removable randomly
    b. neighbors _removable = neighbors _removable - selected data
    c. neighbors(X)= neighbors(X) - selected data
    d. D_ majority = D_ majority - selected data
15. X and neighbors(X) are indelible core
16. DR _ majority = D_ majority –{x $\in$ D_ majority that is noise }

Algorithm 2: Remove Overlap with FDCM approach

Input: D_minority, DR_ majority, min_ minority, r_ minority
Output: FDR_ majority, FD_ minority

1. select x ∈ D_ minority randomly
2. calculate distance from x to D_ minority (*distance1*)
3. calculate t using the formula (2)
4. calculate distance from x to {DR_ majority ⋃ D_ minority} (*distance2*)
5. D_noise = []
6. for i=1:t
7. D_sub^i = {y | distance1(y) <= r_ minority *i and distance1(y) >= r_ minority*(i-1)}
8. $D_threesubs^i$ = {z | distance2(z) <= r_ minority *(i+1) and distance2(z) >= r_ minority *(i-2)}
9. for each X ∈ D_sub^i that X has not visited label
10. D_overlapp = []
11. calculate distance X to $D_threesubs^i$
12. for y ∈ $D_threesubs^i$
    a. if (distance X to $D_threesubs^i$ <= r_ minority and y ∈ D_ minority)
       neighbors (X) = neighbors (X) ⋃ y
    b. if (distance X to $D_threesubs^i$ <= r_ minority /2 and y ∈ DR_ majority)
       D_overlap = D_overlap ⋃ y
13. if (num(neighbors(X)) < min_ minority)
    a. X is noise
    b. D_noise = D_noise U X
14. else
    a. X is core
    b. DR_ majority = DR_ majority - { D_overlap }
15. FDR_ majority = DR_ majority
16. FD_ minority = FD_ minority - {D_noise}

3.3- Complexity Analysis

The computational efficiency of the proposed DBUSB algorithm is derived from its departure from standard density-based calculations, which traditionally necessitate a full pairwise distance matrix with a complexity of $O(N^2)$. Instead of this exhaustive search, the Fast Density Calculation Method (FDCM) utilizes a partitioning strategy that maps the dataset into t discrete concentric subsets relative to a random pivot sample. This geometric decomposition allows the algorithm to localize the density estimation process, ensuring that distance calculations for any given sample are strictly confined to its own subset and its immediate neighboring rings $\big(r(h-1)\big), r(h)$, and $\big(r(h+1)\big)$, rather than the entire dataset.

Mathematically, this partitioning reduces the number of required distance computations by a factor of t , s the search space for neighbors is limited to a small fraction of the total data. The total time complexity T_{total}, is composed of the linear time required for initial partitioning plus the reduced quadratic time for density estimation within the subsets. Consequently, the final time complexity of the proposed method is formally expressed as (Equation 11):

$$T_{total} = O(N \times D) + O(\frac{N^2}{t} \times D) \quad \textbf{(11)}$$

where N is the number of samples, D is the dimensionality, and t is the number of subsets (reduction factor). This formulation demonstrates that as the data volume and spatial distribution (t) increase, the computational cost remains significantly lower than the standard $O(N^2 \times D)$ approach.

4-Experiments setup

In order to evaluate the proposed methods comprehensively and fairly, their performance is compared with that of 7 recent related works and on a wide range of imbalanced datasets with different volumes and imbalance rates.

4.1-Datasets

A wide range of imbalanced datasets with different volumes and rates of imbalance are used in this study. The adult and healthy older people and kdd99 datasets (available at UCI [69]), breast cancer dataset [70], and 17 datasets (available at KEEL [71]) are the ones that are common in most related articles for evaluation. Most of these datasets have overlaps and challenging characteristics of class distribution. In this paper, datasets that have more than two classes are converted into two-class datasets. Table 3 shows the information about these datasets in detail. The im(including ROS, RUS, SMOTE, kmUnder, CNNbalance rate (IR) in the dataset can be calculated by using formula (12). $N_{majority}$ and $N_{minority}$ represent the number of minority and majority class samples, respectively.

$$IR = N_{majority}/N_{minority} \quad \textbf{(12)}$$

Table 3. Describes the Datasets

datasets	Size	No. features	IR
glass0	214	9	2.06
new-thyroid1	215	5	5.14
ecoli3	336	7	8.6
wisconsin	683	9	1.86
pima	768	8	1.87
vehicle0	846	18	3.25
yeast1	1484	8	2.46
ecoil2	336	5.46	7
new-thyroid2	215	5	5.14
segment0	2308	19	6.02
yeast3	1484	8	8.1
yeast-2_vs_4	1484	8	9.08
page-blacks0	5471	10	8.79
splice	3190	3	3.15
penbased	10992	10	9.41
nursery	12690	5	2.20
adult	32561	14	3.01

shuttle	58000	7	5.51
healthy older people	70742	9	2.53
Breast cancer	102294	117	163.2
kdd99-10%	494022	42	4.07

4.2.Parameters initialization

Density-based algorithms usually depend on two parameters: *neighborhood_radius* and *min_ponit*. These two parameters are initialized according to the goals we are pursuing.

Imbalanced datasets face a shortage of minority class samples. Hence, the minority class minpoint (*min_minority*) is set to 2. If the value of min_minority is considered small, a small amount of data will be labeled as noise. According to the goals of this algorithm and the experiments performed on different datasets, we consider the median of distance to the sixth neighbor of 2% of randomly selected samples from the minority class as the minority class radius (*r_ minority*) for our method to apply to different datasets and to be able to independently select the appropriate value for these parameters.

