Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.048062

ARTICLE

An Imbalanced Data Classification Method Based on Hybrid Resampling and Fine Cost Sensitive Support Vector Machine

Bo Zhu*, Xiaona Jing, Lan Qiu and Runbo Li

College of Mechanical and Electrical Engineering, Kunming University of Science & Technology, Kunming, 650500, China

*Corresponding Author: Bo Zhu. Email: zhubo20110720@163.com

Received: 26 November 2023 Accepted: 22 March 2024 Published: 20 June 2024

ABSTRACT

When building a classification model, the scenario where the samples of one class are significantly more than those of the other class is called data imbalance. Data imbalance causes the trained classification model to be in favor of the majority class (usually defined as the negative class), which may do harm to the accuracy of the minority class (usually defined as the positive class), and then lead to poor overall performance of the model. A method called MSHR-FCSSVM for solving imbalanced data classification is proposed in this article, which is based on a new hybrid resampling approach (MSHR) and a new fine cost-sensitive support vector machine (CS-SVM) classifier (FCSSVM). The MSHR measures the separability of each negative sample through its Silhouette value calculated by Mahalanobis distance between samples, based on which, the so-called pseudo-negative samples are screened out to generate new positive samples (over-sampling step) through linear interpolation and are deleted finally (under-sampling step). This approach replaces pseudo-negative samples with generated new positive samples one by one to clear up the inter-class overlap on the borderline, without changing the overall scale of the dataset. The FCSSVM is an improved version of the traditional CS-SVM. It considers influences of both the imbalance of sample number and the class distribution on classification simultaneously, and through finely tuning the class cost weights by using the efficient optimization algorithm based on the physical phenomenon of rime-ice (RIME) algorithm with cross-validation accuracy as the fitness function to accurately adjust the classification borderline. To verify the effectiveness of the proposed method, a series of experiments are carried out based on 20 imbalanced datasets including both mildly and extremely imbalanced datasets. The experimental results show that the MSHR-FCSSVM method performs better than the methods for comparison in most cases, and both the MSHR and the FCSSVM played significant roles.

KEYWORDS

Imbalanced data classification; Silhouette value; Mahalanobis distance; RIME algorithm; CS-SVM

1 Introduction

Imbalanced data classification is not uncommon in various applications. The identification of financial fraud [1] and market research [2] in the financial field, pathological image recognition [3] and disease detection [4] in the medical field, network intrusion detection [5] and spam information identification [6] in the network security field, as well as fault detection [7] and object inspection task

[8] in the industrial field are all too often encountered with imbalanced data. The available samples of the more noteworthy class, such as the fraudulent instance, the positive case, the spam, and the machine failure are usually far less than those of the opposite class, which can be obtained relatively easily but are not so noteworthy. The noteworthy class is commonly called a positive class or minority class, while the opposite class is called a negative class or majority class. Although the positive class is accompanied by higher misclassification costs, it always gets poor classification precision for the traditional classifiers taking no account of data imbalance are naturally biased to the negative class. Scholars have proposed many methods to conquer this problem, which can be divided into three types, namely the data preprocessing approaches, the algorithm approaches, and the hybrid approaches.

The data preprocessing approaches are mainly based on resampling methods, which can be divided into over-sampling, under-sampling, and hybrid sampling methods, and experts in the field consider the hybrid sampling methods to be the best [9]. The Synthetic Minority Over-Sampling Technique (SMOTE) proposed by Chawla et al. [10] is the most famous over-sampling method, which randomly selects one from k nearest neighbors of the positive sample to generate a new positive sample by linear interpolation with a random number within the range (0, 1) as the parameter. However, the SMOTE does not fully consider the distribution of adjacent samples, so it may cause serious inter-class overlap. Han et al. [11] proposed the Borderline SMOTE method, which only performs nearest-neighbor linear interpolation on a few samples on the borderline, making the distribution of the synthesized positive samples more reasonable. He et al. [12] proposed Adaptive Synthetic Sampling (ADASYN), which automatically determines the number of generated samples based on the density of positive samples in the feature space. More samples are generated in areas with lower density, while fewer samples are generated in areas with higher density, which is beneficial for preventing the production of noisy samples. Random Under-Sampling (RUS) is the simplest under-sampling method, which randomly removes negative samples to achieve data balance, so there is a high risk of information loss. Compared to the over-sampling and the under-sampling methods, the hybrid sampling strategy has significant advantages. Goel et al. [13] proposed the SMOTE-Tomek, which firstly oversamples positive samples using SMOTE and then uses Tomek links to clean up the Tomek sample pairs on the borderline to reduce noisy samples. Agrawal et al. [14] proposed the SMOTE and Clustered Under-sampling Technique (SCUT), the SMOTE is used for over-sampling positive samples, while Expectation Maximization (EM) clustering is used for under-sampling negative samples, which performs better on datasets with high imbalance ratio. Batista et al. [15] proposed SMOTEENN, which uses SMOTE to over-sample data first and then uses the Edited Nearest Neighbor (ENN) to clean out the samples with most near neighbors belonging to the opposite class. Due to its significant effect in removing noisy samples, it has become a research hotspot. In recent years, Generative Adversarial Networks (GANs) have been introduced as an oversampling method in imbalanced data processing. Sharma et al. [16] proposed a novel hybrid approach including SMOTE and GAN (SMOTifed-GAN), which lets the samples SMOTE-generated as input for GAN instead of random numbers, to some extent, improve classification performance. Furthermore, Alberto et al. [17] proposed a new method based on pseudo-negative sampling (PNS), which refers to the samples labeled as negative but very close to the positive samples. The Pseudo-negative samples need to be identified and corrected to be the positive samples to balance the dataset. The PNS needs not to change the overall number of samples, so it does not introduce new noisy samples or cause potential information loss. But it simply identifies pseudo-negative samples based on the Euclidean distance between samples, lacking consideration of the overall distribution of the dataset.

Dealing with imbalanced data problems from an algorithmic perspective is mainly based on cost-sensitive learning and ensemble learning. The cost-sensitive learning corrects the classifier's discrimination against negative classes by assigning a greater misclassification cost to the positive class compared to that of the negative class. The academic community has proposed cost-sensitive versions of various traditional classifiers, such as Logistic Regression (LR), Support Vector Machine (SVM), and Back Propagation Neural Network (BPNN), among which SVM, as a classification model that performs well under small sample size, has received research on its cost sensitivity. Article [18] effectively solved the problem of dataset imbalance based on cost-sensitive semi-supervised laplacian support vector machines. Iranmehr [19] improved the performance of classifiers by modifying the loss function based on SVM combined with cost-sensitive algorithms. These proposed methods all introduce cost weights into SVM, and to some extent alleviate the problem of low accuracy of positive classes caused by data imbalance. However, they only roughly determine class cost weights based on the imbalanced ratio of the dataset, because classification accuracy is not only impacted by data imbalance, but also by distribution characteristics such as inter-class overlap, sub-concepts, and abstractly small sample size. Therefore, these types of methods cannot get a higher classification effect, even greatly sacrificing the accuracy of the negative classes while improving the accuracy of the positive class, and then damaging the overall classification performance. Moreover, Adaptive Cost-Sensitive Boosting (Ada-Cost), Easy Ensemble, and e-Xtreme Gradient Boosting (XG-Boost) [20] are ensemble learning algorithms designed to address class imbalance issues. Article [21] successfully applied the Ada-Cost algorithm with SVM as the base classifier to address imbalanced data, achieving promising results. Article [22] validated the feasibility and effectiveness of the proposed Easy Ensemble approach, which transforms the imbalanced class learning problem into a subproblem of ensemble-based balanced learning. In general, ensemble learning can enhance overall performance by leveraging the strengths of multiple models, while also possessing favorable characteristics such as robustness and strong generalization ability.

Some scholars have combined data resampling with classification algorithms for imbalanced data learning to improve recognition accuracy. Liu [23] proposed an integrated classification algorithm based on random hybrid resampling and genetic algorithm. Firstly, random sampling was used to reduce the number of negative class samples, and the SMOTE method was used to generate positive samples to obtain multiple balanced training data subsets. Based on these, multiple XG-Boost classifiers were trained, and then the outputs were integrated by using a simple voting method to obtain the final classifier. Khattak et al. [24] proposed an improved Balance Cascade algorithm that integrates Bootstrap and XG-Boost, in which, the positive samples are sampled using Bootstrap and the negative samples are sampled using the Balance Cascade algorithm, while XG-Boost is used as the base classifier. Zhao et al. [25] proposed an imbalanced data classification method based on under-sampling and an improved SVM. The proposed method is more stable in classifying imbalanced data and exhibits better performance in many cases. Hussein et al. [26] proposed a hybrid approach combining data pre-processing techniques and SVM optimized by improved Simulated Annealing (SA). It generated new minority samples removed redundancy and duplicated majority samples to equalize the number of samples between classes first, and then used the improved SA algorithm to search optimum penalty parameters for the SVM. Their experimental results demonstrate that the approach performs better on ten public imbalanced datasets in comparison with the conventional SVM.

To sum up, the existing resampling methods have the problems of not fully considering the spatial distribution of data and requiring manual specification of sampling ratios, and the existing cost-sensitive and ensemble learning methods have the problems of the decision of cost weights are too simple to accurately capture sample distribution characteristics. To overcome these problems, this article proposes a new method named MSHR-FCSSVM for imbalanced classification. Where the

MSHR means a Hybrid Resampling method based on Silhouette values calculated on Mahalanobis distance and the FCSSVM means a Fine Cost Sensitive SVM.

