AIMS Mathematics, 9(7): 17504–17530.
DOI: 10.3934/math.2024851
Received: 07 March 2024
Revised: 23 April 2024
Accepted: 24 April 2024
Published: 21 May 2024

https://www.aimspress.com/journal/Math

Research article

MSFSS: A whale optimization-based multiple sampling feature selection stacking ensemble algorithm for classifying imbalanced data

Shuxiang Wang[1], Changbin Shao[1,2], Sen Xu[3], Xibei Yang[1] and Hualong Yu[1,*]

[1] School of Computer, Jiangsu University of Science and Technology, Zhenjiang, Jiangsu, China
[2] Jiangsu Key Laboratory of Media Design and Software Technology, Jiangnan University, Wuxi, Jiangsu, China
[3] School of Information Engineering, Yancheng Institute of Technology, Yancheng, Jiangsu, China

* **Correspondence:** Email: yuhualong@just.edu.cn; Tel: +86-15952894360.

Abstract: Learning from imbalanced data is a challenging task in the machine learning field, as with this type of data, many traditional supervised learning algorithms tend to focus more on the majority class while damaging the interests of the minority class. Stacking ensemble, which formulates an ensemble by using a meta-learner to combine the predictions of multiple base classifiers, has been used for solving class imbalance learning issues. Specifically, in the context of class imbalance learning, a stacking ensemble learning algorithm is generally considered to combine with a specific sampling algorithm. Such an operation, however, might suffer from suboptimization problems as only using a sampling strategy may make it difficult to acquire diverse enough features. In addition, we also note that using all of these features may damage the meta-learner as there may exist noisy and redundant features. To address these problems, we have proposed a novel stacking ensemble learning algorithm named MSFSS, which divides the learning procedure into two phases. The first stage combined multiple sampling algorithms and multiple supervised learning approaches to construct meta feature space by means of cross combination. The adoption of this strategy satisfied the diversity of the stacking ensemble. The second phase adopted the whale optimization algorithm (WOA) to select the optimal sub-feature combination from the meta feature space, which further improved the quality of the features. Finally, a linear regression classifier was trained as the meta learner to conduct the final prediction. Experimental results on 40 benchmarked imbalanced datasets showed that the proposed MSFSS algorithm significantly outperformed several popular and state-of-the-art class imbalance ensemble learning algorithms. Specifically, the MSFSS acquired the best results in terms of the F-measure metric on 27 datasets and the best results in terms of the G-mean metric on 26 datasets, out of 40 datasets. Although

it required consuming more time than several other competitors, the increment of the running time was acceptable. The experimental results indicated the effectiveness and superiority of the proposed MSFSS algorithm.

Keywords: whale optimization algorithm; stacking ensemble; imbalanced data classification; sampling; feature selection; meta learning
Mathematics Subject Classification: 68T20

1. Introduction

In recent years, class imbalance learning has gradually developed to be one of the hotspots in the machine learning field and attracts a wide range of attention from researchers [1,2]. The class imbalance learning issue is widely concerning due to two main reasons as follows: (1) with unevenly distributed data, many traditional supervised learning algorithms produce prejudiced results as they always tend to focus more on the majority class and damage the interests of the minority class; (2) there are many real-world applications associated with imbalanced data, including object detection [2], disease diagnosis [3,4], intrusion detection [5], image segmentation [6], text classification [7,8], fault diagnosis [9,10], soil classification [11], air quality prediction [12], and bioinformatics [13–15], etc.

Many methods have been proposed to address class imbalance learning issues, and in general, they are categorized into three groups: data level [16–21], algorithm level [22–25], and ensemble learning [26–31].

Data level, which is also called sampling, reconstructs the data distribution to make the dataset be balanced by either adding the minority instances or removing the majority instances. Specifically, the data-level technique can be seen as a pre-processing solution. Such a technique generally owns two significant advantages as follows: 1) it is easy to implement, and 2) it does not require modifying the complex algorithm structure. However, such a technique alters the original data distribution, which brings a potential risk for the subsequent data modeling.

Algorithm level modifies learning algorithms to fit imbalanced data distributions. This technique could be divided into two groups: cost-sensitive learning methods [22,23] and threshold-moving strategies [24,25]. The former modifies the loss function of the learning algorithms by assigning a higher penalty for the training loss of minority instances, while the latter is a post-processing solution which first directly trains a traditional classifier on skewed data, and then provides positive compensation for the outputs of each minority instance. The merits of such a technique reflect that it can not only maintain the original data distribution, but also tends to acquire more stable classification performance than the data-level strategies. However, it generally requires conducting a complex optimization procedure for acquiring an appropriate penalty weight or compensation threshold.

As for ensemble learning, this method combines multiple classifiers to promote the robustness of class imbalance learning. In particular, the ensemble learning integrates classifiers in various ways, such as bagging [26,27], boosting [28] and stacking [29–31]. It is also noted that ensemble learning itself cannot directly solve a class imbalance learning issue, but when each base classifier adopts a data-level or algorithm-level solution, the issue can be effectively addressed. In comparison to data-level and algorithm-level techniques, ensemble learning always tends to produce more robust classification results. As a meta-learning approach, stacking owns a specific merit in comparison to two

other ensemble learning paradigms, bagging and boosting. That is, it tries to learn how to learn, thus often produces better and/or more robust classification performance than the two others on the same learning tasks [32]. Therefore, we have focused on how to use stacking technique for improving classification performance on imbalanced data in this study.

Specifically, the stacking technique maintains a two-layer structure in which the first layer lays out different classification algorithms, and the second layer first captures the output of each classifier in the first layer to constitute a new feature space, and then trains a new classifier to generate the final predictions. Here, the classifier in the second layer is generally called a meta-learner that is capable of simultaneously capturing the characteristics of various different classifiers, thus the classification performance can be materially improved.

To address the class imbalance problem by the stacking ensemble, a data level or algorithm level technique is generally used on its first layer. The state-of-the-art imbalanced stacking algorithms [29–31] always adopt a specific sampling algorithm to combine with several different supervised learning algorithms to train base learners. Such an operation, however, tends to suffer from suboptimization problems as although those base learners use different supervised learning algorithms, they run on the same training instances. In other words, an excellent ensemble learning model requires diverse enough base learners, while only adopting different learning algorithms does not guarantee that it will maximize the diversity among the base learners. In addition, we all know ensemble learning requires high-quality base learners, and stacking is no exception. We note that in a meta feature space constituted by outputs of all base learners, there might exist poor and redundant features which would damage the meta-learner constructed on the second layer of stacking.

Motivated by the two issues described above, we wish to propose an improved stacking ensemble learning algorithm for classifying imbalanced data. First, we consider to further promote the diversity among base learners when we construct the first layer of stacking. As indicated in [29–31], all emerging class imbalance stacking ensemble learning algorithms inherit the idea of original stacking, that is, producing diverse base learners only by calling different learning algorithms. However, the diversity yielded by such a strategy is limited due to the fact that all learning algorithms are modeled on the same or quite similar training data. To further increase the diversity, we propose to combine multiple different sampling strategies and various learning algorithms to produce base learners.