To set the *r_majority* and *min_ majority*, *min_ majority* is set to 2 to increase the speed of the data reduction process and *r_ majority* is equal to the median of distance to the $wneighbor^{th}$ neighbor of 2% of the majority class samples. According to the FCDM approach, if the number of data in each set decreases, the density is calculated faster. *wneighbor* indicates the number of samples that should be deleted, instead of min_ majority, until the dataset reaches the balance. *wneighbor* can be calculated from formula (13). In this formula, IR is the imbalance rate and Z is the appropriate balance rate for the final dataset. The value of Z is considered 60 to 40.

$$wneighbor = min_majority * IR * Z \quad \textbf{(13)}$$

4.3- Implementation

To implement and evaluate the proposed methods, MATLAB version 2021 is used on a server with 7 cores and 64 GB of RAM. For fair evaluation, the related methods presented in other studies (including ROS, RUS, SMOTE, kmUnder, CNN [72], K-US, and NB-Rec) were also implemented in MATLAB. The SVM and random forest classifiers were used to learn the preprocessed datasets by these methods. These classifiers are the most well-known and widely used classifiers in the field of imbalanced datasets. To implement a random forest, the number of trees is equal to 50.

4.4- Evaluation metrics

5 steps were performed to evaluate and compare the performance of the proposed method in comparison with the other related methods. In the first step, the imbalanced dataset is divided into two parts, training, and testing, in a ratio of 80 to 20. In the second step, the proposed method and other selected methods are applied to the training data to obtain a balanced and pre-processed dataset. In the next step, the classifiers are executed on the pre-processed datasets. In step 4, the model obtained by the classifier is run on the test dataset, and, then, the values of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) are calculated. In the last step, after calculating these 4 parameters, the values of the evaluation metrics can be calculated. The better the method used for data pre-processing and the better the

quality of the balanced data, the better the resulting classification model will be in terms of various evaluation metrics (especially in minority class detection).

Sensitivity and F1-score metrics are the ones used in most related papers [24, 45, 48] to evaluate the balancing methods. Therefore, they are also selected to evaluate the proposed method in this study. Sensitivity (Equation 14) is the most important and widely used evaluation metric to show minority class accuracy. Precision (Equation 15) shows the majority of class errors. Accuracy (Equation 17) represents the overall proportion of correctly classified samples across both classes. The value of the F1-score (Equation 16) metric is affected by sensitivity and precision. Since the proposed method is claimed to be fast and suitable for large datasets, execution time is also considered as an evaluation metric.

$$\text{Sensitivity} = TP/(TP + FN) \tag{14}$$

$$\text{Precision} = TP/(TP + FP) \tag{15}$$

$$\text{F1_score} = 2/(1/sensitivity + 1/precision) \tag{16}$$

$$Accuaracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{17}$$

4.5- Results and discussion

As mentioned before, the training dataset is 80% of the initial imbalanced dataset and the rest is the test one. The data set is randomly divided into two parts, training and testing. In order to minimize the effect of randomness in the results, the experiments were completely repeated 10 times and the presented results are the average numbers obtained in these repetitions. Also, in order to make the presented results fairer regarding the execution time of the algorithms, all the unnecessary services of the operating system and other programs have been disabled and the executions have been carried out under the same conditions.

After balancing the datasets by different methods (including the proposed method) and applying the SVM classifier, the resulting model was evaluated from the point of view of F1-score, sensitivity and accuracy metrics, the results can be seen in Tables 4, 5 and 6. Similarly, Tables 7, 8 and 9 show the values of these three metrics for the model resulting from the random forest classifier. The results of these tables show that the model obtained from both classifiers has a significant advantage when the output data of the proposed method is used as training data compared to competing methods.

As can be seen in these tables, the DBUSB algorithm significantly increases sensitivity and to some extent the F1-score index in both classifiers. The RUS and ROS algorithms do not have an intelligent approach; they act randomly and do not care about the characteristics of the class distribution. ROS and SMOTE algorithms use only the oversampling technique to balance the dataset, which makes it difficult to maintain, manage, and control the balanced dataset. Therefore, since these methods only try to create balance by creating data, they are not suitable for large imbalanced data. The KmUnder algorithm does not remove overlap, which has a significantly negative impact on learning quality. The NB-Rec and KUS algorithms do not balance the datasets properly. This conclusion can be made using the results presented in Figure 5; because the average learning time of the balanced dataset by these methods is higher than the other methods

that have used the undersampling technique [73-77]. This means that the resulting dataset is larger and the majority class is not well reduced.

Table 4. The value obtained for the Sensitivity index for different datasets using svm classifier

Dataset	RUS	ROS	SMOTE	CNN	kmUnder	KUS	NB-Rec	DBUSB
glass0	0.929	0.952	0.952	0.952	0.952	0.952	1	1
new-thyroid1	0.81	1	0.81	1	1	0.714	0.81	1
ecoli3	0.905	0.857	0.81	0.667	0.905	0.81	0.857	1
wisconsin	0.986	0.986	0.986	0.993	0.972	0.986	0.986	1
pima	0.568	0.716	0.685	0.648	0.623	0.722	0.938	1
vehicle0	0.992	0.967	0.95	0.992	1	1	1	1
yeast1	0.419	0.628	0.636	0.481	0.806	0.779	0.907	0.934
ecoil2	0.788	0.818	0.667	0.879	0.848	0.485	0.788	1
new-thyroid2	0.619	0.905	0.667	1	0.952	0.571	0.667	1
segment0	0.985	0.99	0.99	0.99	0.99	0.985	0.99	0.99
yeast3	0.848	0.859	0.788	0.798	0.97	0.707	0.828	0.848
yeast-2_vs_4	0.727	0.788	0.697	0.788	0.818	0.667	0.727	0.939
page-blacks0	0.682	0.732	0.616	0.813	0.735	0.586	0.664	0.795
splice	0.959	0.959	0.946	0.97	0.987	0.974	0.976	0.994
penbased	0.995	0.992	0.984	0.998	0.995	0.975	0.991	0.973
nursery	0.52	0.649	0.654	0.386	0.686	0.669	0.808	0.921
adult	0.748	0.747	0.749	0.715	0.877	0.822	0.905	0.845
shuttle	0.978	0.977	0.486	1	1	0	0.977	0.716
healthy older people	0.978	0.98	0.979	0.991	0.979	0.976	0.984	0.978
Breast cancer	0.987	0.988	0.986	0.986	0.991	0.974	0.991	0.994
kdd99_10%	0.996	0.996	0.995	1	**	0.993	0.996	0.992
average	0.829	0.880	0.811	0.859	0.904	0.778	0.895	**0.949**