The contributions of this article are as follows:

- A new hybrid resampling method called MSHR is proposed, which uses Mahalanobis distance to calculate Silhouette values of samples to better reflect the distribution of the data. The physical phenomenon of the rime-ice (RIME) algorithm is used to find more reasonable pseudo-negative samples, which are based on generating new positive samples that are more correspond to the distribution of existing positive samples.
- Proposed a fine cost-sensitive SVM model called FCSSVM, the cost weights of which are finely determined by the RIME algorithm to make the classification hyper-plane more in accord with the distribution of the imbalanced data.
- Combining the proposed hybrid resampling method with the fine cost-sensitive SVM model to handle imbalanced data classification.

The remainder of this article is organized as follows. Section 1 introduces the key theoretical knowledge that supports the proposed method in this article; Section 2 provides a detailed explanation of the proposed methods and models; Section 3 introduces the experimental datasets and evaluation metrics; Section 4 presents the data experiments and result analysis; Section 5 concludes this article.

2 Related Theories

2.1 Silhouette Value Is Calculated Based on Mahalanobis Distance

The Silhouette score [27] is usually used to evaluate the performance of a clustering algorithm, which is achieved by averaging the Silhouette value of each sample. The Silhouette value measures the similarity between each sample and the class it belongs to, as well as the difference between it and the other classes. The silhouette value of the sample x_i can be calculated by Eq. (1).

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{1}$$

where $a(i)$ indicates the minimum distance between x_i and samples belonging to the same class, and $b(i)$ samples belonging to the other class.

The Silhouette value of a sample is within the range (−1, 1). It means that the sample more strongly belongs to its class than to other classes as it is approaching 1, and the opposite is true as it is approaching −1. So, the bigger the Silhouette value, the better the separability of this sample. Silhouette value is usually calculated based on Euclidean distance. It is proposed in this article to calculate Silhouette values based on Mahalanobis distance.

Mahalanobis distance was proposed by Indian statistician P. C. Mahalanobis [28]. Unlike Euclidean distance, it is based on the overall sample set, considering the connections between the various features of the samples. By introducing the correlation and variance between samples, it can better measure the similarity between samples, so it is commonly used in clustering [29], anomaly detection [30], and pattern recognition [31]. The Mahalanobis distance $D_M(x_i, x_j)$ between the sample x_i and the sample x_j can be calculated by Eq. (2).

$$D_M(x_i, x_j) = \sqrt{(x - y)^T S^{-1} (x - y)} \tag{2}$$

where, S^{-1} is the inverse of the covariance matrix of the dataset, as well as x and y are feature vectors for x_i and x_j, respectively. Yao et al. [32] oversampled imbalanced data based on improved Mahalanobis distance and achieved good results; Siddappa et al. [33] applied the Local Mahalanobis Distance Learning (LMDL) method to the Nearest Neighbor learning method to improve performance of imbalanced classification. This article proposes calculating the Silhouette value based on Mahalanobis distance to reflect the distribution characteristics of samples more accurately under imbalanced data.

2.2 Cost-Sensitive SVM (CS-SVM)

CS-SVM is a variant of SVM designed for imbalanced data and has been commonly used in binary classification [34], multi-classification [35], and regression problems. CS-SVM assigns different misclassification cost weights to different classes and sets higher weights for the positive samples to give them to be given more attention when making predictions, thereby to some extent improving accuracy for the positive class.

Assuming that there are m positive samples and n negative samples, the CS-SVM model can be obtained by resolving the following convex quadratic programming problem.

$$\min \frac{1}{2} \parallel w \parallel^2 + C_1 \sum_{i=1}^{n} \zeta_i + C_2 \sum_{j=1}^{m} \zeta_j \tag{3}$$

where the w is the normal vector of the classification hyper-plane, the C_1 and C_2 are cost weights for the negative class and the positive class, and the C_2 should be set to be greater than the C_1 under-imbalanced data. The ζ_i and ζ_j are slack variables, which are used to indicate the deviation degrees from the correct classification borderline for the negative samples and the positive samples, respectively.

2.3 RIME Algorithm

The RIME optimization algorithm proposed by Su et al. [36] in February 2023 is a new meta-heuristic algorithm inspired by the growth behavior of frost ice in nature. It includes three steps: (1) Simulate the motion of soft fog particles in fog and ice. This step continuously switches between large-scale exploration and small-scale exploration, so it can achieve high efficiency and precision. (2) Simulate the cross behavior between hard fog agents. This step achieves effective information exchange between ordinary agents and optimal agents. (3) Select the most optimal solution by filtering out suboptimal solutions in the population by greedy selection.

The initial population R is composed of n rime agents S_i $(i = 1, 2, \ldots, n)$, each of which consists of d particles x_{ij} $(i = 1, 2, \ldots, n j = 1, 2, \ldots, d)$.

$$R = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_i \end{bmatrix}; S_i = \begin{bmatrix} x_{i1} & x_{i2} & \ldots & x_{ij} \end{bmatrix} \tag{4}$$

The updated position of each rime agent in the condensation process can be calculated by Eq. (5).

$$R_{ij}^{new} = R_{best,j} + r_1 \cdot \cos\theta \cdot \beta \cdot \left(h \left(Ub_{ij} - Lb_{ij}\right) + Lb_{ij}\right), r_2 < E \tag{5}$$

where the $R_{best,j}$ is the jth particle of the best rime agent, the r_1 is a random number within the range (−1, 1), which controls the direction of particle motion. The variable $\cos\theta$ changes with the number of iterations, as shown in Eq. (6). The β is environmental factor is used to simulate the effect of the outer environment, which changes with the number of iterations according to Eq. (7) to accelerate convergence. The h is adhesion degree, which is a random number within the range (0, 1) is used to control the distance between two rime particles. The Ub_{ij} and Lb_{ij} are the upper and lower bounds of the escape space of particles, respectively. The E is coefficient of being attached, increases following iterations and decides the convergence probability of an agent, as shown in Eq. (8). The r_2 is a random number within the range (0, 1), which is used to decide whether to update the particle positions together with the E.

$$\theta = \pi \cdot \frac{t}{10 \cdot T} \tag{6}$$

where the t is the current iteration number and the T is the maximum iteration number.

$$\beta = 1 - \left[\frac{w \cdot t}{T}\right] / w \tag{7}$$

where the w is a constant with a default value equaling to 5, which is used to control the number of segments of the step function β.

$$E = \sqrt{(t/T)} \tag{8}$$

In strong gale conditions, hard-rime growth is simpler and more regular than soft-rime growth, which can be used to update the algorithm between agents, so that the particles of the algorithm can be exchanged and the convergence of the algorithm and the ability to jump out of the local optimum can be improved.

$$R_{ij}^{new} = R_{best,j}, r_3 < F^{normr}(S_i) \tag{9}$$

where R_{ij}^{new} is the new position of the updated particle, and $R_{best,j}$ is the jth particle of the best rime-agent in the rime-population R. The $F^{normr}(S_i)$ indicates the normalized value of the current agent fitness value, indicating the chance of the ith rime-agent being selected. r_3 is a random number within $(-1, 1)$.

3 Methodology and Model Structure

3.1 Hybrid Resampling Based on Silhouette Value

As mentioned earlier, the PNS method proposed in recent years implements hybrid sampling by replacing pseudo-negative samples with positive samples directly. It has the advantage of not changing the overall number of the imbalanced dataset used in this experiment, so introducing no new noisy samples and causing no information loss. However, it simply identifies pseudo-negative samples by calculating Euclidean distances between samples, lacking consideration of the overall distribution of the dataset. In addition, directly flipping the class label to obtain a new positive sample also lacks exploration of the classification borderline.

The MSHR proposed in this article has made two improvements to the PNS. One is to identify pseudo-negative samples by calculating the Silhouette value of each negative sample based on Mahalanobis distance. As mentioned earlier, the Mahalanobis distance incorporates the distribution information of samples, and the Silhouette value is essentially an accurate measure of the intra-class similarity and the inter-class difference. Therefore, it is more suitable for identifying the pseudo-negative samples, which are marked as negative but closer to positive. In addition, the RIME together with a validation dataset and SVM classifier are used to determine the threshold for identifying pseudo-negative samples more cautiously. The second is to linearly interpolate each pseudo-negative sample with the most representative positive sample, which has the maximum Silhouette value, to generate a new positive sample to replace the pseudo-negative sample. Where the reciprocal of the

Imbalance Ratio (IR) of the dataset is used as the interpolation parameter to make the generated new positive sample retreat from the distribution range of the negative class to that of the positive class precisely. Therefore, the risk of inter-class overlap is reduced effectively.

The schematic diagram of the proposed hybrid sampling method is shown in Fig. 1, and the specific measurement depends on different datasets.