As for the second issue, we consider to introduce a feature selection procedure on the meta feature space when we construct the second layer of stacking. Obviously, the feature selection will be helpful for improving the quality of the meta feature space. Here, we wish to adopt a metaheuristic algorithm to implement a wrapper-based feature selection procedure. Specifically, the metaheuristic algorithms are a type of optimization technique inspired by natural or social phenomena used to solve various complex optimization problems. This kind of algorithm combines randomness and strategic search to find approximately optimal solutions within an acceptable time. Several metaheuristic algorithms have been widely used to solve real-world complex optimization problems, including particle swarm optimization (PSO) [33], which simulates the foraging behavior of bird flocks, differential evolution (DE) [34], which simulates biological evolution, and social spider optimization (SSO) [35], which simulates the collaboration and information sharing mechanisms of spiders during predation and web weaving processes. As a recently proposed metaheuristic algorithm, the whale optimization algorithm (WOA) [36] has attracted wide attention due to its three specific merits, i.e., strong global searching ability, a few parameter settings, and fast convergence speed. The WOA has also been applied to solve different real-world optimization problems, including estimating the reopening policies relating to

COVID-19 [37], solving complex power flow optimization problems [38], and selecting the optimal gene group for classifying microarray data [39], etc. Considering the merits owned by the WOA, we have decided to use it to implement a wrapper feature selection procedure on the meta feature space of stacking.

Specifically, we integrated the two modules discussed above to propose a novel stacking algorithm called the whale optimization-based multiple sampling feature selection stacking ensemble (MSFSS) for solving the classification problem of imbalanced data. We compared the proposed algorithm with a baseline and several state-of-the-art algorithms on 40 benchmarked class imbalance datasets, and the experimental results show that the MSFSS significantly outperforms several other competitors. In particular, the MSFSS acquires the best results in term of the F-measure metric on 27 datasets and the best results in term of the G-mean metric on 26 datasets, out of 40 datasets. Although the proposed algorithm requires consuming more time than the compared algorithms, the increment of running time seems to be acceptable. The experimental results indicate the effectiveness and superiority of the proposed MSFSS algorithm.

The novelties and contributions of this study can be easily concluded as follows:

1) To guarantee to produce a sufficient diversity among base learners of stacking, further to generate diverse enough meta features to improve the quality of stacking ensemble, a novel strategy using multiple sampling approaches in combination with multiple learning algorithms is proposed;
2) To improve the quality of the meta learner, a WOA-based feature selection procedure is introduced, which guarantees to sufficiently remove those irrelevant and redundant features from the original meta feature space.

The remainder of this paper is organized as follows. Section 2 firstly explains what class imbalance problem is and analyzes its reasons, and then briefly describes the related work about class imbalance ensemble learning techniques. Section 3 describes the procedure of the proposed MSFSS algorithm in detail. The experimental settings, results, and discussions are presented in Section 4. Finally, Section 5 concludes the contributions of this study.

2. Preliminary and related work

2.1. Class imbalance issue

As indicated in Section 1, the class imbalance issue, which is also called the imbalanced data distribution issue, widely emerges in various real-world applications. For a given dataset, if the instances belonging to a class are much more than the number of instances in the other class, it can be called a class imbalance dataset. As we know, most traditional supervised learning algorithms abide by the empirical risk minimization rule, i.e., the training optimization aims at:

$$f^* = \underset{f \in F}{\operatorname{argmin}} \sum_{i=1}^{N} L(f(x_i), y_i) \tag{1}$$

where x_i and y_i respectively denote the ith training instance and the corresponding training label, N indicates the number of training instances, L represents a specific loss function, and F denotes the candidate function family, then the optimization problem is indeed searching the optimal f^* from F to make the sum of losses be minimal. Based on Eq (1), it is not difficult to observe that when the number of instances belonging to a class overwhelms that of the other class, the majority class obviously contributes more to the total loss. Therefore, the traditional supervised learning algorithms are apt to focus more on the majority class, but damage the interests of the minority class. However, that does

not mean that an imbalanced data distribution certainly makes traditional supervised learning algorithms lose efficacy. When there exists a clear boundary between two classes, the harmfulness of imbalanced data distribution can be almost ignored (see Figure 1(a)). Several previous works [40,41] illustrate that when there exists a severe class overlap (see Figure 1(b)) or a lot of noisy instances (see Figure 1(c)), class imbalance issues would be highlighted. In other words, the skewed data distribution itself is not a problem, but when some other complex distribution factors emerge, it would play a role in destroying the performance of the minority class.

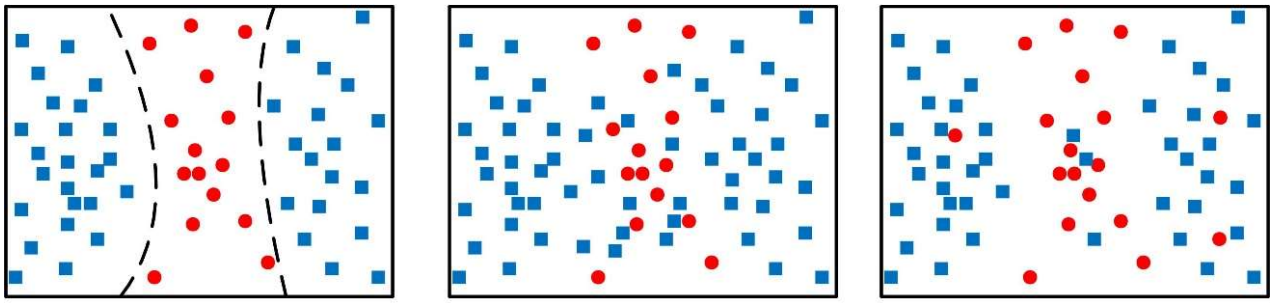


Figure 1. Imbalanced data distribution with (a) a clear boundary, (b) class overlap, and (c) noisy instances.

2.2. Related work

In Section 1, it has been indicated that class imbalance learning generally adopts one of three of the following strategies: data-level approaches, algorithm-level methods, and ensemble learning algorithms. In this study, we pay close attention to the ensemble learning technique as it is always more robust than the other two.

In the context of classifying imbalanced data, ensemble learning algorithms integrate either a data-level technique or algorithm-level strategy into an ensemble paradigm, such as bagging [42], boosting [43] and stacking [32]. Specifically, bagging oversamples or undersamples instances to construct multiple different balanced training subsets, then trains a classifier on each subset, and finally adopts majority voting to predict class labels of test instances. BalancedBagging [26], which is a popular class imbalance ensemble learning algorithm, uses a random undersampling (RUS) technique on majority instances for multiple iterations in which each undersamples the majority instances to combine with all minority instances for constituting a balanced training subset that is sequentially used to train the base learner. SMOTEBagging [27] adopts a similar process as BalancedBagging, but only replaces the RUS with the SMOTE [16] oversampling algorithm.