Table 5. The value obtained for the F1-score index for different datasets using svm classifier

dataset	RUS	ROS	SMOTE	CNN	kmUnder	KUS	NB-Rec	DBUSB
glass0	0.765	0.619	0.619	0.597	0.604	0.619	0.614	0.609
new-thyroid1	0.895	0.979	0.895	0.356	0.979	0.833	0.895	1
ecoli3	0.574	0.619	0.734	0.667	0.493	0.734	0.651	0.678
wisconsin	0.96	0.956	0.956	0.963	0.953	0.956	0.956	0.97
pima	0.599	0.655	0.638	0.633	0.61	0.647	0.641	0.636
vehicle0	0.875	0.879	0.877	0.838	0.786	0.857	0.828	0.869
yeast1	0.51	0.594	0.584	0.547	0.58	0.58	0.554	0.591
ecoil2	0.721	0.717	0.643	0.671	0.683	0.554	0.706	0.957
new-thyroid2	0.765	0.95	0.8	0.292	0.925	0.727	0.8	0.875
segment0	0.948	0.992	0.992	0.884	0.93	0.99	0.978	0.99
yeast3	0.725	0.721	0.808	0.773	0.682	0.778	0.777	0.82
yeast-2_vs_4	0.69	0.708	0.768	0.541	0.6	0.775	0.779	0.74
page-blacks0	0.635	0.663	0.67	0.514	0.619	0.656	0.662	0.629
splice	0.891	0.899	0.903	0.902	0.89	0.892	0.889	0.898
penbased	0.914	0.928	0.962	0.87	0.896	0.957	0.945	0.952
nursery	0.536	0.557	0.559	0.374	0.562	0.559	0.535	0.551
adult	0.671	0.672	0.672	0.667	0.631	0.663	0.631	0.63
shuttle	0.722	0.722	0.512	0.266	0.641	nan	0.731	0.643
healthy older people	0.938	0.939	0.939	0.943	0.939	0.945	0.941	0.939
Breast cancer	0.975	0.969	0.975	0.975	0.972	0.964	0.959	0.979
kdd99_10%	0.986	0.989	0.989	0.33	**	0.988	0.988	0.979
average	0.776	0.797	0.785	0.648	0.749	0.784	0.784	**0.806**

Table 6. The value obtained for the accuracy index for different datasets using SVM classifier

dataset	RUS	ROS	SMOTE	CNN	kmUnder	KUS	NB-Rec	DBUSB
glass0	81.33%	76.53%	76.53%	74.31%	74.79%	76.53%	75.25%	75.31%
new-thyroid1	100.00%	99.07%	100.00%	40.85%	99.07%	95.34%	96.90%	100.00%
ecoli3	76.01%	80.89%	88.35%	85.04%	72.01%	88.35%	81.33%	89.28%
wisconsin	96.06%	95.77%	95.77%	96.34%	95.74%	95.77%	95.77%	97.08%
pima	63.88%	67.29%	66.21%	65.52%	63.81%	67.22%	67.93%	72.88%
vehicle0	85.34%	86.83%	86.80%	81.25%	77.29%	84.73%	81.65%	85.29%
yeast1	60.10%	59.88%	58.74%	60.14%	59.57%	58.85%	59.81%	64.91%
ecoil2	72.63%	72.76%	68.32%	72.71%	73.08%	62.24%	72.22%	96.41%
new-thyroid2	100.00%	98.15%	94.27%	34.20%	97.28%	93.01%	94.27%	96.38%
segment0	93.36%	99.42%	99.42%	84.38%	90.72%	99.30%	97.46%	99.14%
yeast3	67.16%	67.04%	73.30%	68.62%	64.30%	72.07%	71.07%	74.02%
yeast-2_vs_4	70.38%	71.26%	72.64%	53.68%	59.07%	76.71%	74.33%	78.47%
page-blacks0	65.23%	67.00%	69.37%	52.89%	62.65%	67.68%	67.36%	62.38%
splice	88.08%	88.89%	89.37%	89.28%	87.89%	88.07%	88.16%	89.04%
penbased	86.67%	88.59%	93.64%	79.59%	83.50%	92.59%	90.35%	92.14%
nursery	62.91%	63.80%	63.79%	53.15%	62.94%	63.51%	64.67%	66.86%
adult	67.92%	68.04%	67.98%	66.97%	64.31%	67.24%	65.81%	64.55%
shuttle	67.22%	67.24%	64.91%	29.87%	58.75%	N/A	67.89%	65.55%
healthy older people	93.28%	93.30%	93.34%	93.59%	93.35%	93.99%	93.54%	93.39%
Breast cancer	96.33%	95.15%	96.42%	96.42%	95.45%	95.41%	92.98%	96.44%
kdd99_10%	98.34%	98.71%	98.74%	48.72%	N/A	98.66%	98.60%	97.62%
average	80.58%	81.22%	81.81%	67.98%	76.78%	81.86%	80.83%	83.67%