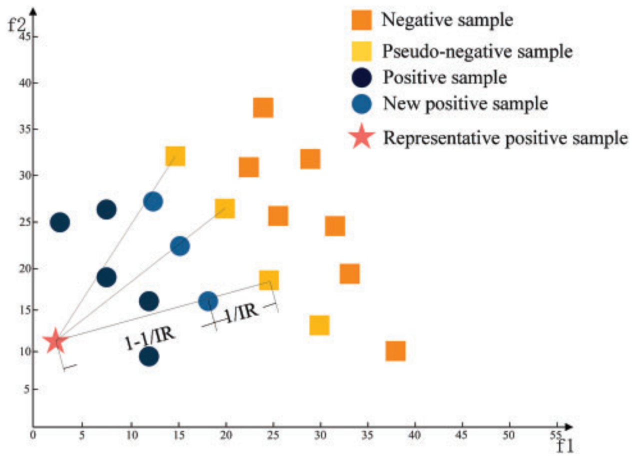

Figure 1: Example plot of MSHR

The MSHR algorithm process is shown in the pseudo-code (Algorithm 1: MSHR). In the pseudocode below, D^+ and D^- represent the positive and negative samples in an imbalanced dataset.

Algorithm 1: Pseudocode of MSHR

```
Input: $\{(x_i, y_i)\}_{i=1}^{|D^-|}, \{(x_j, y_j)\}_{j=1}^{|D^+|}$
Output: $\{(x_i, y_i)\}_{i=1}^{|D^- -i|}, \{(x_j, y_j)\}_{j=1}^{|D^+ +i|}$
1. For $i \leftarrow 1$ to $|D^-|$ do
2.     For $j \leftarrow 1$ to $|D^+|$ do
3.         $x_i, x_j \leftarrow D_{M-01}(x_i, x_j)$
4.         $\min D_{M-01}(x_i, x_j)$
5.     End
6. End
7. For $i \leftarrow 1$ to $|D^-|$ do
8.     For $j \leftarrow 1$ to $|D^-|$ do
9.         $x_i, x_j \leftarrow D_{M-00}(x_i, x_j)$
10.        $\min D_{M-00}(x_i, x_j)$
11.    End
12. End
13. For $i \leftarrow 1$ to $|D^-|$ do
14.    $\min D_M(x_i, x_j), \min D_{M-00}(x_i, x_j) \leftarrow S^-(i)$ followed by Eq. (1)
15. $S^-$ (i) less than the threshold x is a pseudo-negative sample
```

(Continued)

Algorithm 1 (continued)

```
16. Calculate the $S^+$ (i) of $D^+$, max ($S^+$ (i))   corresponds to the representative positive sample x*
17. $x_{new} = \frac{1}{IR} \times x^* + \left(1 - \frac{1}{IR}\right) \times x^-$
18.         $D^+$ append $x_{new}$, $D^-$ delete $x^-$
19.       End
20.   End
```

3.2 *Fine Cost-Sensitive SVM Method*

Unlike most of the existing cost-sensitive models and ensemble learning algorithms that solve imbalanced classification problems by simply determining cost weights based on IR, the FCSSVM applies the RIME algorithm to finely tune the class cost weights. The class cost weights are set to be particles, and the mean of five performance evaluation metrics (described in Section 3.2) obtained on a given validation dataset is set as the fitness function value. FCSSVM considers the impact of class distribution as well as data imbalance to achieve better overall performance. The specific algorithm process is shown in the pseudo-code (Algorithm 2: FCSSVM):

Algorithm 2: Pseudocode of FCSSVM

```
Input: $X$, $class\_weight \leftarrow fitness = 0.2 \times (AUC + F1 + Recall + ACC + G - mean)$
Output: $\hat{X}$, $Class\_\hat{w}eight$, the trained parameters
Initialize the rime population R
Get the current optimal agent and optimal fitness
1. While $t \leq T$
2.     Coefficient of adherence E calculated by Eq. (8)
3.     If {r_2} E
4.         Update rime agent location by the soft-rime search
5.     Strategy
6.     End If
7.     If {r_3} Normalizefitness of {S_i}
8.         Cross updating between agents by the hard-rime
9.     puncture mechanism
10.   End If
11.   If $F\left(R_i^{new}\right) FR_i)$
12.     Select the optimal solution and replace the suboptimal solution
        using the positive greedy selection mechanism
13.     End If
14.     t = t + 1
15.   End While
```

3.3 *Model Structure*

The structure of the proposed MSHR-FCSSVM is shown in Fig. 2.

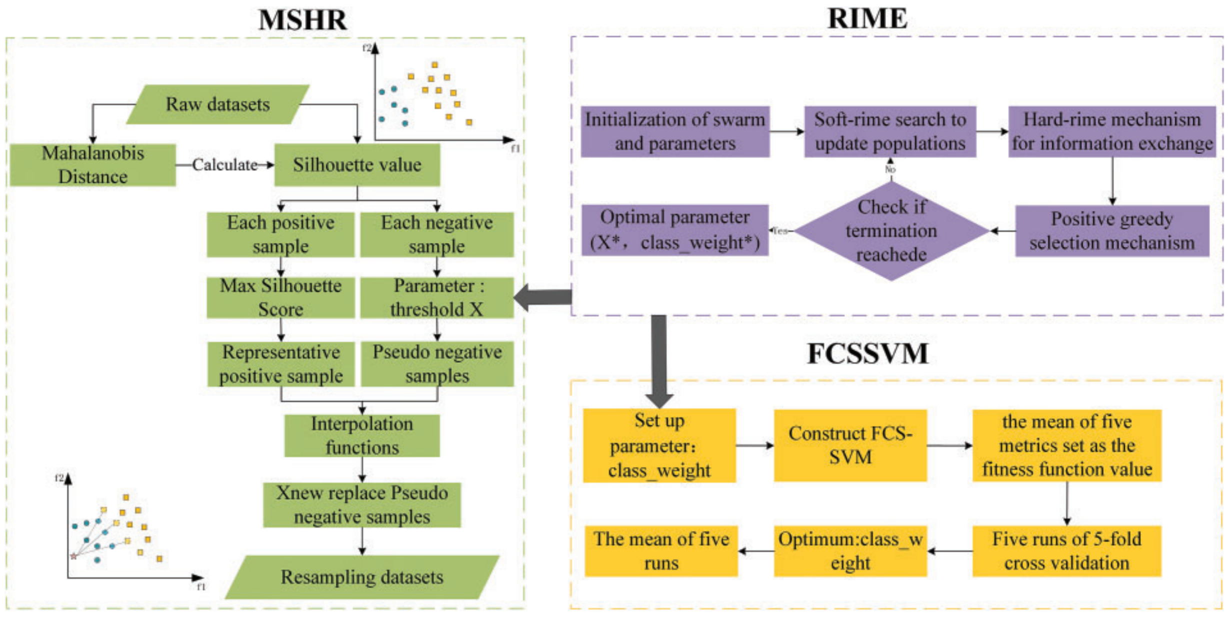
MSHR
Raw datasets
Mahalanobis Distance
Calculate
Silhouette value
Each positive sample
Each negative sample
Max Silhouette Score
Parameter : threshold X
Representative positive sample
Pseudo negative samples
Interpolation functions
Xnew replace Pseudo negative samples
Resampling datasets
RIME
Initialization of swarm and parameters
Soft-rime search to update populations
Hard-rime mechanism for information exchange
Positive greedy selection mechanism
Check if termination reachede
No
Yes
Optimal parameter (X*, class_weight*)
FCSSVM
Set up parameter: class_weight
Construct FCS-SVM
the mean of five metrics set as the fitness function value
Five runs of 5-fold cross validation
Optimum:class_weight
The mean of five runs

Figure 2: The structure of MSHR-FCSSVM

4 Dataset and Evaluation Metrics

4.1 Dataset

Eighteen datasets are selected from the UCI database (https://archive.ics.uci.edu/datasets) and the Keel database (https://sci2s.ugr.es/keel/datasets.php) and two datasets are synthesized for the experiment, eleven of which are mildly imbalanced, and the others are extremely imbalanced. According to Noorhalim et al. [37], a dataset with $1.9 < \text{IR} < 9$ can be defined as mildly imbalanced, and a dataset with IR>=9 is extremely imbalanced. These datasets are described in detail in Table 1.

Table 1: The information of datasets

IR category	Dataset	Number of samples	Number of features	IR
Mildly class imbalanced	pageblocks	548	10	164
	abalone19	4174	8	129.44
	abalone20vs8910	1916	8	72.69
	ecoli	336	7	71.5
	winequalityred3vs5	691	11	68.1
	winequalitywhite39vs5	1482	11	58.28
	winequalitywhite3vs7	900	11	44
	yeast1289vs7	947	8	30.6
	synthetic dataset2	2000	2	19
	ecoli0146vs5	280	6	13
	synthetic dataset1	2000	2	9

(Continued)

Table 1 (continued)

IR category	Dataset	Number of samples	Number of features	IR
	ecoli3	336	7	8.6
	wine_red	1599	11	7.04
	newthyroid	215	5	5.14
	SPECT	269	27	3.89
Extremely class imbalanced	ecoli1	336	7	3.36
	haberman	306	3	2.78
	yeast1	1484	8	2.46
	phoneme	5404	5	2.41
	pima	768	8	1.87

4.2 Model Evaluation Metrics

Six commonly used performance metrics for imbalanced data classification, namely Precision, accuracy (ACC), Recall, F1 score, the area under curve (AUC), and geometric mean of sensitivity (G-mean) [38] are introduced, which are given by the Eqs. (10) to (15). These metrics are calculated based on the confusion matrix shown in Table 2, where True Positive (TP) represents the number of actual positive samples predicted as positive, True Negative (TN) represents the number of actual negative samples predicted as negative, False Negative (FN) represents the number of actual positive samples predicted as negative, and False Positive (FP) represents the number of actual negative samples predicted as positive.