The boosting paradigm takes advantage of the feedback of the classification results to dynamically increase the weights of misclassified instances, further trains a new base classifier, and finally predicts the class labels of test instances by a weighted voting rule. Unlike the parallel generation method adopted by bagging, boosting generates base learners in a serial fashion. RUSBoost [28] is a popular boosting algorithm which is specifically designed to address the class imbalance learning problem. It first randomly undersamples majority instances to acquire a balanced training set, and then calls on the boosting algorithm [43] to improve the classification performance.

As for stacking [32], it adopts a totally different method from bagging and boosting to integrate the outputs of base learners. It first arranges all outputs generated by base learners in order to constitute a new feature space that is called the meta feature space. In this space, each feature simultaneously reflects the characteristics of the corresponding instance and the corresponding base learner. Then, a new meta classifier is further trained on the meta feature space to make the final decision. Specifically, stacking adopts a two-layer structure, and meanwhile, it requires using diverse learning algorithms to guarantee the diversity in the meta feature space.

In the context of class imbalance learning, several stacking ensemble learning algorithms have been developed, including the stable and interpretable rule set (SIRUS) [29], supervised adaptive discriminant analysis (SADA) [30], and neighborhood undersampling stacked ensemble (NUS-SE) [31]. In particular, the SIRUS algorithm first adopts the inverse random undersampling (IRUS) approach [20] to transform the original imbalanced training set to be a balanced one, and then trains different base learners on those transformed training sets to constitute the meta feature space. Similar to SIRUS [29], both SADA [30] and NUS-SE [31] only replace IRUS adopted by SIRUS with the adaptive synthetic (ADASYN) undersampling strategy [17] and a neighborhood density-based undersampling strategy, respectively. We note that these emerging stacking ensemble learning algorithms present a common characteristic, i.e., the diversity of the ensemble completely relies on the adoption of multiple different supervised learning algorithms, but neglects the diversity at the data level, thus they may suffer from a suboptimization problem. In addition, we also note that all of these algorithms directly use all of the meta features to train the meta learner whose quality might be decreased by some poor and redundant meta features. Therefore, it is necessary to develop new imbalance stacking ensemble algorithms to effectively address these two problems. This also motivates our research in this study.

3. Methods

3.1. First-layer improving ensemble diversity by combining multiple sampling approaches and diverse supervised learning algorithms

As we know, to guarantee an ensemble learning algorithm will outperform its individual members, it requires satisfying a sufficient and necessary condition, i.e., each base learner should be accurate and meanwhile, all base learners should be diverse enough [44]. The principle can be described as follows:

$$E = \bar{E} - \bar{P} \tag{2}$$

where E denotes the generalization error of the ensemble learner, $\bar{E}$ and $\bar{P}$ represent the average generalization error and the average diversity of all base learners, respectively. To produce successful ensemble learning models, these two aspects should be considered simultaneously.

Stacking [32] and its several variants [29–31], which are specifically designed for classifying imbalanced data, acquire diversity only by adopting different supervised learning algorithms, by training them on the same or similar data. Therefore, they may suffer from a suboptimization problem due to the lack of diversity. To further improve the diversity among the base learners/meta features, we propose to combine both different sampling strategies and supervised learning algorithms. Suppose that R and C respectively represent resampling and classification algorithms, and H and K denote the number of them, respectively. Then, there exist $H \times K$ different combinations. That means that the new generated meta feature space is $H \times K$ dimensional (see Table 1).

Table 1. Constitution of the meta feature space.

R \ C	C_1	C_2	…	C_{j-1}	C_j	…	C_K
R_1	(R_1, C_1)	(R_1, C_2)	…	(R_1, C_{j-1})	(R_1, C_j)	…	(R_1, C_K)
R_2	(R_2, C_1)	(R_2, C_2)	…	(R_2, C_{j-1})	(R_2, C_j)	…	(R_2, C_K)
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
R_{i-1}	(R_{i-1}, C_1)	(R_{i-1}, C_2)	…	(R_{i-1}, C_{j-1})	(R_{i-1}, C_j)	…	(R_{i-1}, C_K)
R_i	(R_i, C_1)	(R_i, C_2)	…	(R_i, C_{j-1})	(R_i, C_j)	…	(R_i, C_K)
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
R_H	(R_H, C_1)	(R_H, C_2)	…	(R_H, C_{j-1})	(R_H, C_j)	…	(R_H, C_K)

To avoid the generated meta features being overfitted on training instances, an internal 5-fold cross validation procedure is conducted to sequentially produce meta features for instances belonging to each fold. That means that, every time, the instances belonging to four different folds are first integrated to conduct sampling and classifier modeling procedures, and then the instances belonging to the remainder fold are fed into the trained classifier to further obtain their corresponding meta features. This strategy is also widely adopted by the original stacking algorithm and its various variants [29–32]. When a new test instance arrives, it would be respectively put into five trained classifiers to obtain five different predictions, and then their mean value could be calculated as the corresponding meta feature. Figure 2 takes the combination of R_1 and C_1 as an example to describe the generation procedure of the corresponding meta feature.

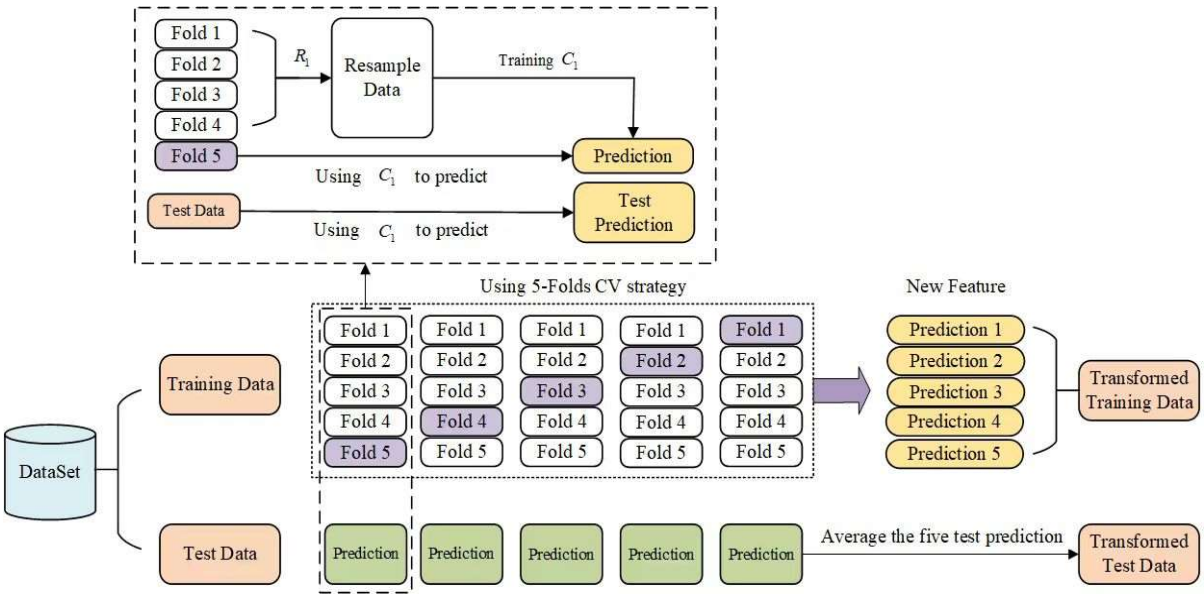


Figure 2. The procedure of generating a meta feature for both training and test data.

