applied sciences

MDPI

Article

An Undersampling Method Approaching the Ideal Classification Boundary for Imbalance Problems

Wensheng Zhou [1,2], Chen Liu [1,2], Peng Yuan [3] and Lei Jiang [3,*]

[1] National Key Laboratory of Offshore Oil and Gas Exploitation, Beijing 100028, China; zhouwsh@cnooc.com.cn (W.Z.); liuchen4@cnooc.com.cn (C.L.)
[2] CNOOC Research Institute Ltd., Beijing 100028, China
[3] School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China; mpenzlp@163.com
* Correspondence: jleihn@hnust.edu.cn

Abstract: Data imbalance is a common problem in most practical classification applications of machine learning, and it may lead to classification results that are biased towards the majority class if not dealt with properly. An effective means of solving this problem is undersampling in the borderline area; however, it is difficult to find the area that fits the classification boundary. In this paper, we present a novel undersampling framework, whereby the clustering of samples in the majority class is conducted and segmentation is then performed in the boundary area according to the clusters obtained; this enables a better shape that fits the classification boundary to be obtained via the performance of random sampling in the borderline area of these segments. In addition, we hypothesize that there exists an optimal number of classifiers to be integrated into the method of ensemble learning that utilizes multiple classifiers that have been obtained via sampling to promote the algorithm. After passing the hypothesis test, we apply the improved algorithm to the newly developed method. The experimental results show that the proposed method works well.

Keywords: classification; cluster-based undersampling; imbalanced problem; optimal number of classifiers

check for updates

Citation: Zhou, W.; Liu, C.; Yuan, P.; Jiang, L. An Undersampling Method Approaching the Ideal Classification Boundary for Imbalance Problems. *Appl. Sci.* **2024**, *14*, 5421. https://doi.org/10.3390/app14135421

Academic Editor: Luis Javier Garcia Villalba

Received: 14 May 2024
Revised: 14 June 2024
Accepted: 20 June 2024
Published: 22 June 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In the era of big data, classification has become an increasingly important learning task in various data mining and analysis applications, including bankruptcy prediction [1], stroke prediction [2], fraud detection [3], and fault diagnosis [4]. Generally speaking, the classification algorithms employed in machine learning are designed to deal with balanced data classification problems. In reality, however, the data collected in practical applications are usually imbalanced, i.e., the number of samples representing different classes is unequal. For example, in a binary classification problem, there are 10 samples in one class and 500 in the other class. When the two types of data overlap one another, the classification algorithm may treat these 10 samples as noise data. It is therefore difficult to predict the minority-class samples when using classification algorithms, as the algorithm tends to learn the characteristics of the majority-class data; this results in the minority-class samples being misclassified as majority-class samples. For example, in practical applications such as cancer prediction, there are far more normal cells than cancer cells. However, there would be serious consequences if the cancer cells were to be misclassified as normal cells.

It is well known that imbalanced problems can be solved via two methods of research: one is oversampling [5,6], whereby data sets are balanced via the random generation of samples for the minority class, and the other is undersampling, whereby data sets are balanced by performing partial sampling in the majority class and the number of samples selected is therefore reduced [7–9]. Although both methods have enabled significant research results to be obtained, we will focus on the undersampling method in this paper.

Great success has been achieved by conducting random sampling and removing samples from the majority class to maintain the balance of data sets [7,9]. Experiments conducted by [10], [11], and [12] proved that not all data have a significant effect on the classification model, and that the contribution of some data is even negligible. Researchers have screened data based on the borderline [13], sensitivity [14], and distribution of data overlap [7] to obtain a new, balanced data set; these methods, unfortunately, may compromise the original distribution characteristics of the data set. Therefore, the cluster-based undersampling technique emerged; for example, [8] obtained good results showing that the majority class was partitioned into clusters. It is clear that regardless of the data sampling method used, it will eventually affect the generation of the classifier, and then affect the shape of the classification boundary. Therefore, our goal is to find an undersampling method that makes the distribution of samples conform to that at the ideal classification boundary.

The main contributions of this paper are as follows:

(1) Identifying a sampling area that is more consistent with the ideal classification boundary shape via the performance of segmentation after clustering and space compression;
(2) Ensuring that the distribution of the sampled data coincides with the sample distribution at the ideal classification boundary by optimizing the number of classifiers obtained and performing undersampling during ensemble learning;
(3) Demonstrating that the proposed method has a clear effect via the performance of experiments using 20 data sets.

In the remainder of this paper, Section 2 discusses related works, Section 3 gives a detailed description of our proposed model, and Section 4 presents extensive experimental results in order to justify the effectiveness of our method. The conclusions and future research directions are outlined in Section 5.

2. Related Work

In this paper, we propose a method that first removes noise, performs undersampling that is based on clustering, and uses ensemble learning to complete the final classification task. Correspondingly, we will introduce relevant work from the following three aspects.

2.1. Noise Filtering in Undersampling

Data imbalance is caused by a significant difference in the number of objects between different classes in a data set. Undersampling, which is able to solve the problem of imbalance, discards certain data in the majority class and forms a new data set with balanced majority-class data and minority-class data [9,15]. In addition, noise will affect the classification result. For example, when there is noise in the majority class and if the undersampling method incorporates noise into the balanced new data set, the accuracy of the classification will be greatly reduced. Although the ensemble learning method can be used alongside multiple sampling in order to weaken this effect, better classification results can be obtained if the noise is eliminated at the very beginning. Therefore, many researchers have introduced noise filtering into the process of undersampling.

Van et al. [10] proposed the use of a threshold adjustment strategy in order to filter data noise. Sáez et al. [11] removed data noise by using an iterative integrated noise filter based on the SMOTE method. Kang et al. [12] addressed the incorporation of a kNN filter into the undersampling method in order to exclude noise samples in the minority class. Yan et al. [5] used the semantic relationship between the attributes of the problem itself to aid in the identification of noise. Indeed, kNN is a good preprocessing method for imbalance classification as long as the noise does not interfere significantly with the results.

2.2. Cluster-Based Sampling Methods

In contrast to random sampling, the cluster-based undersampling method can preserve the distribution characteristics of the data set by dividing the majority class into groups of similar samples through cluster analysis; as a result, the data characteristics of the majority

class are maintained as long as the samples are drawn from these different groups to form a new, balanced data set.

Yen et al. [16] used Kmeans to cluster the majority classes and then randomly sampled the representative objects from clusters of the majority classes according to the proportion of the number of each cluster. Lin et al. [17] set the number of clusters in the majority class to the same number of samples in the minority class; they then used the center point of these clusters and the minority class to form a new data set. Tsai et al. [18] proposed a clustering-based combination method named CBIS that first uses the AP algorithm to perform clustering in the majority class and then uses sample selection techniques to obtain a new data set; it then achieves the final results via ensemble learning. Herein, the number of clusters is adaptively obtained. Li et al. [19] utilized the hierarchical clustering of undersampled fused random forests. The proposed method clusters the majority of samples using a hierarchical clustering algorithm, undersamples each cluster to achieve the balance of data samples, and then constructs a random forest. Compared to the method comprising random undersampling combined with random forest (RF), it improves the prediction accuracy by 16% and the F-value by 17%. Jang and Kim [20] proposed a new method that enables the self-organizing mapping of boundary regions to be described using a normal distribution, and that addresses high-dimensional imbalance problems. This method has been shown to perform well on two high-dimensional data sets used in industrial fault detection. Farshidvard et al. [8] performed clustering in the majority class such that there were no minority samples in the convex hull of each cluster. They believe that this approach can preserve the data distribution in the feature space and achieve good results. Devi et al. [21] presented an algorithm that uses the adaptive clustering method combined with AdaBoost to eliminate the majority of samples with minor contributions; this method therefore aids in the classification process. Guzmán-Ponce et al. [22] adopted a two-stage method that aims to overcome the problem of imbalance by combining DBSCAN and a graph-based process to filter noisy objects in the majority class. Tahfim and Chen [23] used the k-prototypes clustering algorithm to partition majority-class samples and perform initial undersampling, followed by resampling using ADASYN, NearMiss-2, and SMOTETomek. This method has achieved good results when applied to imbalanced large truck collision data.

The majority of the above-mentioned studies present methods that aim to accurately obtain the number of clusters in the majority class. This is, however, a paradox, because if the data of the majority class can be clearly distinguished from each other, then it is more reasonable to divide the majority class into several classes so that a majority class no longer exists. In addition, these methods mainly focus on preserving the data structure of the majority class, and ignore the fact that the boundary points have a greater impact on the construction of the classification model.

2.3. Ensemble Learning in Undersampling

The use of ensemble learning can improve the results of classification algorithms, especially with regard to undersampling. Because the undersampling method only selects part of the samples from the majority class, the information of those unselected samples is lost. Due to its ability to better utilize sample information, the undersampling classification method with ensemble learning has gained popularity [24], whereby the bagging and boosting capabilities of ensemble learning are used in undersampling; for example, the SMOTEBagging function of the ensemble classifier can be utilized [25]. The ensemble of the α-Trees framework (EAT) uses underbagging technology to achieve good results when applied to imbalanced classification problems [26]. In terms of boosting, the technologies commonly used include RUSBoost [27], SMOTEBoost [28], CSBBoost [29], Adaboost, and AsymBoost [30]. In addition, Yang et al. [31] employed progressive density-based weighted ensemble learning, and Ren et al. [32] designed a weighted integration scheme that was obtained via the use of classifiers based on the original imbalanced data set for ensemble learning.