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

$$ACC = \frac{TP + TN}{TP + FP + TN + FN} \tag{11}$$

$$Recall = \frac{TP}{TP + FN} \tag{12}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{13}$$

$$G - mean = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}} \tag{14}$$

$$AUC = 1 - \frac{\frac{FP}{N} + \frac{FN}{M}}{2} \tag{15}$$

Table 2: Confusion matrix

	Predicted positive	Predicted negative
True positive	TP	FN
True negative	FP	TN

The ACC metric reflects the overall classification accuracy of the model; The Precision and Recall metrics respectively reflect the model's ability to eliminate negative samples and recognize positive samples; The F1 score, G-mean, and AUC all reflect the comprehensive performance of the model in identifying positive and negative samples.

4.3 Experimental Confirmation and Results Discussion

4.3.1 Experimental Environment

The experiments are conducted on a computer with an i7-12700F CPU and 32 GB memory. The related software includes the Windows 11.0 OS, PyCharm 2020, the Python 3.8 library, and the Scikit-learn1.1.1 library. All the algorithms proposed in this article are implemented through programming with Python language.

4.3.2 Performance Experiment on the MSHR Method

To verify the effectiveness of the proposed MSHR method, it is applied to experiment on the given 20 imbalanced datasets. A standard SVM considering no cost sensitivity is used as the classifier here to ensure the result is only caused by the resampling method. To make the experimental results more reliable, the 5-fold cross-validation method is applied, and a quarter of the training data are used for running the RIME to set up the Silhouette value threshold X for finding out the pseudo-negative samples. Some resampling methods mentioned earlier, i.e., the SMOTE, the ADASYN, the RUS, the SMOTEENN, the SMOTifed-GAN, and the PNS are used for comparison. The experimental results are shown in Table 3.

Table 3: Results of SMOTE, ADASYN, RUS, SMOTEENN, SMOTifed-GAN, PNS and MSHR

Dataset	Metrics	SVM						
		SMOTE	ADASYN	RUS	SMOTEENN	SMOTifed-GAN	PNS	MSHR
pageblocks	AUC	0.560	0.574	0.561	0.560	0.606	\	**0.654**
	F1	0.206	**0.247**	0.209	0.206	0.205	\	0.209
	Recall	0.124	**0.178**	0.124	0.124	0.108	\	0.124
	ACC	0.906	0.888	0.907	0.906	0.906	\	**0.908**
	G-mean	0.288	0.278	0.296	0.288	0.316	\	**0.319**
abalone19	AUC	0.797	0.788	0.686	0.794	0.809	0.812	**0.823**
	F1	0.258	0.256	0.226	0.254	0.297	0.272	**0.310**
	Recall	0.790	0.773	**0.895**	0.807	0.862	0.823	0.866
	ACC	**0.804**	0.802	0.480	0.782	0.731	0.620	0.733
	G-mean	0.354	0.350	0.309	0.349	0.296	**0.388**	0.301
abalone20vs8910	AUC	0.832	0.839	0.700	0.837	0.825	0.843	**0.885**
	F1	0.257	0.243	0.053	0.233	0.268	0.262	**0.322**
	Recall	0.720	0.740	0.760	0.740	0.713	0.800	**0.820**
	ACC	**0.941**	0.935	0.641	0.932	0.886	0.789	0.819
	G-mean	0.334	0.326	0.144	0.318	0.326	**0.361**	0.348

(Continued)

Table 3 (continued)

Dataset	Metrics	SVM						
		SMOTE	ADASYN	RUS	SMOTEENN	SMOTifed-GAN	PNS	MSHR
ecoli	AUC	0.977	0.971	0.976	0.977	0.969	0.973	**0.986**
	F1	0.972	0.962	0.970	0.970	0.964	0.966	**0.972**
	Recall	0.989	**0.996**	0.989	0.992	0.964	0.982	0.985
	ACC	0.976	0.967	0.974	0.974	0.970	0.971	**0.976**
	G-mean	0.972	0.963	0.970	0.971	0.964	0.967	**0.972**
winequalityred3vs5	AUC	0.612	0.609	0.629	0.590	0.650	0.653	**0.753**
	F1	0.140	0.139	0.141	0.136	0.154	0.150	**0.155**
	Recall	0.650	0.650	0.750	0.650	0.769	0.750	**0.800**
	ACC	0.575	0.570	0.513	0.532	0.585	0.560	**0.592**
	G-mean	0.215	0.214	0.225	0.209	**0.257**	0.237	0.150
winequalitywhite39vs5	AUC	0.618	0.621	0.607	0.608	0.645	0.654	**0.677**
	F1	0.262	0.263	0.356	0.252	0.321	0.322	**0.379**
	Recall	0.480	0.500	0.280	0.620	0.606	0.600	**0.679**
	ACC	0.751	0.738	**0.923**	0.596	0.708	0.718	0.793
	G-mean	0.325	0.328	0.376	0.328	0.375	0.382	**0.442**
winequalitywhite3vs7	AUC	0.795	0.787	0.618	0.776	0.803	0.779	**0.822**
	F1	0.158	0.147	0.197	0.134	0.201	0.226	**0.255**
	Recall	0.775	0.775	0.275	0.775	**0.787**	0.500	0.700
	ACC	0.815	0.799	**0.947**	0.777	0.817	0.766	0.838
	G-mean	0.261	0.251	0.221	0.238	0.279	0.282	**0.323**
yeast1289vs7	AUC	0.685	0.679	0.643	0.671	0.652	0.677	**0.704**
	F1	0.136	0.130	0.102	0.115	0.178	0.186	**0.249**
	Recall	0.616	0.616	0.566	0.666	0.725	**0.733**	0.600
	ACC	0.750	0.738	0.714	0.675	0.809	0.778	**0.825**
	G-mean	0.217	0.211	0.176	0.205	0.236	**0.277**	0.209
synthetic dataset2	AUC	0.733	0.737	0.704	0.751	0.729	0.785	**0.841**
	F1	0.227	0.228	0.222	**0.231**	0.225	0.227	0.226
	Recall	0.383	0.383	0.376	0.425	0.372	0.500	**0.512**
	ACC	0.774	**0.782**	0.692	0.746	0.719	0.776	0.775
	G-mean	0.252	0.252	0.216	0.264	0.236	0.243	**0.285**
ecoli0146vs5	AUC	0.905	0.900	0.878	0.903	0.875	0.922	**0.969**
	F1	0.817	0.779	0.617	0.801	**0.857**	0.821	0.793
	Recall	0.825	0.825	0.825	0.825	0.750	0.820	**0.850**
	ACC	0.975	0.966	0.925	0.971	0.972	0.967	**0.976**
	G-mean	0.831	0.798	0.641	0.812	**0.866**	0.827	0.797
synthetic dataset1	AUC	0.711	0.699	0.695	0.682	0.689	0.718	**0.735**
	F1	0.354	0.319	0.318	0.302	0.335	0.368	**0.375**
	Recall	0.639	0.700	0.700	0.679	0.600	0.710	**0.717**
	ACC	**0.768**	0.699	0.691	0.683	0.750	0.728	0.752
	G-mean	0.396	0.380	0.379	0.363	0.356	**0.423**	0.416

(Continued)

Table 3 (continued)