Table 7. The value obtained for the Sensitivity index for different datasets using random forest classifier

dataset	RUS	ROS	SMOTE	CNN	kmUnder	KUS	NB-Rec	DBUSB
glass0	0.833	0.833	0.833	0.81	0.857	0.857	0.857	0.905
new-thyroid1	1	0.905	0.952	1	1	0.952	0.952	1
ecoli3	0.905	0.571	0.524	0.619	0.857	0.667	0.81	0.952
wisconsin	0.979	0.979	0.986	0.986	0.986	1	1	1
pima	0.642	0.667	0.648	0.691	0.728	0.815	0.883	1
vehicle0	0.983	0.967	0.958	0.958	0.992	0.992	0.992	0.983
yeast1	0.651	0.605	0.605	0.671	0.736	0.798	0.845	0.95
ecoil2	0.788	0.758	0.697	0.848	0.758	0.697	0.788	0.848
new-thyroid2	0.952	0.857	0.905	1	1	0.952	0.952	1
segment0	0.99	0.99	0.99	0.985	0.99	0.98	1	0.985
yeast3	0.889	0.798	0.768	0.788	0.96	0.869	0.869	0.919
yeast-2_vs_4	0.879	0.636	0.697	0.788	0.939	0.758	0.758	0.97
page-blacks0	0.946	0.878	0.878	0.926	0.973	0.929	0.92	0.973
splice	0.978	0.942	0.939	0.972	0.989	0.978	0.989	0.996
penbased	0.994	0.984	0.981	0.994	0.989	0.989	0.986	0.984
nursery	0.647	0.74	0.474	0.483	0.824	0.91	0.944	1
adult	0.775	0.715	0.691	0.699	0.865	0.862	0.891	0.858
shuttle	1	1	1	1	1	1	1	1
healthy older people	0.995	0.994	0.994	0.995	0.994	0.997	0.996	0.978
Breast cancer	0.982	0.983	0.983	0.983	0.986	0.979	0.987	0.991
kdd99_10%	1	1	1	1	**	1	1	0.998
average	0.896	0.848	0.833	0.866	0.921	0.904	0.925	**0.966**

Table 8. The value obtained for the F1-score index for different datasets using random forest classifier

dataset	RUS	ROS	SMOTE	CNN	kmUnder	KUS	NB-Rec	DBUSB
glass0	0.826	0.846	0.826	0.788	0.707	0.717	0.686	0.823
new-thyroid1	0.957	0.95	0.976	0.979	0.822	0.934	0.955	0.957
ecoli3	0.689	0.622	0.63	0.567	0.547	0.625	0.549	0.745
wisconsin	0.953	0.953	0.957	0.957	0.956	0.96	0.96	0.96
pima	0.645	0.655	0.633	0.663	0.692	0.659	0.65	0.668
vehicle0	0.892	0.922	0.924	0.906	0.892	0.845	0.843	0.914
yeast1	0.609	0.614	0.612	0.61	0.593	0.593	0.578	0.6
ecoil2	0.757	0.78	0.786	0.811	0.74	0.787	0.826	0.803
new-thyroid2	0.955	0.923	0.95	0.979	0.814	0.976	0.976	1
segment0	0.985	0.99	0.992	0.975	0.978	0.982	0.993	0.995
yeast3	0.735	0.78	0.769	0.767	0.727	0.764	0.749	0.812
yeast-2_vs_4	0.714	0.68	0.706	0.753	0.737	0.734	0.728	0.764
page-blacks0	0.822	0.893	0.889	0.876	0.781	0.859	0.853	0.817
splice	0.94	0.944	0.944	0.934	0.926	0.94	0.933	0.919
penbased	0.978	0.991	0.99	0.981	0.967	0.987	0.989	0.985
nursery	0.578	0.631	0.468	0.475	0.675	0.687	0.721	0.717
adult	0.7	0.7	0.693	0.686	0.676	0.688	0.67	0.679
shuttle	0.999	0.999	0.999	1	1	0.999	0.999	0.999
healthy older people	0.984	0.986	0.987	0.982	0.987	0.97	0.982	0.981
Breast cancer	0.984	0.982	0.983	0.983	0.982	0.952	0.98	0.986
kdd99_10%	0.999	0.999	0.999	0.999	**	0.998	0.999	0.997
average	0.843	0.850	0.843	0.841	0.810	0.841	0.839	**0.863**

Table 9. The value obtained for the accuracy index for different datasets using random forest classifier

dataset	RUS	ROS	SMOTE	CNN	kmUnder	KUS	NB-Rec	DBUSB
glass0	82.26%	85.06%	82.26%	77.89%	65.26%	66.24%	63.16%	82.22%
new-thyroid1	98.60%	98.24%	99.07%	99.30%	93.73%	97.31%	99.16%	98.60%
ecoli3	71.30%	67.12%	71.91%	64.91%	61.34%	64.35%	61.12%	79.52%
wisconsin	95.34%	95.34%	95.69%	95.69%	95.53%	96.12%	96.12%	96.12%
pima	64.65%	65.37%	62.72%	66.02%	68.61%	63.85%	63.13%	66.36%
vehicle0	86.83%	90.72%	91.07%	88.85%	87.27%	82.02%	81.82%	89.47%
yeast1	61.42%	61.76%	61.47%	61.90%	61.03%	60.15%	59.43%	63.09%
ecoil2	74.45%	78.50%	81.99%	81.63%	73.84%	82.35%	82.11%	79.47%
new-thyroid2	95.77%	96.88%	98.15%	99.27%	92.57%	99.07%	99.07%	100.00%
segment0	98.20%	99.00%	99.34%	96.99%	97.24%	98.33%	99.25%	99.73%
yeast3	67.75%	76.77%	77.01%	75.31%	66.86%	72.84%	70.93%	76.72%
yeast-2_vs_4	66.50%	71.84%	71.19%	73.18%	69.41%	72.03%	71.18%	71.86%
page-blacks0	76.70%	90.47%	89.81%	84.72%	72.66%	82.44%	81.82%	75.83%
splice	92.52%	94.49%	94.63%	91.95%	90.37%	92.54%	91.13%	89.44%
penbased	96.65%	99.64%	99.67%	97.05%	95.04%	98.54%	99.12%	98.58%
nursery	62.00%	65.61%	61.21%	61.88%	69.09%	70.07%	73.18%	74.19%
adult	68.32%	69.46%	69.38%	68.04%	64.93%	66.52%	64.21%	65.25%
shuttle	99.85%	99.85%	99.85%	100.00%	100.00%	99.85%	99.85%	99.85%
healthy older people	97.94%	98.26%	98.41%	97.66%	98.41%	96.06%	97.63%	98.22%
Breast cancer	98.60%	98.12%	98.31%	98.31%	97.83%	92.70%	97.34%	98.13%
kdd99_10%	99.84%	99.84%	99.84%	99.84%	N/A	99.68%	99.84%	99.64%
average	83.60%	85.83%	85.86%	84.78%	81.05%	83.48%	83.36%	85.82%