In these ensemble learning methods, it is important that the total number of samples in the majority class is increased when the model is constructed through multiple learning; however, none of them properly consider whether the number of classifiers to be boosted can be optimized via ensemble learning.

3. Proposed Method

Since multi-class classification problems can be converted into two-class classification problems in order to obtain solutions, this paper studies the two-class imbalanced undersampling problem. Suppose that there is an imbalanced data set D that has two categories: the majority class is D_b and the minority class is D_s. We use clustering to maintain the D_b data distribution characteristics and then perform sampling to solve the problem of imbalance.

3.1. Influence of the Boundary on the Classification Results and the Idea of Our Method

When faced with a classification problem, we design a thought experiment in which there is an indefinite number of samples and sufficient samples at the boundary of different classes. These samples constitute the classification boundary. Because the samples near the classification boundary contribute more to the establishment of the classification model than those far away, for the problem of imbalanced classification, we usually select samples that are near to the boundary when undersampling is performed in the majority class.

The method commonly used to perform undersampling near the boundary is shown in Figure 1. A rough linear classification is performed to obtain $f(x_1)$. Then, $f(x_2)$, which is close to the boundary, is obtained via the parallel translation of $f(x_1)$. Finally, boundary samples are selected for the construction of a classification model via the extraction of the samples in between these two functions. According to the sample distribution shown in Figure 1, the information in the B area is lost. The ideal state is to first obtain the classification of $f(x_3)$, then obtain $f(x_4)$ through parallel translation; then, sampling is performed in the area enclosed by the two functions (Figure 2). However, $f(x_3)$ is generally difficult to obtain, and it is not easy to judge whether it is consistent with the distribution of samples in the majority class. To deal with this situation, we adopt the scheme shown in Figure 3. Firstly, a rough linear classification $f(x_1)$ is obtained. Then, clustering is performed in the majority class to obtain three classes that are parallel to $f(x_1)$: $f(x_{1a})$, $f(x_{1b})$, and $f(x_{1c})$. Finally, sampling is performed in the area enclosed by $f(x_{1a})$, $f(x_{1b})$, $f(x_{1c})$, and $f(x_1)$. In this way, a sample set that fits the classification boundary is obtained for the construction of the final classification model.

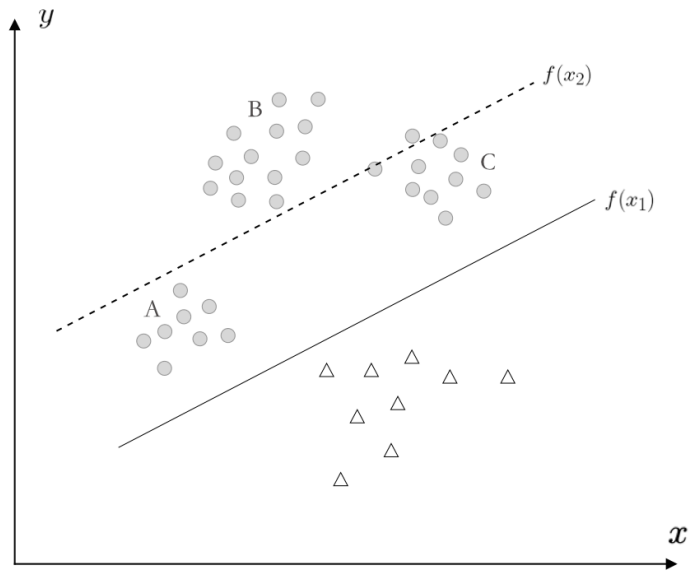


Figure 1. Method of sampling near the boundary.

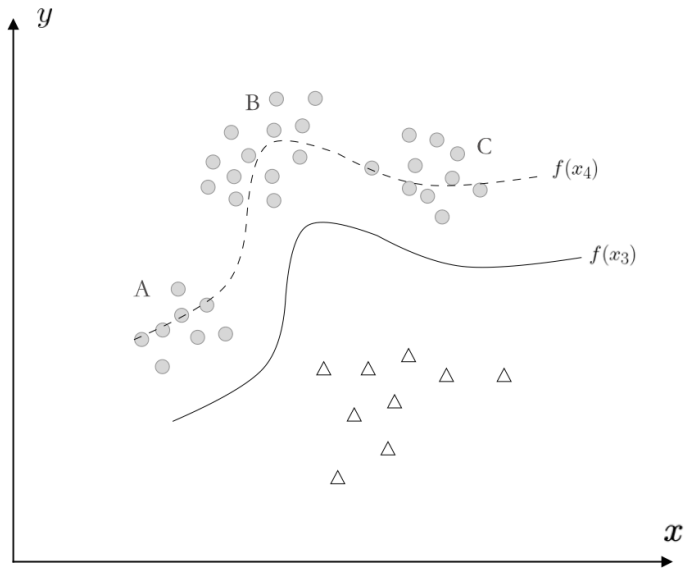


Figure 2. Method of sampling near the nonlinear boundary.

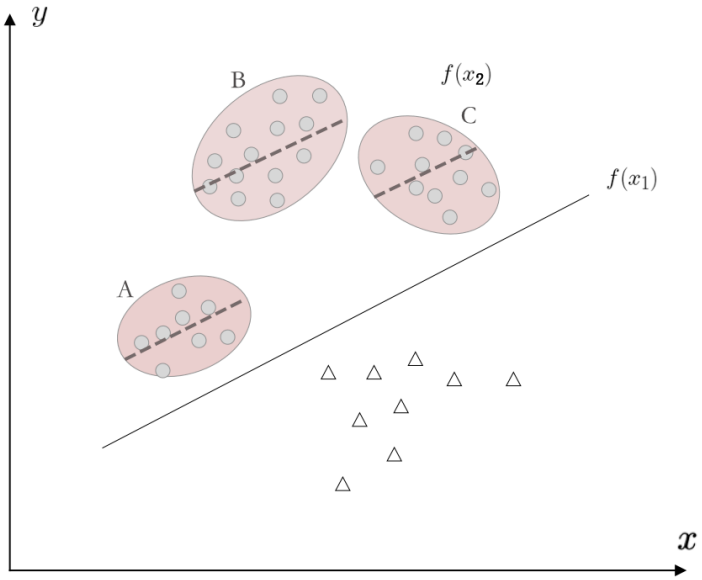


Figure 3. Method of sampling near the linear boundary with partitions.

3.2. A Cluster-Based Sampling Area Fitting the Classification Boundary Morphology and Its Undersampling

The method proposed in this paper requires performing rough division on the linear separation hyperplane prior to executing the other steps. It is therefore unclear whether any random linear method will work. Figure 4 shows that, by using $f(x_1)$-based division, the samples drawn in areas A, B, and C will most likely fall to their left; it also shows that $f(x_2)$-based division will be more uniform in the above-mentioned area. In other words, different linear division methods will have different sampling results. Therefore, it is necessary to find a more reasonable linear division method in order to determine rough boundaries.

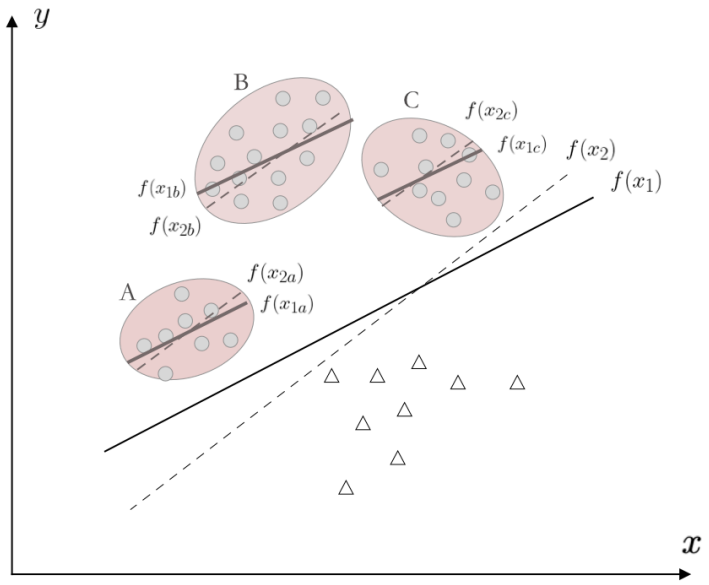


Figure 4. The influence of different linear classification boundaries.

We therefore suggest that the mean error can be used to roughly determine the appropriate linear division $f(x_i)$ in order to obtain the initial linear separation hyperplane.

$$f(x_i) = \underset{f(x_1),f(x_2),\cdots,f(x_n)}{\text{argmmin}} \Sigma_{k=1}^{|D|}(f(x^{(k)}) - y^{(k)})^2. \tag{1}$$

Herein, n is the number of the linear model $f(x_i)$, and $y^{(k)}$ is the label of sample k of D.