In this study, we selected four popular sampling algorithms namely SMOTE [16], ADASYN [17], SMOTE-ENN [18], and SMOTE-Tomek [19], and five different supervised learning algorithms namely K-nearest neighbors (KNN) [45], C4.5 decision tree (C4.5 DT) [46], support vector machine (SVM) [47], Gaussian naïve Bayes (GNB) [48], and linear discriminant analysis (LDA) [49]. Therefore, the number of generated meta features is $4 \times 5 = 20$, and the training procedure requires producing 100 base learners as the adoption of internal 5-fold cross validation. In practical applications, the readers are encouraged to freely replace or extend sampling and supervised learning algorithms.

3.2. Second layer-improving quality of meta features by whale optimization feature selection strategy

As indicated in Eq (2), the quality of an ensemble learning algorithm simultaneously depends on the diversity among base learners and the quality of each base learner. However, we cannot guarantee that the meta feature space is excellent enough, as in this space, some features might be poor or redundant. Therefore, it does not suggest to directly train a meta learner on all meta features. A feature selection procedure may be required to remove those poor and redundant meta features.

In this study, we consider to adopt the whale optimization algorithm (WOA) [36–39] to implement a procedure of wrapper feature selection. Specifically, the WOA, which simulates the hunting behavior of humpback whales, is a recently proposed nature-inspired swarm optimization algorithm. According to some recent studies, in comparison with some traditional swarm optimization algorithms, the WOA has presented several specific merits as follows: (1) stronger global searching ability, (2) less parameter settings, and (3) faster convergence speed [36–39]. Also, we note that the WOA algorithm could be used to solve both continuous and discrete optimization problems, thus in this study, we select to use the WOA algorithm to conduct the feature selection task in the meta feature space.

Specifically, the WOA contains two phases, namely exploitation and exploration, respectively. The former phase takes charge of the convergence of the algorithm, while the latter phase is used to avoid the algorithm being overfitted. Suppose X is the current solution of a whale, then the mathematical model corresponding to the movement of the whale around a prey can be represented as:

$$\vec{D} = |\vec{F} \cdot \overrightarrow{X^*}(t) - \vec{X}(t)| \tag{3}$$

$$\vec{X}(t+1) = \overrightarrow{X^*}(t) - \vec{A} \cdot \vec{D} \tag{4}$$

where t denotes the current iteration, X^* represents the best solution obtained so far, $|\ |$ indicates the absolute value, and $\cdot$ represents element-by-element multiplication. A and F are coefficient vectors which are respectively calculated by the following equations:

$$\vec{A} = 2\vec{a} \cdot \vec{r} - \vec{a} \tag{5}$$

$$\vec{F} = 2 \cdot \vec{r} \tag{6}$$

where $\vec{a}$ specifies a vector constructed by multiple same elements a which gradually conduct a linear decrease from 2 to 0, and $\vec{r}$ denotes a random vector in [0,1]. In particular, the variations of both A and C vectors control the areas where a solution can be located in the neighborhood of the best solution.

The humpback whales move in a shrinking encircling mechanism and along a spiral-shaped path toward the prey. In the WOA, the shrinking encircling behavior is simulated by decreasing the value of a in Eq (5) according to Eq (6),

$$a = 1 - \frac{2t}{MaxI} \tag{7}$$

where $MaxI$ denotes the designated number of iterations in the WOA. The spiral-shaped path is achieved by calculating the distance between the current solution X and the current swarm best solution X^*. Then a spiral equation could be created as

$$\vec{X}(t+1) = D' \cdot e^{bl} \cdot \cos(2\pi l) + \overrightarrow{X^*}(t) \tag{8}$$

where $D' = |\overrightarrow{X^*}(t) - \vec{X}(t)|$, b defines the spiral's shape of the spiral, and l represents a random number in [-1, 1]. During optimization, the shrinking encircling and upwarding spiral-shaped path are respectively given a 50% chance to be conducted as

$$\vec{X}(t+1) = \begin{cases} \overrightarrow{X^*}(t) - \vec{A} \cdot \vec{D}, & if\ p < 0.5 \\ D' \cdot e^{bl} \cdot \cos(2\pi l) + \overrightarrow{X^*}(t), & if\ p \geq 0.5 \end{cases} \tag{9}$$

where p represents a random number in [0, 1].

In addition, to avoid the population dropping into the local optimization area, a few whales are randomly selected at each iteration to conduct random movements as follows:

$$\vec{D} = |\vec{F} \cdot \overrightarrow{X_r} - \vec{X}| \tag{10}$$

$$\vec{X}(t+1) = \overrightarrow{X_r} - \vec{A} \cdot \vec{D} \tag{11}$$

where $\overrightarrow{X_r}$ denotes a randomly chosen whale from the current whale population.

The procedure of the WOA algorithm can be simply described as below.

WOA Algorithm

Input: size of population M, the number of iterations $MaxI$
Output: the optimized solution X^* found by the WOA
Procedure:
Generate the initial population $X_i, i = 1,2, \dots, M$;
Calculate the fitness value for each whale X_i;
X^*=the current best solution;
for j=1 to $MaxI$
  for i = 1 to M
    Update a, A, F, l, and p;
    if $p < 0.5$
      if $|A|<1$
        Use Eq (4) to update the position of $X_i(j)$;

else

Select randomly a whale $\overrightarrow{X_r}$ and update the position of $X_i(j)$ by Eq (11);

end *if*

else

Use Eq (8) to update the position of $X_i(j)$;

end *if*

Calculate the fitness value for $X_i(j)$;

end *for*

Update X^*;

end *for*

Return X^*

In this study, M, which denotes the size of the population in the WOA, is designated as 50, and the number of iterations, $MaxI$, is empirically set to be 200. As for the fitness function of the WOA, it is defined as follows:

$$fit(\vec{X}) = \frac{1}{2}F - measure + \frac{1}{2}G - mean \tag{12}$$

where F-measure and G-mean are both popular performance metrics for evaluating supervised learning algorithms conducted on imbalanced data, which will be further described in detail in Section 4.

In the original $H \times K$ dimensional meta feature space, the WOA is adopted to extract n features ($n < H \times K$), and then a linear regression classifier [50] is trained on it to make the final decision. The procedure of WOA-based feature selection is illustrated in Figure 3. Specifically, to avoid overfitting, the WOA also conducts an internal 5-fold cross validation to implement the wrapper feature selection procedure.