Table 10. The execution time of methods to preprocess each dataset

dataset	RUS	ROS	SMOTE	CNN	kmUnder	KUS	NB-Rec	DBUSB
glass0	0.0025	0.0012	0.0179	0.6318	0.0673	0.0049	0.1209	0.0292
new-thyroid1	0.0005	0.0009	0.0046	0.7542	0.0132	0.0032	0.0698	0.0258
ecoli3	0.0004	0.0002	0.0049	1.2192	0.0096	0.0036	0.1082	0.0363
wisconsin	0.0018	0.0003	0.0288	1.6532	0.0606	0.0118	0.2367	0.1204
pima	0.0023	0.0002	0.0312	2.0294	0.0418	0.0168	0.2405	0.1056
vehicle0	0.0019	0.0007	0.0239	2.4577	0.0368	0.0112	0.2882	0.0936
yeast1	0.0037	0.0002	0.0506	4.6531	0.0824	0.0219	0.4872	0.1876
ecoil2	0.0004	0.0003	0.0052	1.0032	0.0115	0.0032	0.0985	0.0278
new-thyroid2	0.0003	0.0001	0.0034	0.6586	0.0075	0.0027	0.0783	0.0241
segment0	0.0075	0.0005	0.041	7.2949	0.0798	0.0251	0.732	0.2043
yeast3	0.002	0.0009	0.0139	5.5482	0.0473	0.0108	0.4951	0.1505
yeast-2_vs_4	0.0005	0.0002	0.0053	1.7265	0.0116	0.0034	0.1421	0.0446
page-blacks0	0.0312	0.0159	0.0645	22.6821	0.2509	0.0509	1.7427	0.5381
splice	0.0387	0.0048	0.2038	10.5731	0.3484	0.0786	2.6603	0.5714
penbased	0.0996	0.0379	0.201	39.6756	1.6731	0.5707	6.5029	1.4154
nursery	0.1521	0.0011	1.044	105.773	1.9318	0.3738	5.4043	2.7283
adult	1.9953	0.0148	10.4981	147.4218	20.5419	7.1645	40.5913	13.0137
shuttle	2.0646	9.211	6.6075	754.823	36.5848	11.2929	80.0927	8.6212
healthy older people	22.0543	20.0054	58.1692	1703.4	79.7646	24.8498	145.8171	13.6719
Breast cancer	3311.915	3019.187	4588.879	9740.176	7376.299	2574.233	11587.81	4222.733
kdd99_10%	3752.614	3149.825	6126.881	15062.5	**	2876.57	52408.68	4701.165

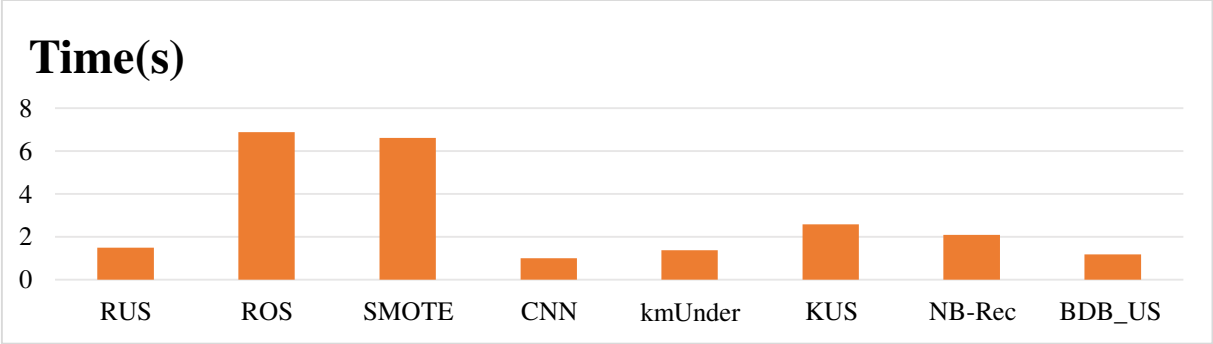


Fig 6. The average learning time of final preprocessed datasets using svm and random forest classifiers.

Table 10 shows the required execution time of different methods for preprocessing and balancing each dataset. Figure 6 shows the average learning time for SVM and random forest classifiers on the output data of different preprocessing methods. The longer this time, the larger the pre-processed data set. For large datasets, reducing the amount of data as much as possible while balancing and having high learning quality can be a significant advantage. According to Table 10, the proposed algorithm requires less execution time than SMOTE, NB-Rec, kmUnder, and CNN algorithms. The KUS algorithm also spends more time on datasets with low imbalance rates than the proposed algorithm because the execution time of KUS depends on the number of samples in the minority class. But as mentioned, this method compared to the proposed method, while providing a larger output data set, does not cope well with overlap data and class balancing problems. It should be noted that for the kdd99_10% dataset, the KmUnder algorithm could not be executed by the specified system due to high memory requirements.