Once the initial partition $f(x_i)$ has been determined, the undersampling is initiated. Firstly, Kmeans clustering is performed in the majority class D_b, and the various clusters C_i are obtained. Then, according to the number of samples for D_b, N_s ($N_s = 1.5 * |D_s|$, here, D_s is the minority class), the number of samples for C_i is obtained, as follows:

$$N_{s,C_i} = \frac{|C_i|}{|D_b|} N_s. \tag{2}$$

Definition 1. *$L_{i,f(x)}$ is the diameter of the cluster facing the separating hyperplane. As shown in Figure 5, we calculate the distance L_j between each element j in C_i and $f(x_1)$; then, $L_{i,f(x)}$ is*

$$L_{i,f(x)} = L_{C_i,max} - L_{C_i,min}. \tag{3}$$

where, $L_{C_i,max}$ is the maximum distance from the samples in C_i, which is perpendicular to $f(x)$. Similarly, $L_{C_i,min}$ is the minimum distance.

If $\alpha < 1$ is the spatial compression coefficient, then the sampling space in C_i is enclosed by a hyperplane with a distance $f(x_i)$ of $L_{(C_i,min)}$ and $\alpha L_{(i,f(x))}$. The samples are obtained from a new undersampled majority class sample, namely set D_{bs}; this can then form a balanced data set with the minority class D_s for classification.

In order to minimize the distance between the sampled data objects and the boundary, we adopted a weighted method to update the samples in the majority class. That is, the closer the point is to the hyperplane, the greater the weight and the higher the probability that it will be sampled. In contrast, the farther the data object is from the hyperplane, the smaller the weight and the smaller the probability of it being sampled. Suppose $f(x_i) = \omega x_i + b$, then the weight formula is as follows:

$$k_i = (Random(0,1))^{\frac{||\omega||}{\omega x_i + b}} \tag{4}$$

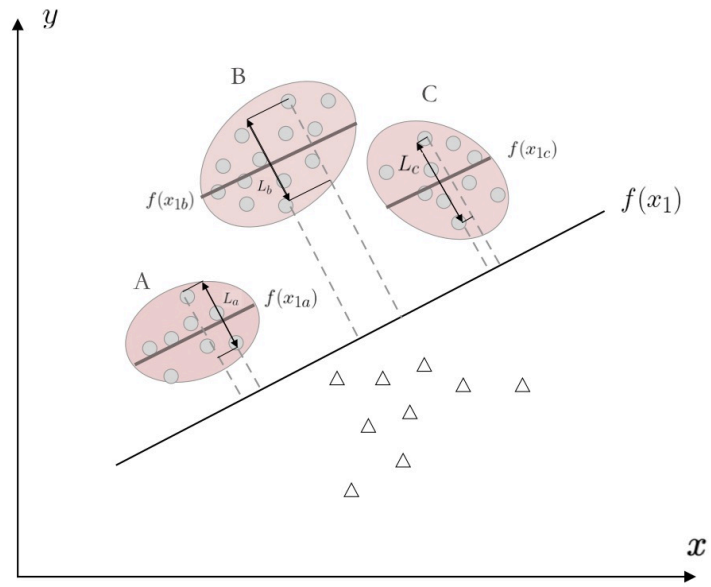


Figure 5. Diameter of the cluster facing the separating hyperplane.

In summary, the basic principle of our undersampling is as follows. Firstly, various clusters that can reflect the distribution of the majority-class samples are obtained via Kmeans clustering. Then, the spatial compression coefficient α is applied in each cluster to obtain sub-sampling spaces that are closer to the boundaries. A greater weight k_i is assigned to samples that are closer to the boundary in order to ensure that these samples are more likely to be selected. Via this approach, a relatively balanced majority class sample set D_{bu} that better reflects the shape of the classification boundary after undersampling can be obtained. The sampling method is shown in Algorithm 1.

Algorithm 1 WUC: Weighted undersampling of boundary space based on clustering

Require: Training data set D
  Majority class D_b
  The clusters of majority class $C_i, i = 0, 1, \cdots, N_c$
  Spatial compression coefficient α
Ensure: Majority class samples set obtained after undersampling D_{bu}
  Obtain hyperplane $\hat{H}$ from D by linear regression
  $D_{bu} = \{\}$
  Obtain the number of samples N_{s,C_i} for each C_i
  for $i = 1$ to N_c **do**
    for $j = 1$ to $|C_i|$ **do**
      Calculate the distance L_j from the sample x_j to the $\hat{H}$
      Use Equation (4) to calculate weight k_j
      $k_j * x_j$
    end for
    Obtain the sub-sampling space Ω_{C_i} of C_i according to the α
    Random sampling N_{s,C_i} times without replacement to obtain D_{bu,C_i}
    $D_{bu} = D_{bu} \bigcup D_{bu,C_i}$
  end for

3.3. Undersampling That Enables Close-Fitting Data Distribution at the Ideal Classification Boundary

During undersampling, a small number of samples are extracted from a large number of data objects in the majority class, thus causing some information loss. Therefore, multiple undersampling is generally used to obtain more classifiers during ensemble learning, which enhances the uniformity of the samples at the classification boundary. The questions

considered in this process are thus as follows: How can the number of classifiers m in ensemble learning be determined? Does the sample distribution that was obtained after determination conform to the ideal data distribution? Herein, one-time undersampling gains a classifier.

Definition 2. *$\phi(x, f(x_1))$ is the sample distribution of the majority class sampling, as determined by the classification boundary function. As shown in Figure 3, there is a linear separation function $f(x_1)$ between the majority class and the minority class, and C_i is obtained by clustering in the majority class. At the boundary of the majority class, $f(x_2)$ is parallel to $f(x_1)$. Then, the space formed by $f(x_2)$ and C_i facing $f(x_1)$ is the undersampled sampling space. We call the sample distribution $\phi(x, f(x_1))$.*

t is the number of classifiers in ensemble learning. When $m \to t$, the data distribution of ensemble learning $\phi(x, t) \to \phi(x, f(x_1))$. Since $f(x_1)$ is obtained via the performance rough classification, it generally does not coincide with the ideal classification $f(x_{ideal})$. As such, when $\phi(x) \to \phi(x, f(x_1))$, the classification results obtained according to this distribution $\phi(x)$ deviate. As is evident from area B in Figure 5, the sample distribution in the sampling space is biased to the left. In a situation with the ideal sampling distribution $\phi(x, f(x_{ideal}))$, where the sampling distribution is mostly uniform, this may work if there are fewer samples on the left side. That is, if m takes a value from 0 to t, the total sampling distribution of t_{opt} in the ensemble learning will approach the ideal state. When $m > t$, $\phi(x, m)$ will gradually deviate from $\phi(x, f(x_1))$, and then begin to approach $\phi(x, f(x_1))$. After analyzing the above situation, we believe that at the initial stage of ensemble learning, the result of the undersampling classification gradually becomes better; then, when it reaches m, the optimal value is obtained. After this, it gradually decreases. Based on this information, we argue that the sample distribution is uniform for $f(x_{ideal})$ when the number of undersampling is taken as m.

In order to further investigate the improvement in the algorithm according to the number of classifiers that were built using the sampling data obtained in the ensemble learning, we tested 20 data sets. For each data set, we assumed that the number of classifiers in ensemble learning ranged from 1 to 2000, and then created a graph. The graph for each data set is very similar to Figure 6. As the number of classifiers increased, the AUC of the classification result exhibited a sawtooth shape, as shown in Figure 6. Therefore, we speculate that the AUC may exhibit periodicity, accompanied by an increase in the number of classifiers in ensemble learning. We then conducted a time series periodicity test on the results of all 20 data sets after performing a Fourier transform and found that they did not show periodicity.

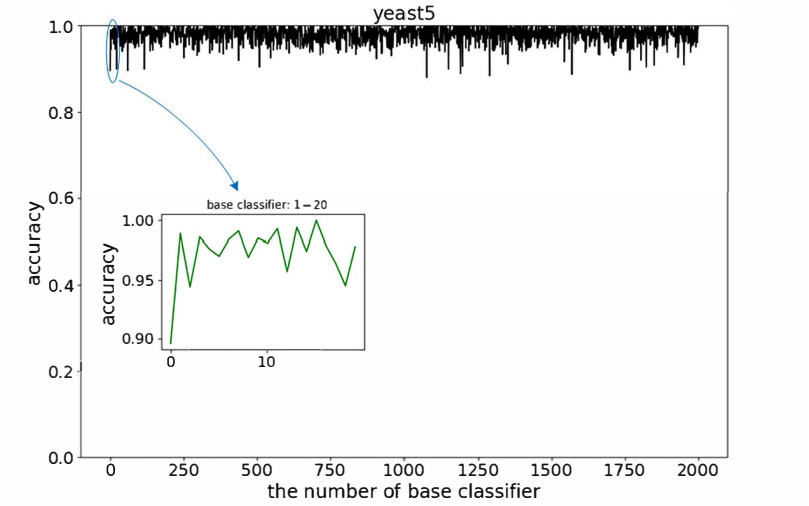


Figure 6. AUC value obtained from the *number of classifiers/undersampling times* in ensemble learning for the range 1–2000 in the *Yeast5* data set.

In this experiment, we found that each data set exhibits a cyclic phenomenon, in which the accuracy rate increases from small to large and then decreases. The sub-picture shown in Figure 6 is an enlargement of the 1–30 samples obtained from the Yeast5 data set. According to its performance, we present the following assumptions.