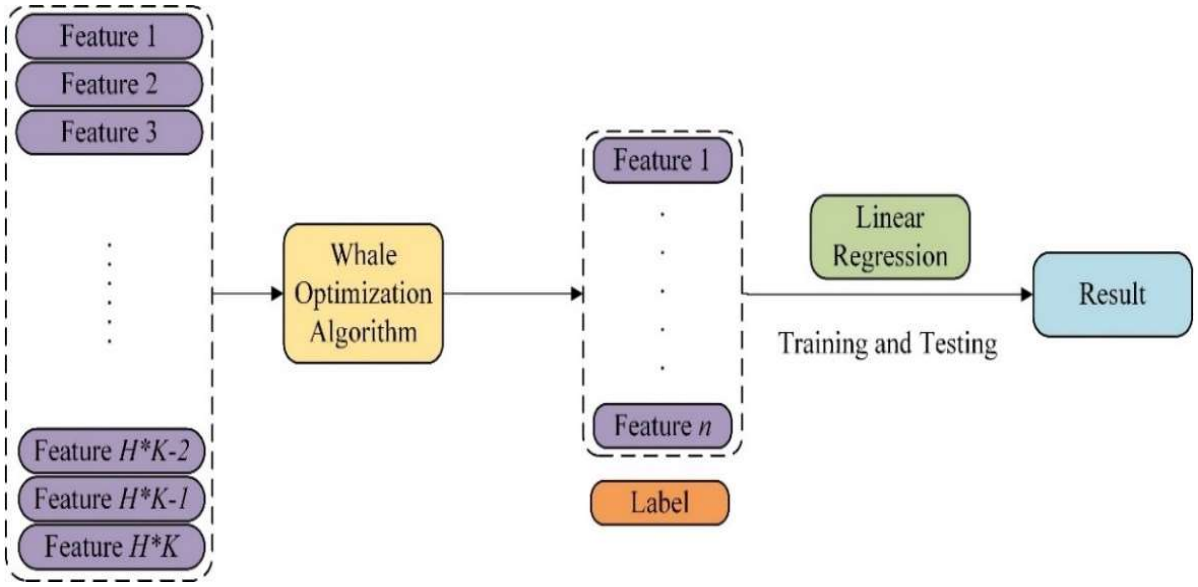


Figure 3. Procedure for selecting features from meta feature space and creating the final meta learner.

3.3. MSFSS algorithm

By integrating *multiple sampling meta feature generation* and *WOA-based meta feature selection* procedures, we propose a novel imbalance stacking ensemble learning algorithm called MSFSS, which is described below.

MSFSS Algorithm

Input: a binary-class imbalanced data set Ψ, the number of whales M, the number of iterations $MaxI$, sampling group $\{R_1, R_2, \dots, R_H\}$, algorithm group $\{C_1, C_2, \dots, C_K\}$

Output: prediction label sequence

Training Procedure:

Divide the data set Ψ into the training set Ψ_{train} and the testing set Ψ_{test};

Divide Ψ_{train} into 5 disjoint subsets $\Psi_{train_1}.\Psi_{train_2}, \Psi_{train_3}, \Psi_{train_4}, \Psi_{train_5}$;

for s= 1 to 5

$\Psi'_{train} = \Psi_{train} - \Psi_{train_s}$;

$\Psi'_{test} = \Psi_{train_s}$;

for w = 1 to H

Run R_w on Ψ'_{train} to acquire the corresponding sampling set $\Psi'_{train-samplin}$;

for v = 1 to K

Use C_v on $\Psi'_{train-samp}$ to train the corresponding classifier $C_{s,w,v}$;

Run $C_{s,w,v}$ on Ψ'_{test} to acquire the corresponding meta features;

end *for*

end *for*

end *for*

Use all meta features to integrate a meta training set $\Psi_{meta-trai}$ to replace the original training set Ψ_{train};

Obtain a shrinking meta training set $\Psi'_{meta-train} = \mathrm{WOA}(\Psi_{meta-train}, M, MaxI)$;

Train a linear regression classifier LR on $\Psi_{meta-train}$;

Testing Procedure:

for w = 1 to H

for $v = 1$ to K

for $s = 1$ to 5

Run $C_{s,w,v}$ on Ψ_{test} to acquire $\Psi^s_{w,v}$;

end *for*

Generate the $((w-1)\times K + v)$th meta feature by averaging $\Psi^1_{w,v}$, $\Psi^2_{w,v}$, $\Psi^3_{w,v}$, $\Psi^4_{w,v}$, $\Psi^5_{w,v}$;

end *for*

end *for*

Use all meta features to integrate a meta test set $\Psi_{meta-test}$ to replace the original training set Ψ_{test};

Extract the shrinking meta test set $\Psi'_{meta-test}$ by inquiring the numerical order of features recorded by the WOA;
Call LR to predict the class labels of all instances on $\Psi'_{meta-test}$ and output the corresponding prediction label sequence.

Without loss of generality, we suppose the average time complexity of all sampling algorithms is φ, the average time complexity of all supervised learning algorithms is τ, and the time complexity of training a linear regression meta learner is ω. Then, it is clear that the time complexity of the MSFSS algorithm is $O((\varphi + \tau)H \times K + M \times MaxI \times \omega)$. Here, φ, τ, and ω all rely on the number of training instances N.

4. Experiments

4.1. Datasets descriptions

In this study, we used 40 benchmarked class imbalance datasets to validate the performance of the proposed MSFSS algorithm. Specifically, these benchmark datasets are extracted from either UCI machine learning repository [51] or Keel data repository [52]. Table 2 describes the detailed information about these data sets, including the number of instances, number of features, class imbalance ratio, and data source of each dataset.

Table 2. Descriptions of the datasets used in this study.

ID	Dataset	Number of instances	Number of features	Class imbalance ratio	Data source
1	EEG_Eye_State	14,980	14	1.22	UCI
2	banana	5,299	2	1.23	UCI
3	Australian	690	14	1.24	UCI
4	brainweb_20000	20,000	3	1.32	UCI
5	MovementAAL_RSS	13,197	4	1.34	UCI
6	liver	345	6	1.37	UCI
7	Elec_Partial	10,000	5	1.37	UCI
8	Ionosphere	351	34	1.78	UCI
9	glass1	214	9	1.81	KEEL
10	ecoli-0_vs_1	220	7	1.85	KEEL
11	wisconsin	683	9	1.86	KEEL
12	diabetes	768	8	1.86	UCI
13	iris0	150	4	2.00	KEEL
14	bupa	300	7	2.00	UCI
15	wine	178	13	2.01	UCI
16	titanic	2,201	3	2.09	UCI
17	German	1,000	24	2.33	UCI
18	phoneme	5,404	5	2.40	UCI
19	sober	72	19	2.42	KEEL
20	ILPD	583	11	2.49	UCI