6-Conclusions

In this paper, a rapid method was presented for calculating data density and a density-based algorithm to deal with large imbalanced datasets. In this algorithm, the goal was to reduce the majority class samples in the highly dense regions until the balance was reached. This proposed algorithm also removes overlap and noise.

In order to comprehensively and fairly evaluate the proposed method, 21 imbalanced datasets that have different imbalance rates and volumes were employed. The preprocessed dataset by the proposed algorithm and 7 compared methods were modeled using SVM and random forest classifiers. Sensitivity, F1_score and accuracy metrics were calculated for the obtained models. The experiment also considered the execution time of preprocessing methods and the average learning time of final preprocessed datasets using two classifiers. The proposed algorithm with high preprocessing speed increased sensitivity while maintaining the structure of the two classes maximally. In fact, the proposed method tries to increase the learning quality of the minority class while removing noise data and solving the overlap problem of classes. On the other hand, due to the high speed in calculating the density, it can have a low execution time while achieving good quality. Due to the relatively good reduction of data volume and the reduction of learning time, this method can be considered as a suitable method for large datasets.

Acknowledgments:

Not applicable.

Funding Information:

The authors received no financial support for conducting this research.

Data Availability Statement:

The dataset link is available at: https://doi.org/10.5281/zenodo.18057270

Competing Interests:

The authors declare no competing interests.

Funding:

No funding was received for this research.

Author contributions:

Sima Mayabadi: Investigation, writing the manuscript, software, and experiments.

Hamid Saadatfar: Supervision, investigation, validation, and editing.

Emrah Arslan: Investigation and editing.

Bulbula Kumeda: Investigation and editing.

Javad Hassannataj Joloudari: Investigation and editing.

References

[1] S. Zeraatkar and F. Afsari, "Interval-valued fuzzy and intuitionistic fuzzy–KNN for imbalanced data classification," *Expert Systems with Applications,* vol. 184, p. 115510, 2021.

[2] A. N. Tarekegn, M. Giacobini, and K. Michalak, "A review of methods for imbalanced multi-label classification," *Pattern Recognition,* vol. 118, p. 107965, 2021.

[3] J. Liu, "A minority oversampling approach for fault detection with heterogeneous imbalanced data," *Expert Systems with Applications,* vol. 184, p. 115492, 2021.

[4] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Information sciences,* vol. 250, pp. 113-141, 2013.

[5] S. Mayabadi and H. Saadatfar, "Two density-based sampling approaches for imbalanced and overlapping data," *Knowledge-Based Systems,* vol. 241, p. 108217, 2022.

[6] S.-J. Yen and Y.-S. Lee, "Cluster-based under-sampling approaches for imbalanced data distributions," *Expert Systems with Applications,* vol. 36, no. 3, pp. 5718-5727, 2009.

[7] C.-F. Tsai, W.-C. Lin, Y.-H. Hu, and G.-T. Yao, "Under-sampling class imbalanced datasets by combining clustering analysis and instance selection," *Information Sciences,* vol. 477, pp. 47-54, 2019, doi: https://doi.org/10.1016/j.ins.2018.10.029.

[8] H. Guo, X. Diao, and H. Liu, "Improving undersampling-based ensemble with rotation forest for imbalanced problem," *Turkish Journal of Electrical Engineering & Computer Sciences,* vol. 27, no. 2, pp. 1371-1386, 2019.

[9] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Systems with Applications,* vol. 73, pp. 220-239, 2017.

[10] B. Krawczyk, M. Galar, Ł. Jeleń, and F. Herrera, "Evolutionary undersampling boosting for imbalanced classification of breast cancer malignancy," *Applied Soft Computing,* vol. 38, pp. 714-726, 2016.

[11] H. Patel, D. Singh Rajput, G. Thippa Reddy, C. Iwendi, A. Kashif Bashir, and O. Jo, "A review on classification of imbalanced data for wireless sensor networks," *International Journal of Distributed Sensor Networks,* vol. 16, no. 4, p. 1550147720916404, 2020.

[12] S. u. Korkmaz, "Deep learning-based imbalanced data classification for drug discovery," *Journal of Chemical Information and Modeling,* vol. 60, no. 9, pp. 4180-4190, 2020.

[13] D. Gan, J. Shen, B. An, M. Xu, and N. Liu, "Integrating TANBN with cost sensitive classification algorithm for imbalanced data in medical diagnosis," *Computers & Industrial Engineering,* vol. 140, p. 106266, 2020.

[14] Z. Xu, D. Shen, T. Nie, and Y. Kou, "A hybrid sampling algorithm combining M-SMOTE and ENN based on Random Forest for medical imbalanced data," *Journal of Biomedical Informatics,* vol. 107, p. 103465, 2020.

[15] Y.-S. Chen, "An empirical study of a hybrid imbalanced-class DT-RST classification procedure to elucidate therapeutic effects in uremia patients," *Medical & biological engineering & computing,* vol. 54, no. 6, pp. 983-1001, 2016.

[16] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, and L. Liang, "Self-balancing federated learning with global imbalanced data in mobile systems," *IEEE Transactions on Parallel and Distributed Systems,* vol. 32, no. 1, pp. 59-71, 2020.

[17] V. López, A. Fernández, J. G. Moreno-Torres, and F. Herrera, "Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. Open problems on intrinsic data characteristics," *Expert Systems with Applications,* vol. 39, no. 7, pp. 6585-6608, 2012.

[18] G. E. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD explorations newsletter,* vol. 6, no. 1, pp. 20-29, 2004.

[19] E. Ramentol *et al.*, "IFROWANN: imbalanced fuzzy-rough ordered weighted average nearest neighbor classification," *IEEE Transactions on Fuzzy Systems,* vol. 23, no. 5, pp. 1622-1637, 2014.

[20] P. Domingos, "Metacost: A general method for making classifiers cost-sensitive," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 1999, pp. 155-164.