Hypothesis 1. *There is an m classifier that obtains the best result in a cycle of ensemble learning under $\phi(x, f(x_1))$ distribution sampling.*

According to Hypothesis 1, there exists an optimal number of classifiers that need to be integrated within a cycle, and we apply Algorithm 2 to determine this number. We set a stop threshold θ and record the result of this ensemble learning. Initially, the number of classifiers in ensemble learning ranges from 1 to a very large number. Then, we increase the number of classifiers and obtain their classification results r_i. When multiple θ consecutive r_i values that are less than the current best result r are observed, the optimal number of classifiers m is output.

Algorithm 2 DNC: Determine the number of classifiers in ensemble learning and output the best result

Require: imbalanced data set D
  stop threshold θ
  Number of ensemble learning iterations n
  Evaluation result $r = 0$
  The mode of classification with undersampling M
Ensure: Optimal times of ensemble learning m
  $Num = 0, ilabel = 0$
  for $i = 1$ to n **do**
    r_i = Adaboost(M,i) // i is the time of ensemble learning
    if $r_i > r$ **then**
      $r = r_i$
      $ilabel = i$
      $Num = 0$
      Save the result of $ilabel$
    else
      $Num++$
    end if
    if $Num > \theta$ **then**
      $m = ilabel$
      Output the result of m
      break
    end if
  end for

3.4. Our Method FDUS and Its Time Complexity

As mentioned previously, the noise reduction achieved by kNN during imbalanced classification can improve the results; we therefore utilized it to perform our preprocessing. By taking the foregoing description of this section into consideration, we named our method FDUS (**F**itting data **D**istribution with **U**nder**S**ampling). This is shown in Algorithm 3.

Algorithm 3 FDUS

Require: imbalanced data set D
Space compression factor α
Ensure: Classification model f
$D' = kNN(D)$ //Data preprocessing and denoising
Obtain $\hat{H}$ by choosing the right linear classifier
Use *Kmeans* to divide majority class
Perform WUC
Use DNC to output result

The time complexity of kNN is $O(N * N)$. We use linear SVM, linear regression, and logistic regression to optimize rough linear classification. The time complexity at this time is $3 * O(N * N)$. The time complexity of *Kmeans* is $O(n * k * t)$. Since both k and t are deterministic values and $k, t \ll N$, the time complexity of *Kmeans* approaches $O(N)$. The time complexity of assigning weights to majority-class samples is $O(N)$, and the time complexity of determining the number of classifiers in ensemble learning and outputting the optimal results is $m * O(N * N)$. Since there is no nesting between processes, the time complexity of the overall sampling algorithm is $O(N * N)$.

4. Experiments and Results

In this section, we first show the experimental details of the proposed method; these include the data sets, the comparison method, and the evaluation index. Then, we verify the aforementioned hypothesis. Finally, we analyze and compare each method based on the experimental results.

4.1. Experimental Setup

Data sets We conducted experiments on 20 imbalanced data sets obtained from **keel** (https://sci2s.ugr.es/keel/imbalanced.php (Visits since 5 November 2005)). Table 1 shows that the range in the imbalance rate (ir) ($ir = \frac{|majority-classsamples|}{|minority-classsamples|}$) of the data sets is $(1.86, 40.5)$, that the range in the examples of the data sets is $(184, 5472)$, and that the range in the dimension of the data sets is $(4, 19)$. It is evident that these data sets include both low-dimensional data and high-dimensional data, with both a low imbalance rate and a high imbalance rate; this enables us to perform a comprehensive evaluation of the proposed method. In these experiments, we used a 5-fold cross validation method to carry out the experiment and repeat it 20 times. Herein, the average of 20 experimental results is used as the final result.

Benchmark methods As mentioned in Section 2, we use five of the latest imbalanced undersampling methods and two classic algorithms; these are listed below as our benchmark methods:

(1) **NB-Tomek** [7] This method accurately identifies and eliminates overlapping instances of imbalanced data sets; therefore, the visibility of minority instances is maximized, and the excessive elimination of data is minimized, thereby reducing information loss.
(2) **USS** [14] This method calculates the sensitivity of majority-class samples. Low-sensitivity examples are considered as either noisy or safe examples, and high-sensitivity examples are regarded as critical examples. It performs undersampling in the majority-class samples via sensitivity.
(3) **RUS** [33] This method randomly selects the same number of samples from the majority class as that from the minority class, and then obtains a classifier by using AdaBoost.
(4) **RBU** [34] This method uses the Gaussian kernel function to calculate the mutual class relationship between majority-class samples and minority-class samples based on the mutual-class potential, and achieves a balance between the two classes through the diffusion kernel radius. Finally, the naive Bayes classifier is applied to achieve classification.

(5) **Centers_NN** [17] This method conducts the clustering learning of most classes in an imbalanced data set, and the nearest neighbors of the cluster center of the undersampling clustering are used for the classification learning of the imbalanced data set.
(6) **CBIS** [18] This method uses the AP algorithm to cluster the data of the majority class; it then uses the IS3 [35] method to select samples from the majority class subsets of each cluster and combines them with the minority-class samples to form a training set. Ensemble learning is then used to obtain the classifier.
(7) **UA-KF** [12] This method performs noise filtering on minority-class data, randomly undersamples majority-class data, and finally uses AdaBoost for ensemble learning.

Table 1. Data set statistics.

Data Set	Dimension	Examples	ir
abalone-17_vs_7-8-9-10	8	2338	39.31
car-vgood	6	1728	25.58
dermatology-6	34	358	16.9
ecoli1	7	336	3.36
glass-0-1-6_vs_4-5-6	9	214	3.2
glass-0-1-6_vs_5	9	184	19.44
iris0	4	150	2.0
kr-vs-k-three_vs_eleven	6	2935	35.23
led7digit-0-2-4-5-6-7-8-9_vs_1	7	443	10.97
new-thyroid1	5	215	5.14
page-blocks0	10	5472	8.79
segment0	19	2308	6.02
shuttle-c0-vs-c4	9	1829	13.87
vehicle0	18	846	3.25
vehicle2	18	846	2.88
vowel0	13	988	9.98
winequality-white-3_vs_7	11	900	44
wisconsin	9	683	1.86
yeast-0-5-6-7-9_vs_4	8	528	9.35
yeast5	8	1484	32.73

Parameters and Metrics In our proposed method, we set $k = 15$ in the kNN filtering step, and use five CART trees as weak classifiers in the AdaBoost step. It is believed that if the ratio of majority-class samples to minority-class samples exceeds 1.5, then the data are imbalanced; therefore, we set the ratio of the sampled data to 1.5 (majority class vs. minority class).

In an imbalanced data set, the accuracy of classification is biased to the data of the majority class, and therefore the results are not considered to be accurate. In this work, we use the F-measure, AUC, and G-mean, which are commonly used in research on imbalance in order to evaluate the experimental performance [36–38]. Herein, the AUC can be obtained by calculating the area of the ROC, which is a curve showing the relationship between the *False Positive Rate* and *True Positive Rate*. The formula is as follows.

$$False\ Positive\ Rate = \frac{FP}{FP + TN} \tag{5}$$

$$True\ Positive\ Rate = \frac{TP}{TP + FN} \tag{6}$$

$$False\ Negative\ Rate = \frac{TN}{TN + FP} \tag{7}$$

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

$$F\text{-}measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{10}$$

$$G\text{-}mean = \sqrt{Precision \times Recall} \tag{11}$$

4.2. Hypothesis Test to Determine If There Is an Optimal Value in a Cycle of Ensemble Learning

The results of the ensemble learning performed by the method proposed in this paper are presented. The number of classifiers is set to a range of 1 to 2000. We then select data in the first cycle of the AUC that are similar to those presented in Figure 6.

The selected data generally remove the first data, and then the remaining values are used to perform a normal distribution hypothesis test. The detection results of the 20 data sets used in this paper are recorded in Table 2. **stat** is a statistical measure that is used to measure the degree of fit between the sample data and the normal distribution, and **P** is a probability value; when P is less than the significance level of 0.05, it is considered to not follow a normal distribution. As shown in Table 2, the p values are greater than 0.05%, indicating that the confidence level is 95%. This shows that all the data have passed the hypothesis test, and therefore follow a normal distribution. Therefore, they all hold an optimal value in a cycle.

Table 2. Normal distribution test of the data sets.

Data Set	Stat	p	Number of Classifiers
abalone-17_vs_7-8-9-10	0.954	0.766	0–30
car-vgood	0.795	0.074	1–10
dermatology-6	1	1	1–10
ecoli1	0.949	0.156	0–30
glass-0-1-6_vs_4-5-6	0.964	0.636	0–20
glass-0-1-6_vs_5	0.938	0.081	0–30
iris0	1	1	0–5
kr-vs-k-three_vs_eleven	0.917	0.508	0–5
led7digit-0-2-4-5-6-7-8-9_vs_1	0.929	0.442	0–10
new-thyroid1	0.835	0.05	1–10
page-blocks0	0.776	0.051	0–5
segment0	0.858	0.222	0–5
shuttle-c0-vs-c4	1	1	0–30
vehicle0	0.959	0.315	1–30
vehicle2	0.904	0.429	1–10
vowel0	0.872	0.273	0–5
winequality-white-3_vs_7	0.959	0.284	0–30
wisconsin	0.806	0.091	0–56
yeast-0-5-6-7-9_vs_4	8	528	9.35
yeast5	0.847	0.186	0–5

4.3. Effect of Different Clustering Methods on Research Results

In order to discuss the impact of clustering algorithms for large-class partitioning on the entire method, we selected two classic methods, namely DBSCAN and SVR, and compared them with Kmeans. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [39] is a data-clustering algorithm that is used to discover clusters of arbitrary shapes and is based on the concept of density reachability and density connectivity. Support Vector Clustering (SVC) [40] is used to cluster data points in a high-dimensional space. From Table 3, it can be seen that the results of the three methods are very similar, with Kmeans performing slightly better than DBSCAN and SVC. According to the principle of Occam's razor, we chose to use Kmeans in our method.