Continued on next page

ID	Dataset	Number of instances	Number of features	Class imbalance ratio	Data source
21	glass-0-1-2-3_vs_4-5-6	214	9	3.19	KEEL
22	blood	748	4	3.20	UCI
23	ecoli1	336	7	3.36	KEEL
24	skin	245,057	3	3.81	UCI
25	usps	1,500	241	4.00	UCI
26	appendicitis	106	8	4.04	UCI
27	new-thyroid1	215	5	5.14	KEEL
28	ecoli2	336	7	5.46	KEEL
29	musk	6,598	166	5.48	UCI
30	segment0	2,308	19	6.01	KEEL
31	glass-3-5_vs_1-2-6-7	214	10	6.13	KEEL
32	glass	214	9	6.37	KEEL
33	page-blocks0	5,472	10	8.78	KEEL
34	ecoli-0-4-6_vs_5	203	7	9.15	KEEL
35	HTRU2	17,898	8	9.92	UCI
36	vowel0	988	13	9.97	UCI
37	ecoli-0-1-4-7_vs_2-3-5-6	336	8	10.58	KEEL
38	ecoli4	336	8	15.80	KEEL
39	dermatology-6	358	34	16.90	KEEL
40	wilt	4,839	5	17.54	UCI

4.2. Experimental settings

4.2.1 Compared algorithms and parameter settings

In this study, we compared the proposed MSFSS algorithm with a baseline algorithm: stacking [32], two representative class imbalance ensemble learning algorithms respectively adopting bagging and boosting paradigms: BalancedBagging [26] and RUSBoost [28], and three popular stacking variants for classifying imbalanced data: SIRUS [29], SADA [30], and NUS-SE [31]. Specifically, all sampling techniques used in these algorithms sampled totally balanced datasets. Both BalancedBagging [26] and RUSBoost [28] used a C4.5 decision tree [46] as the base learner, and the number of base learners was designated as 50. Considering that our proposed MSFSS algorithm requires producing 20 different base learners, thus to guarantee the impartiality of the compared experiments, the stacking [32], SIRUS [29], SADA [30] and NUS-SE [31] algorithms all generate 20 base classifiers, which means that the stacking requires disturbing the feature space four times, while three other algorithms require independently conducting the sampling procedure four times. As for all of the supervised learning algorithms, they adopt the default parameters in scikit-learn 1.3.1[1]. Table 3 presents the details about these compared algorithms, and explains the reasons for choosing them.

[1] https://scikit-learn.org/

Table 3. The details of the compared algorithms.

No	Algorithm	Settings	Reasons for choice
1	BalancedBagging	The base learner uses a C4.5 decision tree, the number of base learners in the ensemble is designated as 50, and the sampling strategy adopts random undersampling (RUS).	It serves as a popular ensemble class imbalance learning algorithm based on bagging paradigm.
2	RUSBoost	The base learner uses a C4.5 decision tree, the number of base learners in the ensemble is designated as 50, and the sampling strategy adopts random undersampling (RUS).	It serves as a popular ensemble class imbalance learning algorithm based on boosting paradigm.
3	Stacking	The base learners use KNN, a C4.5 decision tree, a support vector machine (SVM), Gaussian naïve Bayes, and linear discriminant analysis (LDA), the number of base learners in the ensemble is designated as 20, and no sampling strategy is used.	It serves as a baseline algorithm using stacking paradigm.
4	SIRUS	The base learners use KNN, a C4.5 decision tree, a support vector machine (SVM), Gaussian naïve Bayes, and linear discriminant analysis (LDA), the number of base learners in the ensemble is designated as 20, and the sampling strategy adopts inversed RUS (IRUS).	It serves as a state-of-the-art stacking variant aiming at imbalanced data classification.
5	SADA	The base learners use KNN, a C4.5 decision tree, a support vector machine (SVM), Gaussian naïve Bayes, and linear discriminant analysis (LDA), the number of base learners in the ensemble is designated as 20, and the sampling strategy adopts ADASYN.	It serves as a state-of-the-art stacking variant aiming at imbalanced data classification.
6	NUS-SE	The base learners use KNN, a C4.5 decision tree, a support vector machine (SVM), Gaussian naïve Bayes, and linear discriminant analysis (LDA), the number of base learners in the ensemble is designated as 20, and the sampling strategy adopts neighborhood undersampling.	It serves as a state-of-the-art stacking variant aiming at imbalanced data classification.

4.2.2 Performance evaluation metrics

It is well known that on imbalanced classification tasks, the traditional classification accuracy or classification error rate is not an applicative performance evaluation metric. Therefore, in this study, we adopted two popular metrics, F-measure and G-mean for evaluating the performance of various class imbalance learning algorithms. F-measure tests the tradeoff between two metrics, precision and recall, while G-mean tests the tradeoff between two other metrics, true positive rate (TPR) and true negative rate (TNR). Specifically, these two metrics are calculated as follows.

$$F-measure=\frac{2\times \text{precision}\times \text{recall}}{\text{precision}\times \text{recall}} \tag{13}$$

$$G - mean = \sqrt{\text{TPR} \times \text{TNR}} \tag{14}$$

where precision, recall, TPR and TNR can be further calculated by,

$$\text{precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{15}$$

$$\text{recall} = \text{TPR} = \frac{\text{TP}}{\text{TP+FN}} \tag{16}$$

$$\text{TNR} = \frac{\text{TN}}{\text{TN+FP}} \tag{17}$$

where TP, TN, FP, and FN further rely on the statistics described in Table 4.

In addition, we used external 5-fold cross validation to detect the performance of each algorithm. Considering the randomness of the experiments, each experiment was randomly repeated 50 times, and then the average results were presented.

Table 4. Confusion matrix.

	Predict positive	Predict negative
Actual positive	TP	FN
Actual negative	FN	TN

4.3. Results and discussions

Experimental results are illustrated in Tables 5 and 6. It can be clearly observed that the Stacking-based algorithms obviously outperform to two algorithms based on other ensemble learning paradigms. Specifically, the BalancedBagging and RUSBoost only respectively produce 4 best results on the F-measure metric, and 5 and 3 best results on the G-mean metric, which are significantly less than that produced by stacking series algorithms. We believe that it associates to different mechanisms adopted by these ensemble learning paradigms as both bagging and boosting only integrate the classification results by simple voting methods, but as a meta learning strategy, the stacking tries to learn how to learn.

Table 5. Performance comparison of various algorithms in terms of the F-measure metric.

Dataset	BalancedBagging	RUSBoost	Stacking	SIRUS	SADA	NUS-SE	MSFSS
EEG_Eye_State	0.8843	0.7191	0.9640	0.9623	0.9668	0.9031	**0.9691**
banana	0.8718	0.6922	0.8860	0.8889	0.8835	**0.8897**	0.8787
Australian	**0.8777**	0.8382	0.8686	0.8623	0.8613	0.8639	0.8542
brainweb_20000	0.9421	0.9469	0.9469	0.9491	0.9427	**0.9493**	0.9448
MovementAAL_RSS	0.7440	0.6766	0.7805	0.7648	0.7641	**0.7811**	0.7399
liver	0.6431	0.6223	0.5912	0.6028	0.6143	0.5955	**0.6552**
Elec_Partial	0.6862	0.6658	0.6665	0.6661	0.6646	0.6686	**0.6902**
Ionosphere	0.9565	0.8979	**0.9787**	**0.9787**	**0.9787**	0.9565	**0.9787**
glass1	0.7045	0.6408	0.6991	0.6629	0.6933	0.6485	**0.7380**