[21] C. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on Big Data," *Information sciences,* vol. 275, pp. 314-347, 2014.

[22] X. Wu and J. A. O'Sullivan, "Fellow," *IEEE, Xingquan Zhu, Gong-Qing Wu, and Wei Ding" Data Mining with Big Data" IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING,* vol. 26, no. 1, 2014.

[23] H. K. Lee and S. B. Kim, "An overlap-sensitive margin classifier for imbalanced and overlapping data," *Expert Systems with Applications,* vol. 98, pp. 72-83, 2018.

[24] M. M. Nwe and K. T. Lynn, "KNN-based overlapping samples filter approach for classification of imbalanced data," in *International Conference on Software Engineering Research, Management and Applications*, 2019: Springer, pp. 55-73.

[25] S. Visa and A. Ralescu, "Learning imbalanced and overlapping classes using fuzzy sets," in *Proceedings of the ICML*, 2003, vol. 3, pp. 97-104.

[26] J. Stefanowski, "Overlapping, rare examples and class decomposition in learning classifiers from imbalanced data," in *Emerging paradigms in machine learning*: Springer, 2013, pp. 277-306.

[27] V. García, R. A. Mollineda, and J. S. Sánchez, "On the k-NN performance in a challenging scenario of imbalance and overlapping," *Pattern Analysis and Applications,* vol. 11, no. 3, pp. 269-280, 2008.

[28] S. Das, S. Datta, and B. B. Chaudhuri, "Handling data irregularities in classification: Foundations, trends, and future challenges," *Pattern Recognition,* vol. 81, pp. 674-693, 2018.

[29] Koziarski, Michał, Michał Woźniak, and Bartosz Krawczyk. "Combined cleaning and resampling algorithm for multi-class imbalanced data with label noise." Knowledge-Based Systems 204 (2020): 106223.

[30] Pei, Wenbin, Ruohao Dai, Bing Xue, Mengjie Zhang, Qiang Zhang, Yiu-Ming Cheung, and Shuyin Xia. "EvoSampling: A Granular Ball-based Evolutionary Hybrid Sampling with Knowledge Transfer for Imbalanced Learning." arXiv preprint arXiv:2412.10461 (2024).

[31] Elreedy, Dina, Amir F. Atiya, and Firuz Kamalov. "A theoretical distribution analysis of synthetic minority oversampling technique (SMOTE) for imbalanced learning." Machine Learning 113, no. 7 (2024): 4903-4923.

[32] Raslan, Khaled SH, Almohammady S. Alsharkawy, and K. R. Raslan. "iHHO-SMOTe: A cleansed approach for handling outliers and reducing noise to improve imbalanced data classification." arXiv preprint arXiv:2504.12850 (2025).

[33] Leng, Qiangkui, Jiamei Guo, Jiaqing Tao, Xiangfu Meng, and Changzhong Wang. "OBMI: oversampling borderline minority instances by a two-stage Tomek link-finding procedure for class imbalance problem." Complex & Intelligent Systems 10, no. 4 (2024): 4775-4792.

[34] I. Tomek, "Two modifications of CNN," 1976.

[35] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Transactions on Systems, Man, and Cybernetics,* no. 3, pp. 408-421, 1972.

[36] Pereira, Rodolfo M., Yandre MG Costa, and Carlos N. Silla Jr. "MLTL: A multi-label approach for the Tomek Link undersampling algorithm." Neurocomputing 383 (2020): 95-105.

[37] Swana, Elsie Fezeka, Wesley Doorsamy, and Pitshou Bokoro. "Tomek link and SMOTE approaches for machine fault classification with an imbalanced dataset." Sensors 22, no. 9 (2022): 3246.

[38] Penumala, S., and K. Santhi Sree. "A comparative study of sampling techniques for imbalanced credit card fraud detection." International Journal for Research in Applied Science and Engineering Technology 12, no. 7 (2024): 789-800.

[39] Q. Dai, J.-w. Liu, and Y. Liu, "Multi-granularity relabeled under-sampling algorithm for imbalanced data," *Applied Soft Computing,* p. 109083, 2022.

[40] Khan, Azal Ahmad, Omkar Chaudhari, and Rohitash Chandra. "A review of ensemble learning and data augmentation models for class imbalanced problems: Combination, implementation and evaluation." Expert Systems with Applications 244 (2024): 122778.

[41] A. Guzmán-Ponce, J. S. Sánchez, R. M. Valdovinos, and J. R. Marcial-Romero, "DBIG-US: A two-stage under-sampling algorithm to face the class imbalance problem," *Expert Systems with Applications,* vol. 168, p. 114301, 2021.

[42] C. Bunkhumpornpat and K. Sinapiromsaran, "DBMUTE: density-based majority under-sampling technique," *Knowledge and Information Systems,* vol. 50, no. 3, pp. 827-850, 2017.

[43] M. Beckmann, N. F. Ebecken, and B. S. P. de Lima, "A KNN undersampling approach for data balancing," *Journal of Intelligent Learning Systems and Applications,* vol. 7, no. 04, p. 104, 2015.

[44] P. Ghaderyan, A. Abbasi, and M. H. Sedaaghi, "An efficient seizure prediction method using KNN-based undersampling and linear frequency measures," *Journal of neuroscience methods,* vol. 232, pp. 134-142, 2014.

[45] P. Vuttipittayamongkol and E. Elyan, "Neighbourhood-based undersampling approach for handling imbalanced and overlapped data," *Information Sciences,* vol. 509, pp. 47-70, 2020.

[46] S. Goyal, "Handling class-imbalance with KNN (neighbourhood) under-sampling for software defect prediction," *Artificial Intelligence Review,* vol. 55, no. 3, pp. 2023-2064, 2022.

[47] V. S. Akondi, V. Menon, J. Baudry, and J. Whittle, "Novel K-means clustering-based undersampling and feature selection for drug discovery applications," in *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2019: IEEE, pp. 2771-2778.