Dataset	Metrics	SVM						
		SMOTE	ADASYN	RUS	SMOTEENN	SMOTifed-GAN	PNS	MSHR
ecoli3	AUC	0.892	0.893	0.895	0.903	0.897	0.904	**0.907**
	F1	0.623	0.609	0.557	0.619	0.603	0.683	**0.743**
	Recall	0.900	0.914	**0.971**	0.942	0.528	0.742	0.600
	ACC	0.886	0.877	0.835	0.879	0.918	0.926	**0.931**
	G-mean	0.656	0.647	0.616	0.660	0.607	**0.690**	0.648
wine_red	AUC	0.713	0.706	0.624	0.667	0.725	0.757	**0.769**
	F1	0.360	0.348	0.281	0.312	0.396	0.440	**0.455**
	Recall	0.812	0.837	0.820	**0.847**	0.803	0.715	0.722
	ACC	0.638	0.607	0.477	0.532	0.715	0.639	**0.717**
	G-mean	0.433	0.429	0.373	0.402	0.454	0.421	**0.494**
newthyroid	AUC	0.874	0.897	0.796	0.877	0.871	0.896	**0.923**
	F1	**0.818**	0.726	0.664	0.812	0.713	0.725	0.640
	Recall	0.771	**0.914**	0.657	0.785	0.842	0.842	0.785
	ACC	0.944	0.886	0.890	0.939	0.860	0.932	**0.950**
	G-mean	0.827	0.743	0.684	0.821	0.777	0.777	**0.850**
SPECT	AUC	0.770	0.747	**0.826**	0.746	0.772	0.816	0.781
	F1	0.592	0.551	0.624	0.529	0.602	0.642	**0.783**
	Recall	0.727	0.727	0.909	0.818	0.713	0.914	**0.935**
	ACC	0.796	0.759	0.777	0.703	0.794	**0.818**	0.791
	G-mean	0.603	0.568	0.657	0.565	0.612	0.658	**0.789**
ecoli1	AUC	0.888	0.889	0.886	0.892	0.914	0.875	**0.937**
	F1	0.775	0.753	0.747	0.758	**0.840**	0.795	0.786
	Recall	0.900	**0.940**	0.940	0.940	0.866	0.820	0.829
	ACC	0.882	0.861	0.856	0.865	0.895	0.892	**0.898**
	G-mean	0.784	0.770	0.764	0.773	0.745	**0.797**	0.790
haberman	AUC	0.635	0.615	0.624	0.626	0.669	0.620	**0.699**
	F1	0.449	0.440	0.422	0.452	0.513	0.511	**0.528**
	Recall	0.412	0.493	0.343	0.500	0.517	**0.843**	0.525
	ACC	0.740	0.673	**0.757**	0.686	0.737	0.667	0.741
	G-mean	0.454	0.458	0.438	0.462	0.494	0.478	**0.537**
yeast1	AUC	0.714	0.711	0.716	0.698	0.716	0.681	**0.776**
	F1	0.591	0.586	0.593	0.574	0.592	0.558	**0.593**
	Recall	0.760	0.844	0.737	0.867	0.782	**0.872**	0.725
	ACC	0.694	0.655	0.707	0.627	0.704	0.600	**0.712**
	G-mean	0.606	**0.616**	0.605	0.610	0.609	0.598	0.603
phoneme	AUC	0.834	0.830	0.830	0.833	0.819	0.866	**0.893**
	F1	0.731	0.721	0.724	0.728	0.712	0.703	**0.731**
	Recall	0.908	**0.938**	0.913	0.916	0.804	0.820	0.803
	ACC	0.803	0.786	0.796	0.798	0.823	0.804	**0.825**
	G-mean	**0.745**	0.741	0.740	0.743	0.722	0.667	0.733

(Continued)

Table 3 (continued)

Dataset	Metrics	SVM						
		SMOTE	ADASYN	RUS	SMOTEENN	SMOTifed-GAN	PNS	MSHR
pima	AUC	0.716	0.713	0.709	0.725	0.796	0.797	**0.799**
	F1	0.637	0.637	0.629	0.653	0.665	0.610	**0.679**
	Recall	0.711	0.735	0.705	0.809	0.728	**0.962**	0.615
	ACC	0.717	0.707	0.710	0.699	0.723	0.658	**0.733**
	G-mean	0.642	0.643	0.633	0.666	0.670	0.678	**0.680**

As can be seen from Table 3, compared with the other methods, the MSHR takes an advantage in the overall performance. For three of the extremely imbalanced datasets (i.e., the ecoli, winequalityred3vs5, and winequalitywhite39vs5), and two of the mildly imbalanced datasets (i.e., the wine_red and pima), it gets the highest values on 4 of the 5 metrics.

Fig. 3 contrasts the sample numbers of the two classes in each dataset before and after being resampled by the MSHR. It can be seen that the number imbalance of each dataset is reduced, nonetheless, it is still far from being eliminated. It is because the MSHR is by nature to make the classification borderline cleaner by diminishing the pseudo-negative samples and generating new positive samples accordingly, but not to pursue merely an abstract balance on sample numbers.

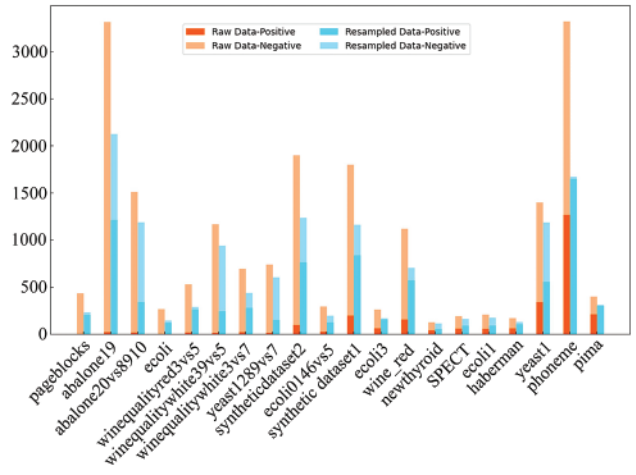

Figure 3: Positive and negative sample numbers before and after using MSHR

To further study how the MSHR acts on an imbalanced dataset, a simulated imbalanced dataset (IR = 10) containing only 3 features is generated for visualization. Fig. 4a shows the raw dataset displayed in a 3D scatter chart. The positive samples and the negative samples overlapped on the borderline. Figs. 4b to 4h show the results given by the MSHR and the other resampling methods.

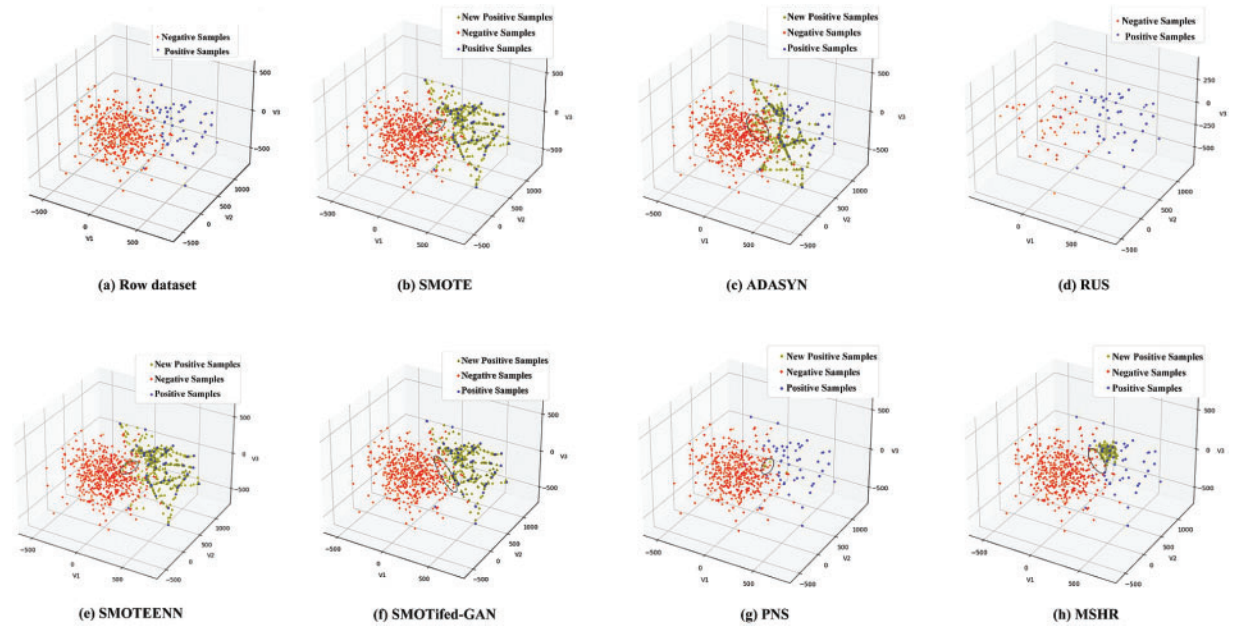

Figure 4: Results given by the MSHR and the other resampling methods

The SMOTE method leads to a more serious inter-class overlap for it does not discriminate between samples inside and outside the overlap area when choosing core samples to generate new samples. The ADASYN method generates new samples only based on the positive samples nearing the borderline and lacks the consideration of the overall distribution of the positive class, so it intensifies inter-class overlap, too. The RUS method effectively reduces negative samples but loses important samples in favor of constructing classification boundaries for its way of randomly choosing negative samples to delete. The SMOTEENN method relieves inter-class overlap to some extent through cleaning of samples being regarded as noise, but it is still not well for lacking consideration of the overall distribution of the dataset. The SMOTifed-GAN is limited by the quality of the samples generated by SMOTE and the training of the GAN model, therefore, it is greatly affected by the characteristics of the sample distribution, it has the same problem as SMOTE. The PNS method generates very few positive samples for it identifies pseudo-negative samples based on spatial Euclidean distance and may cease to be effective as the clustering centers of positive and negative samples are far apart. Additionally, the new samples generated by it are still trapped in the overlapping area for they are simply replacements of the pseudo-negative samples. In contrast, the MSHR can effectively screen out the pseudo-negative samples based on the Silhouette value calculated by Mahalanobis distance and generates new positive samples more agreed with the distribution of the positive class through linear interpolation, so making the borderline cleaner and is more in favor of classification. We conducted runtime statistics on MSHR with other resampling methods, and the results are shown in Table 4.

As can be seen from Table 4, compared with traditional resampling methods, MSHR has a longer runtime due to the optimization process. However, the training requirements of the GAN network result in its time consumption being longer than MSHR. According to the literature review, there are still many studies that have introduced intelligent optimization algorithms in the process of resampling. A method named Ant Colony Optimization Resampling (ACOR) was proposed in [39], and a genetic optimization resampling method was proposed in [40]. They all significantly improved the performance of imbalanced classification through parameter optimization at the cost of efficiency. Therefore, though our method has no advantages on account of time efficiency, given its better overall performance, it still can be regarded as valuable for classification applications that do not lay too much stress on efficiency.