Continued on next page

Dataset	BalancedBagging	RUSBoost	Stacking	SIRUS	SADA	NUS-SE	MSFSS
ecoli-0_vs_1	0.9801	0.9736	**0.9862**	0.9797	0.9733	0.9797	**0.9862**
wisconsin	0.9470	0.9143	0.9624	0.9553	0.9625	0.9574	**0.9641**
diabetes	0.6666	0.4285	0.5454	0.6399	0.6274	0.5454	**0.6885**
iris0	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
bupa	0.5852	0.6095	0.6250	0.5046	0.5986	0.2907	**0.6831**
wine	0.9833	0.9833	**0.9956**	0.9873	0.9916	0.9916	**0.9956**
titanic	0.5911	**0.6077**	0.5415	0.6006	0.6006	0.5985	0.6006
German	0.6400	0.6506	0.6101	0.6206	0.5666	0.6451	**0.6842**
phoneme	0.8111	0.7079	0.8112	0.7997	0.8246	0.7971	**0.8312**
sober	0.7453	0.7396	0.7287	0.7488	0.7430	0.6730	**0.8242**
ILPD	**0.5157**	0.5119	0.0849	0.0372	0.0988	0.0300	0.4905
glass-0-1-2-3_vs_4-5-6	0.9038	0.8800	0.8368	0.8431	0.8681	0.8331	**0.9083**
blood	0.4636	0.4394	0.1452	0.0392	0.2759	0.1435	**0.4832**
ecoli1	0.7865	0.7007	0.7930	0.8034	0.7596	0.7644	**0.8070**
skin	0.9981	0.9186	**0.9989**	0.9982	**0.9989**	**0.9989**	**0.9989**
usps	0.7543	0.6404	0.9128	0.9272	0.9198	0.9143	**0.9328**
appendicitis	0.9600	0.9600	**1.0000**	0.9266	0.9800	0.8300	**1.0000**
new-thyroid1	0.9269	0.9099	0.9777	**0.9882**	0.9777	**0.9882**	0.9461
ecoli2	**0.7857**	**0.7857**	**0.7857**	**0.7857**	**0.7857**	**0.7857**	**0.7857**
musk	0.9582	0.9682	0.9884	0.9748	0.9873	0.9875	**0.9899**
segment0	0.9745	0.9834	0.9815	0.9907	0.9894	0.9907	**0.9932**
glass-3-5_vs_1-2-6-7	0.8413	0.8014	0.8978	0.7969	0.8169	0.0000	**0.9314**
glass	0.8359	0.8247	**0.8575**	0.8107	0.8125	0.8103	0.8155
page-blocks0	0.8310	0.6615	**0.8716**	0.7986	0.8489	0.6871	0.8028
ecoli-0-4-6_vs_5	0.9555	0.9777	0.9714	0.9377	**1.0000**	0.9714	0.9777
HTRU2	0.8539	0.7787	0.8695	0.8744	**0.8757**	0.8700	0.8700
vowel0	0.8484	0.8823	0.9706	0.8956	**0.9945**	0.4460	**0.9945**
ecoli-0-1-4-7_vs_2-3-5-6	0.9384	0.9664	0.9777	0.9356	**1.0000**	**1.0000**	0.9818
ecoli4	0.7433	0.7974	**0.9666**	0.9066	**0.9666**	0.8400	**0.9666**
dermatology-6	0.8800	**1.0000**	**1.0000**	0.9666	0.9666	0.9333	**1.0000**
wilt	0.9828	0.9657	0.9894	0.9917	**0.9921**	0.9850	0.9825

Table 6. Performance comparison of various algorithms in terms of the G-mean metric.

Dataset	BalancedBagging	RUSBoost	Stacking	SIRUS	SADA	NUS-SE	MSFSS
EEG_Eye_State	0.8946	0.7419	0.9677	0.9653	0.9683	0.9065	**0.9727**
banana	0.8837	0.7200	0.9029	0.9025	0.8917	**0.9032**	0.8899
Australian	**0.8656**	0.8238	0.8588	0.8556	0.8539	0.8547	0.8467
brainweb_20000	0.9487	0.9537	0.9529	**0.9550**	0.9461	**0.9550**	0.9518
MovementAAL_RSS	0.7256	0.6571	0.7322	0.7221	0.7244	**0.7327**	0.7083
liver	0.6812	0.6667	0.7038	0.6835	0.6888	0.7066	**0.7080**

Continued on next page

Dataset	BalancedBagging	RUSBoost	Stacking	SIRUS	SADA	NUS-SE	MSFSS
Elec_Partial	0.7334	0.7052	0.7563	0.7500	0.7484	0.7511	**0.7679**
Ionosphere	0.9363	0.7985	**0.9789**	**0.9789**	**0.9789**	0.9363	**0.9789**
glass1	0.7587	0.7126	0.7918	0.7845	0.7932	0.7698	**0.8175**
ecoli-0_vs_1	0.9864	**0.9932**	0.9864	0.9801	0.9864	0.9864	0.9864
wisconsin	0.9606	0.9350	0.9670	0.9645	0.9651	0.9649	**0.9774**
diabetes	0.7508	0.5416	0.7276	0.7442	0.7268	0.7276	**0.7587**
iris0	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
bupa	0.6777	0.6996	**0.7972**	0.7616	0.7554	0.5923	0.7557
wine	0.9744	0.9740	**0.9912**	0.9799	0.9873	0.9873	**0.9912**
titanic	0.6830	0.6997	0.8279	0.7684	0.7684	0.7640	**0.8289**
German	0.7483	0.7521	0.7181	0.7319	0.6792	0.7301	**0.7868**
phoneme	0.8733	0.8064	0.8794	0.8440	0.8635	0.8420	**0.8797**
sober	0.8053	0.7952	0.8175	0.8235	0.8387	0.7721	**0.8806**
ILPD	**0.6487**	0.6463	0.3460	0.1078	0.3549	0.1114	0.6104
glass-0-1-2-3_vs_4-5-6	0.9439	0.9343	0.9043	0.8905	0.9118	0.8766	**0.9473**
blood	0.6408	0.6124	0.5147	0.2377	0.5795	0.5321	**0.6511**
ecoli1	0.8782	0.8161	0.8601	0.8525	0.8205	0.7953	**0.8856**
skin	0.9993	0.9683	0.9990	0.9982	0.9989	0.9990	**0.9997**
usps	0.8547	0.7854	0.9632	0.9531	**0.9689**	**0.9689**	**0.9689**
appendicitis	0.9878	0.9878	**1.0000**	0.9581	0.9816	0.8483	**1.0000**
new-thyroid1	0.9701	0.9758	0.9855	**0.9885**	0.9855	**0.9885**	0.9770
ecoli2	0.8924	0.8247	**0.9429**	**0.9429**	**0.9429**	**0.9429**	**0.9429**
musk	0.9473	0.9240	0.9839	0.9114	0.9460	0.9830	**0.9877**
segment0	0.9893	0.9921	0.9952	0.9972	0.9946	0.9972	**0.9983**
glass-3-5_vs_1-2-6-7	0.9264	0.9159	0.9504	0.9258	0.9310	0.0000	**0.9862**
glass	**0.9458**	0.9400	0.9277	0.9103	0.8984	0.9019	0.9251
page-blocks0	**0.9512**	0.8246	0.9254	0.9043	0.9203	0.8459	0.8923
ecoli-0-4-6_vs_5	0.9944	0.9972	0.9972	0.9421	**1.0000**	0.9972	0.9972
HTRU2	0.9416	0.8580	**0.9601**	0.9311	0.9338	0.9274	0.9444
vowel0	0.9620	0.9609	0.9924	0.9364	0.9946	0.5700	**0.9994**
ecoli-0-1-4-7_vs_2-3-5-6	0.9934	0.9967	0.9984	0.9565	**1.0000**	**1.0000**	0.9983
ecoli4	0.8315	0.9458	**0.9984**	0.9433	**0.9984**	0.8847	**0.9984**
dermatology-6	0.8900	**1.0000**	**1.0000**	0.9985	0.9985	0.9970	**1.0000**
wilt	0.9516	0.9150	**0.9548**	0.9324	0.9170	0.9006	0.9478