[48] W.-C. Lin, C.-F. Tsai, Y.-H. Hu, and J.-S. Jhang, "Clustering-based undersampling in class-imbalanced data," *Information Sciences,* vol. 409, pp. 17-26, 2017.

[49] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of artificial intelligence research,* vol. 16, pp. 321-357, 2002.

[50] Mujahid, Muhammad, E. R. O. L. Kına, Furqan Rustam, Monica Gracia Villar, Eduardo Silva Alvarado, Isabel De La Torre Diez, and Imran Ashraf. "Data oversampling and imbalanced datasets: an investigation of performance for machine learning and feature engineering." Journal of Big Data 11, no. 1 (2024): 87.

[51] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning," in *International conference on intelligent computing*, 2005: Springer, pp. 878-887.

[52] J. A. Sáez, J. Luengo, J. Stefanowski, and F. Herrera, "SMOTE–IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering," *Information Sciences,* vol. 291, pp. 184-203, 2015.

[53] S. Feng, J. Keung, X. Yu, Y. Xiao, and M. Zhang, "Investigation on the stability of SMOTE-based oversampling techniques in software defect prediction," *Information and Software Technology,* vol. 139, p. 106662, 2021.

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

[55] G. Douzas, F. Bacao, and F. Last, "Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE," *Information Sciences,* vol. 465, pp. 1-20, 2018.

[56] P. Chaudhari, H. Agarwal, and V. Bhateja, "Data augmentation for cancer classification in oncogenomics: an improved KNN based approach," *Evolutionary Intelligence,* vol. 14, no. 2, pp. 489-498, 2021.

[57] M. Zaharia *et al.*, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*, 2012, pp. 15-28.

[58] S. Del Río, V. López, J. M. Benítez, and F. Herrera, "On the use of mapreduce for imbalanced big data using random forest," *Information Sciences,* vol. 285, pp. 112-137, 2014.

[59] R. C. Bhagat and S. S. Patil, "Enhanced SMOTE algorithm for classification of imbalanced big-data using random forest," in *2015 IEEE International Advance Computing Conference (IACC)*, 2015: IEEE, pp. 403-408.

[60] I. Triguero *et al.*, "Evolutionary undersampling for imbalanced big data classification," in *2015 IEEE Congress on Evolutionary Computation (CEC)*, 2015: IEEE, pp. 715-722.

[61] Eroğlu, Duygu Yılmaz, and Mestan Şahin Pir. "Hybrid oversampling and undersampling method (HOUM) via safe-level SMOTE and support vector machine." (2024).

[62] Alamri, Maram, and Mourad Ykhlef. "Hybrid undersampling and oversampling for handling imbalanced credit card data." IEEE Access 12 (2024): 14050-14060.

[63] Hou, Guodong, Dong Ling Tong, Soung Yue Liew, and Peng Yin Choo. "Comparative Analysis of Resampling Techniques for Class Imbalance in Financial Distress Prediction Using XGBoost." Mathematics 13, no. 13 (2025): 2186.

[64] Hassani, Hossein, Mohammad Reza Entezarian, Sara Zaeimzadeh, Leila Marvian, and Nadejda Komendantova."An oversampling-undersampling strategy for large-scale data linkage." Frontiers in Big Data 8 (2025): 1542483.

[65] J. M. Johnson and T. M. Khoshgoftaar, "Thresholding strategies for deep learning with highly imbalanced big data," in *Deep Learning Applications, Volume 2*: Springer, 2021, pp. 199-227.

[66] E. M. Hassib, A. I. El-Desouky, L. M. Labib, and E.-S. M. El-kenawy, "WOA+ BRNN: An imbalanced big data classification framework using Whale optimization and deep neural network," *soft computing,* vol. 24, no. 8, pp. 5573-5592, 2020.

[67] G. Yang, G. Chen, C. Li, J. Fu, Y. Guo, and H. Liang, "Convolutional rebalancing network for the classification of large imbalanced rice pest and disease datasets in the field," *Frontiers in Plant Science,* p. 1150, 2021.

[68] F. Rodríguez-Torres, J. F. Martínez-Trinidad, and J. A. Carrasco-Ochoa, "An Oversampling Method for Class Imbalance Problems on Large Datasets," *Applied Sciences,* vol. 12, no. 7, p. 3424, 2022.

[69] https://archive.ics.uci.edu/ml/datasets.php (accessed june, 2020).

[70] "KDD Cup." https://www.kdd.org/kdd-cup (accessed june, 2020).

[71] A. F. J. Alcalá-Fdez, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera. "KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework." http://www.keel.es/ (accessed.

[72] P. Hart, "The condensed nearest neighbor rule (corresp.)," *IEEE transactions on information theory,* vol. 14, no. 3, pp. 515-516, 1968.

[73] Z. Sun *et al.*, "Undersampling method based on minority class density for imbalanced data," *Expert Systems with Applications*, vol. 249, p. 123328, 2024.

[74] K. Bhandari, K. Kumar, and A. L. Sangal, "DBOS_US: A density-based graph under-sampling method to handle class imbalance and class overlap issues in software fault prediction," *The Journal of Supercomputing*, vol. 80, no. 15, pp. 22682-22725, 2024.

[75] D. Lee and H. Kim, "Adaptive oversampling via density estimation for online imbalanced classification," *Information*, vol. 16, no. 1, p. 23, 2025.

[76] Z. Ahmed and S. Das, "Association rule based undersampling technique for addressing imbalanced classification," *2024 IEEE Silchar Subsection Conference (SILCON 2024)*, IEEE, 2024.

[77] X. Yang *et al.*, "An oversampling algorithm for high-dimensional imbalanced learning with class overlapping," *Knowledge and Information Systems*, pp. 1-29, 2024.