Table 4: Comparison of resampling time between MSHR and other resampling methods

Dataset	Run time (s)						
	SMOTE	ADASYN	RUS	SMOTEENN	SMOTifed-GAN	PNS	MSHR
ecoli	5.987	6.353	6.009	5.932	22.928	5.914	16.877
ecoli0146vs5	6.232	6.094	6.756	8.477	19.991	6.343	16.104
ecoli3	6.486	6.048	7.765	8.203	23.222	6.025	16.889
newthyroid	7.451	6.949	6.133	7.167	18.240	6.114	14.473
SPECT	6.711	6.141	6.078	6.004	20.404	5.973	23.092
ecoli1	6.037	7.286	6.680	6.509	23.460	5.980	16.723
haberman	7.969	6.763	6.130	5.964	20.095	6.012	16.663

4.3.3 Performance Experiment on the MSHR-FCSSVM

To verify the effectiveness of the proposed MSHR-FCSSVM method, comparative experiments were conducted with some other methods. In which, the Ada-cost, the Easy-Ensemble, and the XG-Boost are both ensemble learning methods reported to be effective, the CS-SVM is a commonly used cost-sensitive SVM whose class cost weights are set directly according to IR. In addition, the LR and the BPNN are introduced into our model framework to replace SVM for comparison, i.e., the MSHR-FCSLR and the MSHR-RCSBPNN. Five runs of the 5-fold cross-validation method are used to ensure the reliability of the experiment, and the average results for each dataset are shown in Table 5.

Table 5: Performance comparison of MSHR-FCSSVM with other methods

Dataset	Metrics	Ada-cost	Easy-ensemble	XG-boost	CS-SVM	MSHR-FCSLR	MSHR-FCSBPNN	MSHR-FCSSVM
pageblocks	AUC	0.815	0.831	0.792	0.823	**0.813**	0.838	0.592
	F1	**0.423**	0.414	0.300	0.185	0.208	0.200	0.250
	Recall	0.754	0.645	0.400	**1.000**	0.109	0.119	0.144
	ACC	0.663	0.719	**0.983**	0.102	0.736	0.818	0.914
	G-mean	0.433	**0.438**	0.315	0.319	0.318	0.315	**0.446**
abalone19	AUC	0.741	0.730	0.500	0.588	0.765	0.774	**0.817**
	F1	0.268	0.137	0.000	0.115	0.352	0.336	**0.385**
	Recall	0.495	0.785	0.000	**1.000**	0.796	0.802	0.952
	ACC	0.981	0.676	**0.992**	0.108	0.763	0.852	0.893
	G-mean	0.271	0.222	0.000	0.187	0.296	0.285	**0.301**

(Continued)

Table 5 (continued)

Dataset	Metrics	Ada-cost	Easy-ensemble	XG-boost	CS-SVM	MSHR-FCSLR	MSHR-FCSBPNN	MSHR-FCSSVM
abalone20vs8910	AUC	0.814	0.810	0.887	0.603	0.802	0.792	**0.888**
	F1	0.156	0.108	0.234	0.126	0.325	0.307	**0.337**
	Recall	0.706	0.960	0.399	**1.000**	0.854	0.600	0.860
	ACC	**0.978**	0.783	0.965	0.114	0.918	0.976	0.973
	G-mean	0.265	0.233	0.259	0.216	0.314	0.316	**0.339**
ecoli	AUC	0.983	0.960	**0.989**	0.984	0.985	0.983	0.987
	F1	0.959	0.953	0.943	0.946	0.965	0.971	**0.972**
	Recall	0.938	0.957	0.964	**0.992**	0.982	0.980	0.985
	ACC	0.971	0.961	0.952	0.952	0.963	0.968	**0.976**
	G-mean	0.959	0.954	0.944	0.947	0.956	0.962	**0.972**
winequality-red3vs5	AUC	0.695	0.697	0.634	0.447	0.703	0.753	**0.799**
	F1	0.133	0.124	0.123	0.128	0.147	0.152	**0.158**
	Recall	0.400	0.700	0.300	**1.000**	0.500	0.603	0.800
	ACC	0.982	0.695	0.966	0.117	**0.985**	0.857	0.620
	G-mean	0.223	0.252	0.215	0.220	0.242	0.240	**0.254**
winequality-white39vs5	AUC	0.635	0.706	**0.821**	0.525	0.690	0.707	0.709
	F1	0.294	0.275	0.156	0.130	0.348	0.341	**0.432**
	Recall	0.280	0.720	0.200	**0.919**	0.800	0.820	0.850
	ACC	**0.979**	0.693	0.962	0.215	0.466	0.544	0.677
	G-mean	0.296	0.269	0.162	0.219	0.340	0.362	**0.440**
winequality-white3vs7	AUC	0.722	0.802	0.806	0.348	0.800	0.798	**0.824**
	F1	0.213	0.234	0.158	0.043	0.215	0.213	**0.255**
	Recall	0.450	0.850	0.150	**1.000**	0.696	0.695	0.705
	ACC	0.756	0.856	**0.962**	0.022	0.817	0.812	0.842
	G-mean	0.246	0.249	0.170	0.149	0.299	0.298	**0.311**
yeast1289vs7	AUC	0.619	0.718	0.773	0.728	0.750	0.758	**0.798**
	F1	0.241	0.235	0.184	0.107	0.261	0.225	**0.272**
	Recall	0.266	**0.733**	0.300	0.700	0.612	0.566	0.533
	ACC	0.950	0.704	**0.960**	0.643	0.831	0.825	0.870
	G-mean	0.208	0.204	0.200	0.201	0.208	0.209	**0.215**
synthetic dataset2	AUC	0.749	0.740	0.745	0.687	0.806	0.795	**0.849**
	F1	0.336	0.322	0.328	0.256	**0.356**	0.335	0.338
	Recall	0.345	0.333	0.339	0.223	0.468	0.436	**0.525**
	ACC	0.781	0.751	0.773	**0.916**	0.835	0.831	0.832
	G-mean	0.402	0.391	0.399	0.255	0.396	0.392	**0.403**
ecoli0146vs5	AUC	0.857	0.950	0.940	0.947	0.846	0.878	**0.952**
	F1	0.674	0.611	0.759	0.585	0.600	0.552	**0.800**
	Recall	0.750	**1.000**	0.650	0.800	0.750	0.750	0.860
	ACC	0.950	0.907	**0.971**	0.917	0.928	0.803	0.924
	G-mean	0.695	0.663	0.779	0.614	0.612	0.616	**0.797**

(Continued)

Table 5 (continued)

Dataset	Metrics	Ada-cost	Easy-ensemble	XG-boost	CS-SVM	MSHR-FCSLR	MSHR-FCSBPNN	MSHR-FCSSVM
synthetic dataset1	AUC	0.651	0.663	0.591	0.607	0.636	0.621	**0.774**
	F1	0.333	0.304	0.322	0.314	**0.418**	0.396	0.409
	Recall	0.400	0.580	0.487	0.296	0.606	0.597	**0.718**
	ACC	0.853	0.738	0.834	**0.925**	0.699	0.692	0.805
	G-mean	0.339	0.346	0.345	0.309	0.395	0.385	**0.417**
ecoli3	AUC	0.860	0.926	0.906	0.886	0.850	0.847	**0.940**
	F1	0.524	0.614	0.460	0.459	0.719	0.708	**0.727**
	Recall	0.600	**1.000**	0.342	0.742	0.835	0.829	0.828
	ACC	**0.986**	0.868	0.919	0.818	0.904	0.895	0.934
	G-mean	0.530	0.666	0.506	0.497	0.723	0.721	**0.734**
wine_red	AUC	0.565	0.553	**0.821**	0.579	0.509	0.537	0.597
	F1	0.658	0.568	0.293	0.733	0.501	0.584	**0.738**
	Recall	0.682	0.725	0.182	**0.998**	0.516	0.525	0.985
	ACC	0.800	0.711	**0.847**	0.582	0.590	0.506	0.597
	G-mean	0.659	0.601	0.374	0.760	0.521	0.509	**0.762**
newthyroid	AUC	0.925	0.866	0.896	**0.993**	0.916	0.913	0.939
	F1	0.701	0.695	0.702	0.599	0.713	0.713	**0.717**
	Recall	0.757	0.771	0.742	**1.000**	0.800	0.714	0.800
	ACC	**0.972**	0.962	0.958	0.781	0.876	0.853	0.883
	G-mean	0.708	0.717	0.756	0.654	0.735	0.725	**0.757**
SPECT	AUC	0.823	0.678	0.716	0.761	0.792	0.854	**0.858**
	F1	0.644	0.468	0.121	0.387	0.618	0.632	**0.854**
	Recall	0.527	0.654	0.072	0.436	0.706	0.727	**0.800**
	ACC	0.801	0.692	0.796	0.732	0.714	**0.811**	0.799
	G-mean	0.850	0.489	0.172	0.394	0.851	0.853	**0.862**
ecoli1	AUC	0.774	0.876	0.940	0.917	0.908	0.901	**0.941**
	F1	0.649	0.759	0.741	0.727	0.766	0.765	**0.796**
	Recall	0.653	**0.880**	0.746	0.819	0.779	0.781	0.790
	ACC	**0.941**	0.874	0.886	0.860	0.856	0.823	0.898
	G-mean	0.654	0.767	0.747	0.734	0.728	0.713	**0.778**
haberman	AUC	0.546	0.601	0.669	0.606	0.714	0.715	**0.740**
	F1	0.339	0.431	0.044	0.424	0.431	0.421	**0.434**
	Recall	0.337	0.625	0.025	**0.663**	0.525	0.565	0.575
	ACC	**0.845**	0.590	0.737	0.529	0.721	0.738	0.741
	G-mean	0.340	0.460	0.070	0.456	0.459	0.460	**0.461**
yeast1	AUC	0.746	0.707	0.738	0.740	0.755	0.713	**0.778**
	F1	0.569	0.572	0.433	0.550	0.569	0.573	**0.580**
	Recall	0.790	0.725	0.318	**0.874**	0.774	0.569	0.804
	ACC	0.712	0.719	0.758	0.586	0.618	**0.774**	0.664
	G-mean	0.597	0.594	0.465	0.592	0.598	0.598	**0.604**