Table 3. Different ratios of AUC, F-measure ('F' for short), and G-mean ('G' for short) with different clustering methods.

Data Set	Kmeans			DBSCAN			SVC		
	AUC	F	G	AUC	F	G	AUC	F	G
abalone-17_vs_7-8-9-10	0.8823	0.7782	0.7463	0.8812	0.7778	0.7447	0.8792	0.7986	0.7612
car-vgood	0.9739	0.7935	0.7877	0.9750	0.7871	0.7910	0.9648	0.7861	0.7816
dermatology-6	1	1	0.9817	0.9978	0.9993	0.9958	1	1	0.9813
ecoli1	0.9308	0.8801	0.8526	0.9285	0.8619	0.8357	0.9291	0.8689	0.8435
glass-0-1-6_vs_4-5-6	0.9958	0.8923	0.8595	0.9961	0.8982	0.8601	0.9912	0.8829	0.8546
glass-0-1-6_vs_5	1	1	1	1	1	1	1	1	1
iris0	0.9961	0.9895	0.9897	0.9836	0.9865	0.9867	0.9924	0.9876	0.9877
kr-vs-k-three_vs _eleven	0.9972	0.9618	0.9657	0.9981	0.9633	0.9672	0.9961	0.9588	0.9562
led7digit-0-2-4-5-6-7-8-9_vs_1	0.9869	0.8633	0.8431	0.9910	0.8639	0.8482	0.9923	0.8702	0.8682
new-thyroid1	0.9875	0.9867	0.9871	0.9855	0.9843	0.9847	0.9817	0.9831	0.9803
page-blocks0	0.9659	0.9051	0.8974	0.9583	0.8979	0.8928	0.9706	0.9033	0.9011
segment0	0.9993	0.9874	0.9879	0.9990	0.9862	0.9865	0.9993	0.9874	0.9879
shuttle-c0-vs-c4	1	1	1	1	1	1	1	1	1
vehicle0	0.9777	0.9455	0.9367	0.9815	0.9465	0.9386	0.9818	0.9466	0.9391
vehicle2	0.9868	0.9162	0.9042	0.9837	0.9108	0.9027	0.9842	0.9125	0.9034
vowel0	0.9857	0.8989	0.9064	0.9816	0.8959	0.9036	0.9861	0.8994	0.9066
winequality-white-3_vs_7	0.9600	0.9333	0.9082	0.9612	0.9345	0.9095	0.9588	0.9311	0.9061
wisconsin	0.9794	0.9183	0.8974	0.9805	0.9201	0.8996	0.9748	0.9166	0.8967
yeast-0-5-6-7-9_vs_4	0.8484	0.7385	0.7172	0.8429	0.7331	0.7138	0.8490	0.7392	0.7181
yeast5	0.9862	0.9581	0.9534	0.9836	0.9569	0.9514	0.9878	0.9518	0.9515

4.4. Effect of Proper Separation of the Hyperplane on Research Results

We suggest that the separating hyperplane chosen will affect the final result of the algorithm. Therefore, our FDUS method selects an optimal H_{opt} from among the separated hyperplanes H obtained by the SVM, linear classification, and logistic classification before proceeding with the other steps. To verify this idea and prove that the optimization of H can assist the algorithm, we apply the H that was obtained using the above three methods directly to the FDUS method in order to perform experiments. Each experiment was performed 20 times, and then their average results were entered in Table 4.

Table 4. Different ratios of AUC, F-measure ('F' for short), and G-mean ('G' for short) with different separated hyperplanes.

Data Set	SVM			Linear Classification			Logistic Regression		
	AUC	F	G	AUC	F	G	AUC	F	G
abalone-17_vs_7-8-9-10	0.8823	0.7782	0.7463	0.8691	0.7757	0.7643	0.8445	0.8391	0.8381
car-vgood	0.9739	0.7935	0.7877	0.9813	0.7827	0.8001	0.9520	0.8669	0.8803
dermatology-6	1	1	0.9817	0.9659	0.8961	0.9056	0.9750	0.9333	0.9333
ecoli1	0.9308	0.8801	0.8526	0.8631	0.8288	0.7903	0.9531	0.8387	0.7983
glass-0-1-6_vs_4-5-6	0.9958	0.8923	0.8595	0.9300	0.8789	0.8437	0.9482	0.8729	0.8144
glass-0-1-6_vs_5	1	1	1	0.8667	0.8667	0.8309	0.8500	0.8333	0.8309
iris0	0.9961	0.9895	0.9897	0.9895	0.9895	0.9897	0.9900	0.9895	0.9897
kr-vs-k-three_vs _eleven	0.9972	0.9618	0.9657	0.9261	0.9050	0.9050	0.9625	0.9018	0.9090
led7digit-0-2-4-5-6-7-8-9_vs_1	0.9869	0.8633	0.8431	0.8783	0.8189	0.8182	0.8581	0.8081	0.8054
new-thyroid1	0.9875	0.9867	0.9871	0.9625	0.9867	0.9871	0.9607	0.9581	0.9603
page-blocks0	0.9659	0.9051	0.8974	0.8974	0.8959	0.8930	0.9256	0.8932	0.8906
segment0	0.9993	0.9874	0.9879	0.9702	0.9619	0.9618	0.9879	0.9866	0.9878
shuttle-c0-vs-c4	1	1	1	0.9960	0.9961	0.9497	0.9560	0.9639	0.9692
vehicle0	0.9777	0.9455	0.9367	0.9316	0.9207	0.9086	0.9150	0.9190	0.9066
vehicle2	0.9868	0.9162	0.9042	0.8794	0.8790	0.8714	0.9533	0.9165	0.9098
vowel0	0.9857	0.8989	0.9064	0.9007	0.8980	0.9017	0.9121	0.8677	0.8828
winequality-white-3_vs_7	0.9600	0.9333	0.9082	0.8900	0.7968	0.8127	0.9000	0.8667	0.8828
wisconsin	0.9794	0.9183	0.8974	0.8725	0.8712	0.8695	0.9647	0.8782	0.8617
yeast-0-5-6-7-9_vs_4	0.8484	0.7385	0.7172	0.7158	0.7146	0.7134	0.7338	0.7103	0.6832
yeast5	0.9862	0.9581	0.9534	0.9533	0.9429	0.9433	0.9922	0.9367	0.9373

It can be seen from Table 4 that the final results achieved by the different separation hyperplanes obtained by the SVM, linear classification, and logistic classification also have their own advantages and disadvantages. This shows that the FDUS algorithm is closely related to the initial H. Secondly, it can be seen from Table 3 that, according to an evaluation of the data using the AUC, the final result of the FDUS method based on linear classification and the *car-vgood* data set is the best; it can also be observed that the method based on logistic regression classification achieves the best results for the two data sets of *ecoli1* and *yeast5*, and that the SVM-based algorithm shows the best results for the remaining data sets. This proves that the optimization of the separated hyperplane can improve the results of the algorithm.

4.5. Comparison of FDUS with Benchmarks

In this section, the proposed FDUS is compared with seven representative algorithms on 20 numerical data sets. Tables 5–7 present the evaluations of the AUC, F-measure, and G-mean. Each of their results is obtained from the average of 20 operations. We will compare and analyze these results from four aspects, while examining the performance of FDUS.

Table 5. AUC results using various comparison methods. Abbreviations: Tomek, NB-Tomek; Centers, Centers_nn.