(Continued)

Table 5 (continued)

Dataset	Metrics	Ada-cost	Easy-ensemble	XG-boost	CS-SVM	MSHR-FCSLR	MSHR-FCSBPNN	MSHR-FCSSVM
phoneme	AUC	0.905	0.915	0.906	0.888	0.912	0.908	**0.917**
	F1	0.741	0.717	0.685	0.640	0.716	0.704	**0.763**
	Recall	0.835	0.836	0.622	**0.964**	0.889	0.874	0.905
	ACC	**0.849**	0.806	0.832	0.680	0.810	0.826	0.834
	G-mean	0.741	0.725	0.688	0.679	0.768	0.753	**0.772**
pima	AUC	0.746	0.747	**0.827**	0.809	0.785	0.698	0.794
	F1	0.635	0.673	0.592	0.676	0.676	0.636	**0.717**
	Recall	0.722	0.782	0.500	0.798	0.681	**0.907**	0.694
	ACC	0.714	0.736	**0.759**	0.734	0.693	0.636	0.699
	G-mean	0.706	0.680	0.604	0.685	0.712	0.666	**0.722**

As can be seen from Table 5, the proposed MSHR-FCSSVM performs best from the look on the whole. It outperforms the others on two metrics in 4 datasets, on three metrics in 14 datasets, and on four metrics in 1 dataset. The Ada-cost, the Easy Ensemble, the XG-Boost, and the CS-SVM behave better on a certain metric of some datasets. The Ada-cost method updates the cost value of misclassified samples in each round of training based on the previous round of testing results. More negative samples have a chance to be increased cost values than the positive ones, leading the Acc metric to improve steadily. The Easy Ensemble method simply splits the negative samples into multiple subsets of equal size as that of the positive class to achieve balance in sample number, which may result in loss of distribution information of the data, so behave worst and only win on a metric for 5 datasets. The XG-Boost method sets class weight parameters according to the IR directly, so it improves the performance of the positive class to some extent but it may come at the cost of significant performance reduction of the negative class. The CS-SVM gets very high Recall values on most of the datasets. This is because it excessively enlarges the class weight of the positive class based on the IR, which makes the positive samples classified finely but impairs the accuracy of negative samples meanwhile, so they do not perform well on the comprehensive metrics. The MSHR-FCSLR is a maximum likelihood estimation algorithm that is easy to implement but often has poor classification performance when the feature space is large. A typical example includes SPECT. The MSHR-FCSBPNN is a local search optimization algorithm that is prone to getting stuck in local optima, and the network parameters are greatly affected by the characteristics of the samples.

Two reasons can be given for the advantage of our method. The first is the contribution of the MSHR on resampling datasets, and the second is that the FCSSVM seeks a fine cost-weight balance between the two classes to achieve good comprehensive performance.

4.3.4 Comparison of the RIME Algorithm and the Particle Swarm Optimization (PSO)

Taking the pageblocks dataset with the highest IR as an example, the optimization performances of the RIME and the PSO on the cost weights of the FCSSVM were explored. The (a) and (b) of Fig. 5 show the change processes of the fitness value, respectively, which is set as the arithmetic mean of the introduced five metric values given by the optimized FCSSVM on the validation dataset, and 50 iterations were recorded. To make the comparison fair, these two algorithms are all run 5 times.

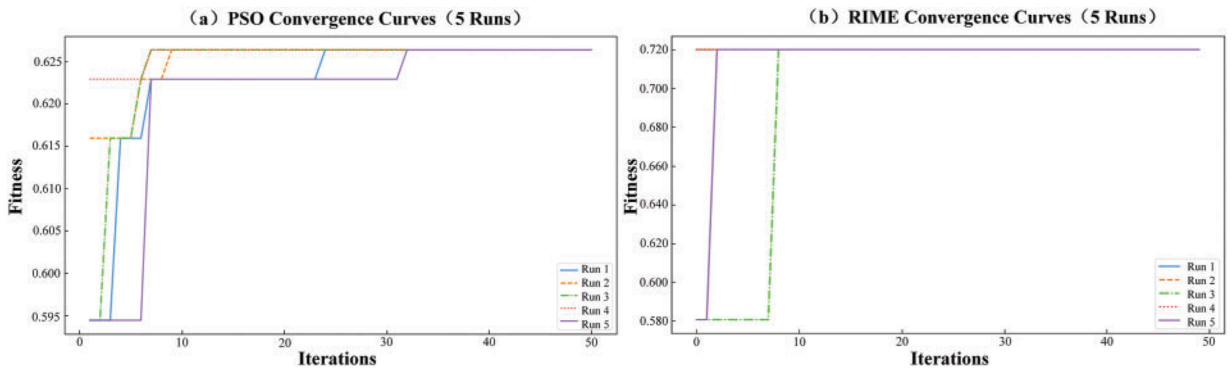

Figure 5: Comparison of the optimization process of RIME and PSO

Five runs of the RIME all achieved a higher fitness value (0.723) in no more than 10 iterations, while those of the PSO only got 0.627. In addition, the RIME only takes 48.17 s for 50 iterations as the PSO takes 192.34 s. That means the RIME takes an advantage over the PSO on both optimization results and efficiency.

5 Conclusion and Future Works

This article proposes an imbalanced data classification method consisting of a new hybrid resampling method MSHR and a fine cost-sensitive SVM FCSSVM. Among them, the MSHR measures sample separability based on the sample Silhouette value calculated by Mahalanobis distance to identify pseudo-negative samples, and based on which to generate new positive samples (over-sampling), and finally removes the pseudo-negative samples (under-sampling). The MSHR not only alleviates the imbalance in the sample number of the two classes, but also to some extent suppresses the inter-class overlapping, making class boundaries cleaner, and thus improving classification performance. The FCSSVM is different from traditional cost-sensitive models which only consider sample number imbalance and directly obtain cost weights based on the IR value. It also considers the impact of class distribution on classification performance by finely optimizing the cost weight parameters based on a fitness value reflecting the validation accuracy, so it can improve the classification performance further. The experimental results show that the proposed method can achieve better classification performance on both mildly and extremely imbalanced datasets, and the application of the RIME optimization algorithm also plays an important role in it. Our future work will focus on how to better optimize the threshold of sample Silhouette value for discriminating pseudo-negative samples and the cost weight values of classifiers, especially improvement of the optimization efficiency.

Acknowledgement: We thank the editors and all anonymous reviewers for their helpful comments. In addition, we thank our friends for their contributions to this article.

Funding Statement: This study was financially supported by the Yunnan Major Scientific and Technological Projects (Grant No. 202302AD080001) and the National Natural Science Foundation, China (No. 52065033).

Author Contributions: The authors confirm contribution to the article as follows: Study conception and design: Bo Zhu, Xiaona Jing; data collection: Xiaona Jing, Lan Qiu; analysis and interpretation of results: Bo Zhu, Xiaona Jing; draft manuscript preparation: Xiaona Jing, Runbo Li. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Readers can access the UCI database (https://archive.ics.uci.edu/datasets) and the Keel database (https://sci2s.ugr.es/keel/datasets.php) used in the article.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] A. Somasundaram and S. Reddy, "Parallel and incremental credit card fraud detection model to handle concept drift and data imbalance," *Neural Comput. Appl.*, vol. 31, no. 1, pp. 3–14, Jan. 2019. doi: 10.1007/s00521-018-3633-8.

[2] M. Amini, J. Rezaeenour, and E. Hadavandi, "A cluster-based data balancing ensemble classifier for response modeling in bank direct marketing," *Int. J. Comput. Intell.*, vol. 14, no. 4, pp. 1550022, Dec. 2015. doi: 10.1142/S1469026815500224.

[3] M. Rezaei, H. Yang, and C. Meinel, "Recurrent generative adversarial network for learning imbalanced medical image semantic segmentation," *Multimed.*, vol. 79, no. 79, pp. 1–20, Feb. 2019. doi: 10.1007/s11042-019-7305-1.

[4] C. Huang *et al.*, "Sample imbalance disease classification model based on association rule feature selection," *Pattern Recognit. Lett.*, vol. 133, no. 16, pp. 280–286, May 2020. doi: 10.1016/j.patrec.2020.03.016.

[5] H. Du, Z. Lin, and Z. Yan, "Network intrusion detection based on selective ensemble learning," presented at the 2019 Int. Conf. Netw. Netw. Appl. (NaNA), Daegu, Korea (South), Oct. 10–13, 2019. doi: 10.1109/NaNA.2019.00038.