Data Set	FDUS	Tomek	USS	RUS	RBU	Centers	CBIS	UA-KF
abalone-17_vs_7-8-9-10	0.8823	0.8758	0.8123	0.8142	0.8183	0.8554	0.8592	0.8446
car-vgood	0.9813	0.9642	0.6241	0.9118	0.9446	0.9678	0.9757	0.9446
dermatology-6	1	0.9849	0.9588	0.9575	0.9985	0.9903	0.9875	0.9985
ecoli1	0.9531	0.9211	0.8529	0.8439	0.9038	0.8975	0.9186	0.9141
glass-0-1-6_vs_4-5-6	0.9958	0.9803	0.7808	0.9785	0.9914	0.9372	0.9547	0.9914
glass-0-1-6_vs_5	1	0.8985	0.8502	0.8262	0.8779	0.8612	0.8922	0.8528
iris0	0.9961	0.9978	0.9901	0.9878	0.9754	0.9979	0.9824	0.9489
kr-vs-k-three_vs _eleven	0.9972	0.9953	0.7693	0.9863	0.9881	0.9860	0.9562	0.9881
led7digit-0-2-4-5-6-7-8-9_vs_1	0.9869	0.9029	0.6875	0.9841	0.9862	0.9355	0.9556	0.9862
new-thyroid1	0.9875	0.9851	0.9486	0.9522	0.9567	0.9554	0.9741	0.9284
page-blocks0	0.9659	0.9007	0.8396	0.8276	0.9132	0.8932	0.9607	0.8818
segment0	0.9993	0.9905	0.9167	0.9064	0.9756	0.9818	0.9815	0.9029
shuttle-c0-vs-c4	1	0.9997	0.8757	0.9199	0.9846	0.9978	0.9464	0.9334
vehicle0	0.9777	0.9457	0.8742	0.8866	0.9437	0.9038	0.9567	0.9296
vehicle2	0.9868	0.9598	0.6504	0.9686	0.9738	0.9187	0.9715	0.9738
vowel0	0.9857	0.9719	0.8405	0.8279	0.9401	0.9557	0.9782	0.9518
winequality-white-3_vs_7	0.9600	0.8058	0.7587	0.7845	0.7968	0.7794	0.7968	0.7903
wisconsin	0.9794	0.9686	0.8938	0.9092	0.9429	0.9753	0.9532	0.9063
yeast-0-5-6-7-9_vs_4	0.8484	0.8049	0.7884	0.8142	0.8315	0.8295	0.8103	0.7935
yeast5	0.9922	0.9852	0.8978	0.8278	0.9651	0.9552	0.9773	0.8661
Average	0.9738	0.9419	0.8305	0.8958	0.9354	0.9287	0.9394	0.9164

Cases with complex sample distribution. Here, FDUS exhibits significant advantages compared to NB-tomek and RBU. In terms of the average results obtained in the AUC, F-measure and G-mean evaluations, FDUS exceeds NB-Tomek by 3.18%, 8.50%, and 6.58%, respectively. It also exceeds RBU by 3.84%, 4.47%, and 4.31%, respectively. Compared to NB-Tomek, FDUS exhibits good priority in each data set. The number of data sets for which FDUS is superior to RBU is 20, 19, and 18 with respect to the AUC, F-measure, and G-mean, respectively. NB-Tomek is mainly designed for the complex situation in which majority-class samples overlap with minority-class samples, while RBU uses Gaussian kernel functions to deal with the complex distribution of samples in the majority-class sample space. We argue that FDUS alleviates the problem of sample overlap by undersampling in each partition after clustering in the majority class. At the same time, undersampling in the determined majority-class sample space enables some of the minor errors introduced by the NB-Tomek method to be avoided. Therefore, FDUS is superior to NB-Tomek. For RBU, some of the sample distribution features that are captured by Gaussian kernel functions are far from the classification boundary, while those captured by FDUS are closer. Therefore, FDUS exhibits better performance after undersampling.

Table 6. F-measure results using various comparison methods. Abbreviations: Tomek, NB-Tomek; Centers, Centers_nn.

Data Set	FDUS	Tomek	USS	RUS	RBU	Centers	CBIS	UA-KF
abalone-17_vs_7-8-9-10	0.7782	0.7547	0.7066	0.7169	0.7033	0.7164	0.7423	0.7244
car-vgood	0.7827	0.2727	0.7093	0.9064	0.8722	0.8063	0.7783	0.8722
dermatology-6	1	0.9493	0.9354	0.9382	0.9199	0.9732	0.9627	0.9199
ecoli1	0.8387	0.8785	0.8098	0.8375	0.8812	0.8551	0.8723	0.8644
glass-0-1-6_vs_4-5-6	0.8923	0.8667	0.7700	0.9318	0.8582	0.7991	0.9442	0.8582
glass-0-1-6_vs_5	1	0.9177	0.8063	0.8214	0.8619	0.9197	0.8947	0.8385
iris0	0.9895	0.9707	0.9434	0.9477	0.9042	0.9675	0.9228	0.9375
kr-vs-k-three_vs_eleven	0.9618	0.7799	0.7948	0.9629	0.8991	0.9348	0.9517	0.8991
led7digit-0-2-4-5-6-7-8-9_vs_1	0.8633	0.6654	0.7297	0.9346	0.7652	0.8607	0.8625	0.7652
new-thyroid1	0.9867	0.9195	0.8494	0.9491	0.9323	0.9543	0.9631	0.8564
page-blocks0	0.9051	0.8924	0.8741	0.8113	0.8901	0.8873	0.9085	0.8473
segment0	0.9874	0.8930	0.9524	0.9059	0.9581	0.9386	0.9534	0.8934
shuttle-c0-vs-c4	0.9633	0.8902	0.8574	0.8752	0.9584	0.9459	0.9259	0.9162
vehicle0	0.9455	0.9006	0.8112	0.8262	0.8928	0.8612	0.9372	0.9059
vehicle2	0.9162	0.8945	0.5942	0.9296	0.8672	0.8452	0.9414	0.8672
vowel0	0.8989	0.9004	0.8711	0.7973	0.8968	0.8697	0.8958	0.8427
winequality-white-3_vs_7	0.9333	0.7697	0.7275	0.7229	0.7574	0.7178	0.7704	0.7202
wisconsin	0.9183	0.8792	0.8706	0.8013	0.8821	0.8257	0.8769	0.7809
yeast-0-5-6-7-9_vs_4	0.7385	0.7169	0.7087	0.6943	0.7057	0.7274	0.7143	0.7062
yeast5	0.9367	0.8238	0.8727	0.8078	0.9365	0.9133	0.9193	0.8247
Average	0.9118	0.8268	0.8097	0.8559	0.8671	0.8660	0.8869	0.8420

The case in which the sample features sensitivity and noise. We selected USS and UA-KF to conduct comparisons. Across each data set, FDUS significantly outperforms UA-KF when evaluated using the AUC metric. In the evaluation of the F-measure and G-mean, FDUS outperforms UA-KF on 18 and 16 data sets, respectively, demonstrating its superior performance. Compared with UA-KF, FDUS is 5.74% superior to the AUC, 6.98% superior to F-measure, and 7.31% superior to G-mean with regard to the average of the evaluation indicators for all data sets. For USS, in terms of the average values for the AUC, F-measure, and G-mean evaluations, FDUS exhibits significant priority in each data set, with values of 14.32%, 10.21%, and 12.93%, respectively. Due to the sensitivity of USS for all samples in a majority class, samples near the boundary are also included in the calculation. This may lead to some important samples in the boundary region being incorrectly processed due to their high sensitivity. UA-KF, on the other hand, only employs KNN to address noise issues in minority-class samples. In contrast, FDUS eliminates the influence of outliers far from the classification boundary through spatial compression throughout the entire KNN data preprocessing process. This enables FDUS to perform well in terms of noise and sensitivity.

Ensemble learning and undersampling across the majority class. In order to compare FDUS with ensemble learning methods that do not partition in the majority class, we chose RUS. Regarding the average results of the AUC, F-measure, and G-mean evaluations, FDUS performs exceptionally well, with 7.80%, 5.59%, and 6.37% higher values compared to RUS. Meanwhile, 20, 15, and 16 data sets dominate, respectively. Because random undersampling is performed directly on the majority class without partitioning, the balanced majority class obtained may not necessarily match the original sample distribution well, which could lead to a worse score than that obtained with FDUS.

Comparison with cluster-based undersampling methods. We chose Centers_NN and CBIS for a comparison of their performance. Regarding the average results of the AUC, F-measure, and G-mean evaluations, FDUS exceeds Centers_NN by 4.50%, 4.58%, and 4.60%, respectively. It also exceeds CBIS by 3.43%, 2.49%, and 3.23%, respectively. The number of data sets in which FDUS outperforms Centers_NN is 19, 18, and 19, respectively, and the number of data sets in which FDUS surpasses CBIS is 20, 16, and 15, respectively. The Centers_NN and CBIS methods use the NN and AP algorithms, respectively, to cluster the majority class and obtain the sample distribution characteristics. However, during undersampling, they operate within the entire class sample space. As a result, some samples that are distant from the classification boundary are retained; their number is

set to m. Furthermore, because of these m samples, they have m fewer samples near the boundary compared to FDUS. Consequently, FDUS achieves a closer approximation to the ideal classification boundary than the boundaries generated by these methods. Hence, FDUS demonstrates superior performance.

Table 7. G-mean results using various comparison methods. Abbreviations: Tomek, NB-Tomek; Centers, Centers_nn.

Data Set	FDUS	Tomek	USS	RUS	RBU	Centers	CBIS	UA-KF
abalone-17_vs_7-8-9-10	0.7463	0.7193	0.6571	0.7087	0.7091	0.6939	0.7347	0.7047
car-vgood	0.8001	0.3788	0.4310	0.8912	0.8634	0.7716	0.7812	0.7156
dermatology-6	1	0.9713	0.9702	0.9021	0.9013	0.9675	0.9321	0.9693
ecoli1	0.7983	0.8396	0.7393	0.8312	0.8387	0.8279	0.8633	0.8083
glass-0-1-6_vs_4-5-6	0.8595	0.8845	0.7536	0.8762	0.8186	0.7977	0.9116	0.8882
glass-0-1-6_vs_5	1	0.8294	0.7706	0.8121	0.8438	0.8559	0.8473	0.8173
iris0	0.9897	0.9425	0.9369	0.9418	0.9033	0.9566	0.9159	0.9159
kr-vs-k-three_vs_eleven	0.9657	0.8799	0.6561	0.9191	0.9012	0.9411	0.8991	0.9631
led7digit-0-2-4-5-6-7-8-9_vs_1	0.8431	0.8263	0.6445	0.8661	0.7516	0.8390	0.8659	0.9001
new-thyroid1	0.9871	0.9255	0.8466	0.9163	0.9503	0.9431	0.9497	0.8097
page-blocks0	0.8974	0.8145	0.8657	0.8081	0.8985	0.8795	0.9002	0.8146
segment0	0.9879	0.9288	0.8583	0.9015	0.9427	0.9405	0.9589	0.8889
shuttle-c0-vs-c4	0.9617	0.9047	0.8619	0.8432	0.9501	0.9572	0.9144	0.8444
vehicle0	0.9367	0.9286	0.8101	0.8334	0.8987	0.9068	0.9278	0.8778
vehicle2	0.9042	0.8978	0.6048	0.8971	0.8421	0.8525	0.9401	0.9073
vowel0	0.9064	0.8822	0.8684	0.7561	0.8872	0.8777	0.8728	0.8028
winequality-white-3_vs_7	0.9082	0.7208	0.7379	0.7427	0.7686	0.6905	0.7462	0.6962
wisconsin	0.8974	0.8712	0.8681	0.8341	0.8762	0.8248	0.8208	0.7508
yeast-0-5-6-7-9_vs_4	0.7172	0.7014	0.6801	0.7055	0.7126	0.6898	0.7074	0.7002
yeast5	0.9373	0.8815	0.8976	0.7839	0.9242	0.9101	0.9076	0.8076
Average	0.9022	0.8364	0.7729	0.8385	0.8591	0.8562	0.8699	0.8291

Based on the above analysis of the four scenarios, FDUS offers the following advantages when evaluated against the three indicators. Firstly, it effectively handles data contaminated by noise and outliers. Secondly, it successfully captures the features of majority-class samples with complex distributions, leading to a model that more closely aligns with the ideal classification boundary. Thirdly, as demonstrated by the data sets used in the experiment, FDUS is capable of handling highly imbalanced data sets.

Stability Comparison of Algorithms

We used 20 results pertaining to each algorithm to draw box plots and examine the stability of these algorithms. As is evident in Figure 7, the FDUS method proposed in this paper is the most stable of all the algorithms. In the box plots of the 20 data sets, the area in which the FDUS results are scattered is very small, and almost all of them form a straight line; this shows that the stability of the algorithm is excellent. Other algorithms, however, mostly show the characteristics of larger scattered areas in these data sets; this is because these algorithms all use random sampling methods that lead to unstable results. Although our method also uses random sampling, the result of each experiment remains immensely stable. This shows that an optimal value for FDUS exists, and that the algorithm can find the optimal value or the approximate optimal value. This phenomenon coincides with the assumption that an optimal value exists in FDUS ensemble learning, which we have proved by testing our hypothesis.

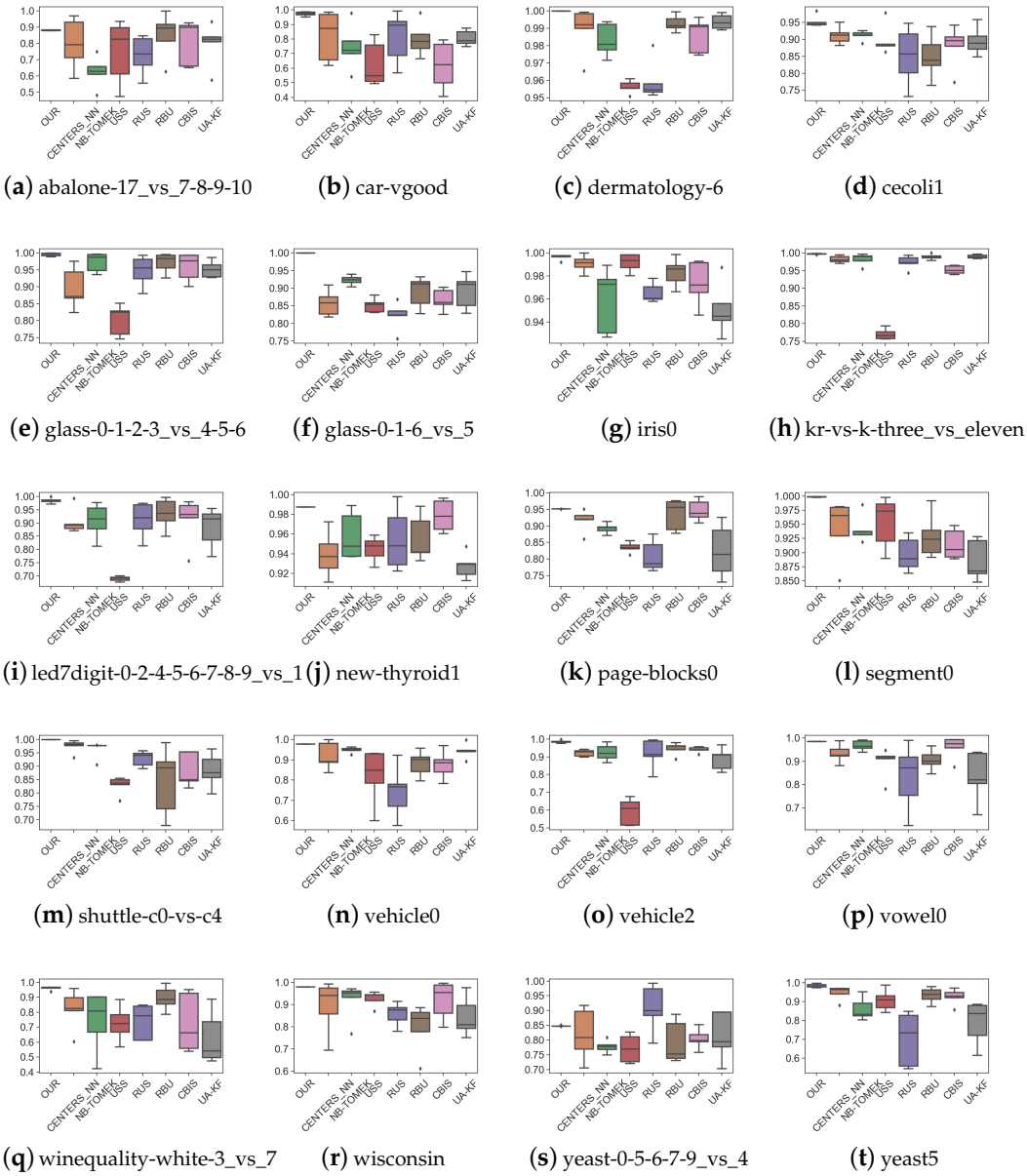


Figure 7. Box plots with AUC.

5. Conclusions

The imbalanced classification method based on clustering is first used to perform cluster analysis and then perform a random sampling in each cluster to obtain a balanced data set. This traditional approach enables the characteristics of the original data set distribution to be obtained; however, it also retains the data with a low contribution to the classification and thus reduces the accuracy of the classification model. To solve this problem, we propose the FDUS method. It firstly runs the clustering algorithm in the majority class, and then performs a random sampling in the borderline area near to the minority class in the clusters just obtained. In this way, the classification boundary obtained with FDUS conforms to the original shape of the data set between the two classes. In order to enhance the consistency between the classification boundary and the ideal shape, we adopted an ensemble learning method that utilizes multiple classifiers to improve the algorithm. According to the experimental analysis performed on the use of different numbers of classifiers for ensemble learning in FDUS, we put forward the hypothesis that there is an optimal number of models to be in ensemble learning, and proved it via the

performance of a hypothesis test. Having been validated by 20 data sets and compared with seven baseline methods, FDUS conclusively and consistently exhibits superior performance.

The FDUS method also, however, exhibits limitations, as it can only be applied to two-class classification problems. Meanwhile, the problem of imbalance exists in different fields, which may require the handling of multimodal data. In the future, we will extend our research to the evaluation of multi-class classification problems and multimodal data.

Author Contributions: Investigation, C.L.; Writing—original draft, W.Z. and P.Y.; Writing—review and editing, C.L. and L.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Open Fund Project of the National Key Laboratory of Offshore Oil and Gas Development.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

Acknowledgments: The authors would like to express their gratitude to Zenghua Zhang, a technical expert at the National Key Laboratory of Offshore Oil and Gas Exploration, for providing guidance on this paper.