[6] S. S. Ranjan and B. B. Gupta, "Classification of spammer and nonspammer content in online social network using genetic algorithm-based feature selection," *Enterp. Inf. Syst.*, vol. 14, no. 5, pp. 710–736, May 2020. doi: 10.1080/17517575.2020.1712742.

[7] L. Taehyung, L. K. Bum, and K. C. Ouk, "Performance of machine learning algorithms for class-imbalanced process fault detection problems," *IEEE Trans. Semicond. Manuf.*, vol. 29, no. 4, pp. 436–445, Nov. 2016. doi: 10.1109/tsm.2016.2602226.

[8] R. Ferreira, J. Barroso, and V. Filipe, "Conformity assessment of informative labels in car engine compartment with deep learning models," *J. Phys.: Conf. Ser.*, vol. 2278, pp. 012033, Feb. 25, 2022. doi: 10.1088/1742-6596/2278/1/012033.

[9] A. Mahadevan and M. Arock, "A class imbalance-aware review rating prediction using hybrid sampling and ensemble learning," *Multimed.*, vol. 80, no. 5, pp. 1–28, Oct. 2020. doi: 10.1007/s11042-020-10024-2.

[10] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jan. 2002. doi: 10.1613/jair.953.

[11] H. Han, W. Y. Wang, and B. H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," presented at the 2005 ICIC, Hefei, China, Aug. 23–26, 2005. doi: 10.1007/11538059_91.

[12] H. He, Y. Bai, E. A. Garcia, and L. Shutao, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE Int. Joint Conf. Neural Netw. (IEEE World Cong. Comput. Intell.)*, Hong Kong, China, Jun. 1–6, 2008, pp. 1322–1328.

[13] G. Goel, L. Maguire, Y. Li, and S. McLoone, "Evaluation of sampling methods for learning from imbalanced data," presented at the 9th ICIC, Nanning, China, Jul. 28–31, 2013.

[14] A. Agrawal, H. L. Viktor, and E. Paquet, "SCUT: Multi-class imbalanced data classification using SMOTE and cluster-based undersampling," presented at the 2015 7th Int. Joint Conf. Knowl. Discov., Knowl. Eng. Knowl. Manage. (IC3K), Lisbon, Portugal, Nov. 12–14, 2015.

[15] G. E. Batista, A. L. C. Bazzan, and M. C. Monard, "Balancing training data for automated annotation of keywords: A case study," *Wob*, vol. 3, pp. 10–18, 2003.

[16] A. Sharma, P. K. Singh, and R. Chandra, "SMOTified-GAN for class imbalanced pattern classification problems," *IEEE Access*, vol. 10, pp. 30655–30665, Mar. 2022. doi: 10.1109/ACCESS.2022.3158977.

[17] G. L. Alberto *et al.*, "A sampling method of imbalanced data based on sample space," (in Chinese), *Acta Autom. Sinica*, vol. 48, no. 10, pp. 2549–2563, Feb. 2023.

[18] J. Wan, M. Yang, and Y. Chen, "Cost sensitive semi-supervised Laplacian support vector machine," *Acta Electonica Sinica*, vol. 40, no. 7, pp. 1410, Jul. 2012. doi: 10.3969/j.issn.0372-2112.2012.07.020.

[19] A. Iranmehr, H. Masnadi-Shirazi, and N. Vasconcelos, "Cost-sensitive support vector machines," *Neurocomput.*, vol. 343, pp. 50–64, May 2019. doi: 10.1016/j.neucom.2018.11.099.

[20] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd Acm Sigkdd Int. Conf. Knowl. Discov. Data Min.*, San Francisco, California, USA, Aug. 13–17, 2016. doi: 10.1145/2939672.2939785.

[21] X. Tao *et al.*, "Self-adaptive cost weights-based support vector machine cost-sensitive ensemble for imbalanced data classification," *Inf. Sci.*, vol. 487, pp. 31–56, Jun. 2019. doi: 10.1016/j.ins.2019.02.062.

[22] Z. Y. Wu, W. F. Lin, and Y. Ji, "An integrated ensemble learning model for imbalanced fault diagnostics and prognostics," *IEEE Access*, vol. 6, pp. 8394–8402, Jan. 2018. doi: 10.1109/ACCESS.2018.2807121.

[23] H. Liu, "Spam web detection based on hybrid-sampling and genetic algorithm," (in Chinese), *J. Beijing Univ. Posts Telecommun.*, vol. 42, no. 6, pp. 111–117, Jun. 2020.

[24] A. Khattak, P. W. Chan, F. Chen, H. R. Peng, and C. M. Matara, "Missed approach, a safety-critical go-around procedure in aviation: Prediction based on machine learning-ensemble imbalance learning," *Adv. Meteorol.*, vol. 2023, pp. 1–24, Jul. 2023. doi: 10.1155/2023/9119521.

[25] Z. X. Zhao, G. L. Wang, and X. D. Li, "An improved SVM based under-sampling method for classifying imbalanced data," (in Chinese), *Zhongshan Daxue Xuebao/Acta Scientiarum Natralium Universitatis Sunyatseni*, vol. 51, no. 6, pp. 10–16, Feb. 2012.

[26] H. Hussein, S. Anwar, and M. Ahmad, "Imbalanced data classification using SVM based on improved simulated annealing featuring synthetic data generation and reduction," *Comput. Mater. Contin.*, vol. 75, no. 1, pp. 547–564, Feb. 2023. doi: 10.32604/CMC.2023.036025.

[27] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, pp. 53–65, 1987. doi: 10.1016/0377-0427(87)90125-7.

[28] P. C. Mahalanobis, "On the generalized distance in statistics," in *Proc. Nat. Inst. Sci.*, Apr. 1936, vol. 2, no. 1, pp. 49–55.

[29] C. C. Chen, Y. H. Lin, J. M. Yih, and S. Y. Juan, "Construct concept structure for linear algebra based on cognition diagnosis and clustering with mahalanobis distances," *Adv. Mat. Res.*, vol. 1220, no. 211–212, pp. 756–760, Apr. 2011. doi: 10.4028/www.scientific.net/AMR.211-212.756.

[30] Y. Wang, Q. Miao, T. Duan, K. L. Tsui, and M. G. Pecht, "Online anomaly detection for hard disk drives based on mahalanobis distance," *IEEETR*, vol. 62, no. 1, pp. 136–145, Jan. 2013. doi: 10.1109/TR.2013.2241204.

[31] X. J. Peng and D. Xu, "Twin mahalanobis distance-based support vector machines for pattern recognition," *Inf. Sci.*, vol. 200, no. 10–12, pp. 22–37, Jun. 2012. doi: 10.1016/j.ins.2012.02.047.

[32] L. Yao and T. B. Lin, "Evolutionary mahalanobis distance-based oversampling for multi-class imbalanced data classification," *Sens.*, vol. 21, no. 19, pp. 6616, Feb. 2022. doi: 10.3390/s21196616.

[33] N. G. Siddappa and T. Kampalappa, "Imbalance data classification using local mahalanobis distance learning based on nearest neighbor," *SN Comput. Sci.*, vol. 1, no. 9, pp. 168–183, Mar. 2020. doi: 10.1007/s42979-020-0085-x.

[34] N. Liu, J. Shen, M. Xu, D. Gan, Q. E. Shi and B. Gao, "Improved cost-sensitive support vector machine classifier for breast cancer diagnosis," *Math. Probl.*, vol. 2018, pp. 1–13, Dec. 2018. doi: 10.1155/2018/3518959.

[35] F. Xia, Y. W. Yang, L. Zhou, F. X. Li, M. Cai and D. D. Zeng, "A closed-form reduction of multi-class cost-sensitive learning to weighted multi-class learning," *Pattern Recognit.*, vol. 42, no. 7, pp. 1572–1581, Jun. 2008. doi: 10.1016/j.patcog.2008.12.011.

[36] H. Su *et al.*, "RIME: A physics-based optimization," *Neurocomput.*, vol. 532, pp. 183–214, Mar. 2023. doi: 10.1016/j.neucom.2023.02.010.

[37] N. Noorhalim, A. Ali, and S. M. Shamsuddin, "Handling imbalanced ratio for class imbalance problem using SMOTE," presented at the Proc. Third Int. Conf. Comput., Math. Stat. (iCMS2017), Langkawi, Malaysia, Nov. 2017. doi: 10.1007/978-981-13-7279-7_3.

[38] C. Wang, Y. F. Liu, X. C. Wang, and G. R. Yan, "Appraisal identification of classifier's performance," (in Chinese), *Electron. Design Eng.*, vol. 19, no. 8, pp. 13–15, Aug. 2011. doi: 10.3969/j.issn.1674-6236.2011.08.004.

[39] M. Li, A. Xiong, L. Wang, S. B. Deng, and J. Ye, "ACO resampling: Enhancing the performance of oversampling methods for class imbalance classification," *Knowl.-Based Syst.*, vol. 196, pp. 105818, May 2020. doi: 10.1016/j.knosys.2020.105818.

[40] N. Zhou, L. Lau, R. Bai, and T. Moore, "A genetic optimization resampling based particle filtering algorithm for indoor target tracking," *Remote Sens.*, vol. 13, no. 1, pp. 132, Jan. 2021. doi: 10.3390/rs13010132.