Conflicts of Interest: Authors Wensheng Zhou and Chen Liu were employed by the company CNOOC Research Institute Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Salah Al-Deen, S.; Castillo, P.A.; Faris, H. Cost-sensitive metaheuristic optimization-based neural network with ensemble learning for financial distress prediction. *Appl. Sci.* **2022**, *12*, 6918. [CrossRef]
2. Alruily, M.; El-Ghany, S.A.; Mostafa, A.M.; Ezz, M.; El-Aziz, A.A. A-tuning ensemble machine learning technique for cerebral stroke prediction. *Appl. Sci.* **2023**, *13*, 5047. [CrossRef]
3. Han, S.; Zhu, K.; Zhou, M.; Cai, X. Competition-driven multimodal multiobjective optimization and its application to feature selection for credit card fraud detection. *IEEE Trans. Syst. Man Cybern. Syst.* **2022**, *52*, 7845–7857. [CrossRef]
4. Liu, Y.; Zhou, J.; Zhang, D.; Wei, S.; Yang, M.; Gao, X. Fault Diagnosis Method of Box-Type Substation Based on Improved Conditional Tabular Generative Adversarial Network and AlexNet. *Appl. Sci.* **2024**, *14*, 3112. [CrossRef]
5. Yan, Y.; Xu, Y.; Xue, J.H.; Lu, Y.; Wang, H.; Zhu, W. Drop loss for person attribute recognition with imbalanced noisy-labeled samples. *IEEE Trans. Cybern.* **2023**, *53*, 7071–7084. [CrossRef] [PubMed]
6. Li, T., Wang, Y., Liu, L., Chen, L.P. Subspace-based minority oversampling for imbalance classification. *Inf. Sci.* **2023**, *621*, 371–388. [CrossRef]
7. Vuttipittayamongkol, P.; Elyan, E. Neighbourhood-based undersampling approach for handling imbalanced and overlapped data. *Inf. Sci.* **2020**, *509*, 47–70. [CrossRef]
8. Farshidvard, A.; Hooshm, F.; MirHassani, S.A. A novel two-phase clustering-based under-sampling method for imbalanced classification problems. *Expert Syst. Appl.* **2023**, *213*, 119003. [CrossRef]
9. Sun, H.; Tian, C.; Xiao, J.; Yang, Y. Learn Stable MRI Under-sampling Pattern with Decoupled Sampling Preference. *IEEE Trans. Comput. Imaging* **2024**, *10*, 246–260. [CrossRef]
10. Van Hulse, J.; Khoshgoftaar, T.M.; Napolitano, A. A novel noise filtering algorithm for imbalanced data. In Proceedings of the Ninth International Conference on Machine Learning and Applications, Washington, DC, USA, 12–14 December 2010. [CrossRef]
11. Sáez, J.A.; Luengo, J.; Stefanowski, J.; Herrera, F. SMOTE–IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. *Inf. Sci.* **2015**, *291*, 184–203. [CrossRef]
12. Kang, Q.; Chen, X.; Li, S.; Zhou, M. A noise-filtered under-sampling scheme for imbalanced classification. *IEEE Trans. Cybern.* **2016**, *47*, 4263–4274. [CrossRef] [PubMed]
13. Dixit, A.; Mani, A. Sampling technique for noisy and borderline examples problem in imbalanced classification. *Appl. Soft Comput.* **2023**, *142*, 110361. [CrossRef]
14. Zhang, J.; Wang, T.; Ng, W.W.; Zhang, S.; Nugent, C.D. Undersampling near decision boundary for imbalance problems. In Proceedings of the International Conference on Machine Learning and Cybernetics (ICMLC), Kobe, Japan, 7–10 July 2019. [CrossRef]
15. Hoyos-Osorio, J.; Alvarez-Meza, A.; Daza-Santacoloma, G.; Orozco-Gutierrez, A.; Castellanos-Dominguez, G. Relevant information undersampling to support imbalanced data classification. *Neurocomputing* **2021**, *436*, 136–146. [CrossRef]

16. Yen, S.-J.; Lee, Y.-S. Cluster-based under-sampling approaches for imbalanced data distributions. *Expert Syst. Appl.* **2009**, *36*, 5718–5727. [CrossRef]
17. Lin, W.C.; Tsai, C.F.; Hu, Y.H.; Jhang, J.S. Clustering-based undersampling in class-imbalanced data. *Inf. Sci.* **2017**, *409*, 17–26. [CrossRef]
18. Tsai, C.F.; Lin, W.C.; Hu, Y.H.; Yao, G.T. Under-sampling class imbalanced datasets by combining clustering analysis and instance selection. *Inf. Sci.* **2019**, *477*, 47–54. [CrossRef]
19. Li, J.; Wang, H.; Song, C.; Han, R.; Hu, T. Research on Hierarchical Clustering Undersampling and Random Forest Fusion Classification Method. In Proceedings of the 2021 IEEE International Conference on Progress in Informatics and Computing (PIC), Shanghai, China, 17–19 December 2021; pp. 53–57.
20. Jang, J.; Kim, C.O. Unstructured borderline self-organizing map: Learning highly imbalanced, high-dimensional datasets for fault detection. *Expert Syst. Appl.* **2022**, *188*, 116028. [CrossRef]
21. Devi, D.; Namasudra, S.; Kadry, S. A boosting-aided adaptive cluster-based undersampling approach for treatment of class imbalance problem. *Int. J. Data Warehous. Min. (IJDWM)* **2020**, *16*, 60–86. [CrossRef]
22. Guzmán-Ponce, A.; Sánchez, J.S.; Valdovinos, R.M.; Marcial-Romero, J.R. DBIG-US: A two-stage under-sampling algorithm to face the class imbalance problem. *Expert Syst. Appl.* **2021**, *168*, 114301. [CrossRef]
23. Tahfim, S.A.S.; Chen, Y. Comparison of Cluster-Based Sampling Approaches for Imbalanced Data of Crashes Involving Large Trucks. *Information* **2024**, *15*, 145. [CrossRef]
24. Bai, L.; Ju, T.; Wang, H.; Lei, M.; Pan, X. Two-step ensemble under-sampling algorithm for massive imbalanced data classification. *Inf. Sci.* **2024**, *665*, 120351. [CrossRef]
25. Feng, W.; Huang, W.; Ren, J. Class imbalance ensemble learning based on the margin theory. *Appl. Sci.* **2018**, *8*, 815. [CrossRef]
26. Park, Y.; Ghosh, J. Ensembles of (α)-Trees for Imbalanced Classification Problems. *IEEE Trans. Knowl. Data Eng.* **2012**, *26*, 131–143. [CrossRef]
27. Kinoshita, T.; Fujiwara, K.; Kano, M.; Ogawa, K.; Sumi, Y.; Matsuo, M.; Kadotani, H. Sleep spindle detection using RUSBoost and synchrosqueezed wavelet transform. *IEEE Trans. Neural Syst. Rehabil. Eng.* **2020**, *28*, 390–398. [CrossRef] [PubMed]
28. Rajagopalan, S.; Singh, J.; Purohit, A. VMD-Based Ensembled SMOTEBoost for Imbalanced Multi-Class Rotor Mass Imbalance Fault Detection and Diagnosis Under Industrial Noise. *J. Vib. Eng. Technol.* **2024**, *12*, 1457–1478. [CrossRef]
29. Salehi, A.R.; Khedmati, M. A cluster-based SMOTE both-sampling (CSBBoost) ensemble algorithm for classifying imbalanced data. *Sci. Rep.* **2024**, *14*, 5152. [CrossRef] [PubMed]
30. Jiang, L.; Yuan, P.; Liao, J.; Zhang, Q.; Liu, J.; Li, K. Undersampling of approaching the classification boundary for imbalance problem. *Concurr. Comput. Pract. Exp.* **2023**, *35*, 1–17. . [CrossRef]
31. Yang, K.; Yu, Z.; Chen, C.P.; Cao, W.; You, J.; Wong, H.S. Incremental weighted ensemble broad learning system for imbalanced data. *IEEE Trans. Knowl. Data Eng.* **2021**, *34*, 5809–5824. [CrossRef]
32. Ren, J.; Wang, Y.; Mao, M.; Cheung, Y.M. Equalization ensemble for large scale highly imbalanced data classification. *Knowl.-Based Syst.* **2022**, *242*, 108295. [CrossRef]
33. He, H.; Garcia, E.A. Learning from imbalanced data. *IEEE Trans. Knowl. Data Eng.* **2009**, *21*, 1263–1284.
34. Koziarski, M. Radial-based undersampling for imbalanced data classification. *Pattern Recognit.* **2020**, *102*, 107262. [CrossRef]
35. Waseem, M.; Lin, Z.; Liu, S.; Jinai, Z.; Rizwan, M.; Sajjad, I.A. Optimal BRA based electric demand prediction strategy considering instance-based learning of the forecast factors. *Int. Trans. Electr. Energy Syst.* **2021**, *31*, e12967. [CrossRef]
36. Bradley, A.P. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognit.* **1997**, *30*, 1145–1159. [CrossRef]
37. Powers, D.M.W. Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. *arXiv* **2020**, arXiv:2010.16061.
38. Guo, H.; Liu, H.; Wu, C.; Zhi, W.; Xiao, Y.; She, W. Logistic discrimination based on G-mean and F-measure for imbalanced problem. *J. Intell. Fuzzy Syst.* **2016**, *31*, 1155–1166. [CrossRef]
39. Schubert, E.; Sander, J.; Ester, M.; Kriegel, H.P.; Xu, X. DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN. *ACM Trans. Database Syst. (TODS)* **2017**, *42*, 1–21. [CrossRef]
40. Ben-Hur, A.; Horn, D.; Siegelmann, H.T.; Vapnik, V. Support vector clustering. *J. Mach. Learn. Res.* **2001**, *2*, 125–137. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.