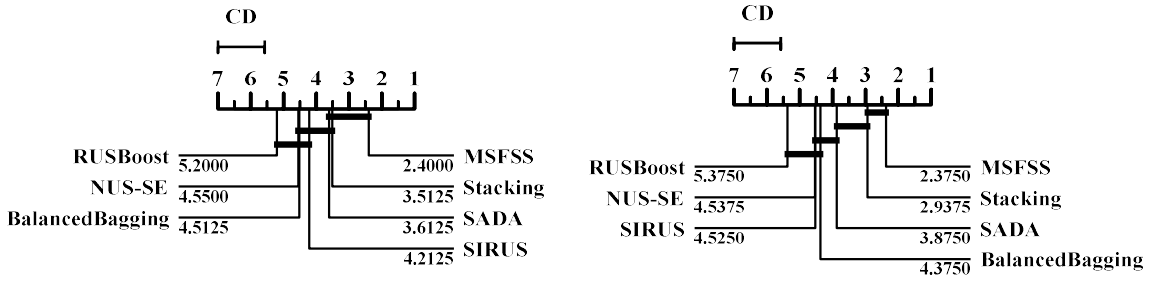
Additionally, we observed an interesting phenomenon, that is, stacking seemed to perform better than its three popular variants which rely on sampling techniques to address class imbalance problems. We analyzed the reasons for this from the following two aspects: (1) most datasets used in our experiments own relatively low class imbalance ratios, which are not enough to destroy the performance of stacking, while in such a scenario, three undersampling-based variants tend to abandon some important classification information, thereby decreasing the classification performance, and (2)

as indicated in Section 2, all three stacking variants adopt a single sampling technique to generate similar training subsets, which is apt to destroy the diversity of the ensemble learner.

Furthermore, it is observed that the proposed MSFSS algorithm performs significantly better than all other compared algorithms. Specifically, the MSFSS has produced the best F-measure results on 27 datasets, and the best G-mean results on 26 datasets, respectively. In contrast to other ensemble learning algorithms, it profits from the following two aspects: enhancing ensemble diversity by introducing the multiple sampling strategies, and improving the quality of the meta feature space by introducing WOA-based feature selection procedure. The results meet our expectations, and meanwhile, indicate the effectiveness and superiority of the proposed MSFSS algorithm.

4.4. Significance analysis

Next, we also analyzed the experimental results with statistics to further observe whether there exist significant differences among these compared algorithms. In particular, we conducted the Freidman test and Nemenyi post-hoc test [53,54], and presented the statistical results in the form of a critical difference (CD) graph (see Figure 4).



(a) (b)

Figure 4. CD statistics graphs of seven compared algorithms on 40 datasets in terms of the (a) F-measure and (b) G-mean.

In Figure 4, we observed that on both metrics, the MSFSS had acquired the lowest average rankings, 2.4 in terms of the F-measure metric, and 2.375 in terms of the G-mean metric, which indicated that it was the best algorithm among all of the compared algorithms on these two metrics. We also noted that in the context of the F-measure, the MSFSS significantly outperformed all of the other algorithms except stacking and SADA, while on the G-mean metric, the MSFSS performed significantly better than all of the other algorithms except the stacking.

4.5. Parameter discussions

In addition to sampling and supervised learning algorithms, there also exist two key parameters which may influence the performance of the proposed MSFSS algorithm. They are, the population size M and the iteration times $MaxI$ of the WOA. Specifically, the $MaxI$ can be seen as a tradeoff between the performance and the efficiency. Figures 5 and 6 present the variances of F-measure and G-mean

performances with increased iteration times on the first ten datasets emerging in Table 2, respectively.

It is not difficult to observe that in the initial phase of the WOA, the performance of the MSFSS improves rapidly, then after 100 iterations, the performance tends to promote slowly until 200 iterations, then the performance will become stable. That means that if a too-small *MaxI* is designated, the optimization procedure of the WOA might not be mature, and may produce a sub-optimized result for MSFSS, while if we give *MaxI* an oversized value, some unnecessary time would be consumed, further decreasing the efficiency of the MSFSS algorithm. It is a safe setting to make the *MaxI* be 200.

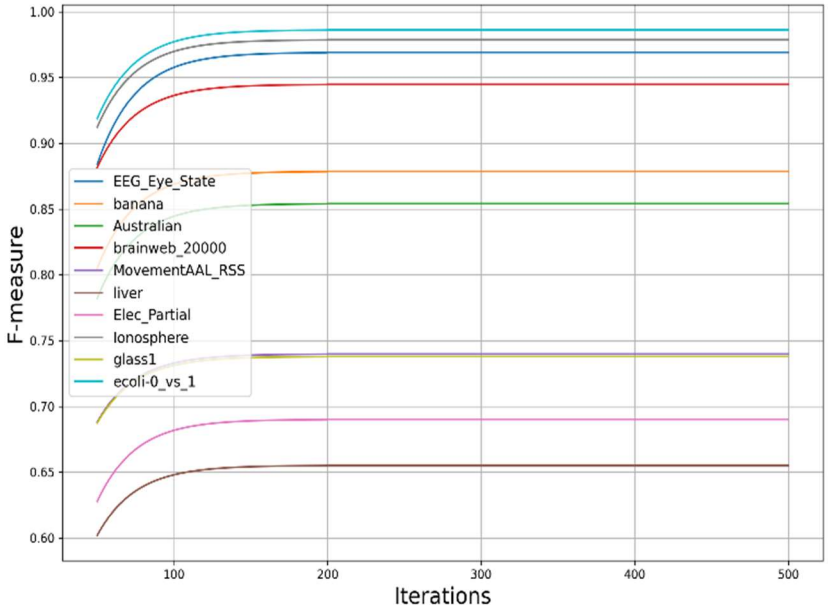


Figure 5. The variance of F-measure performance with increased iterations.

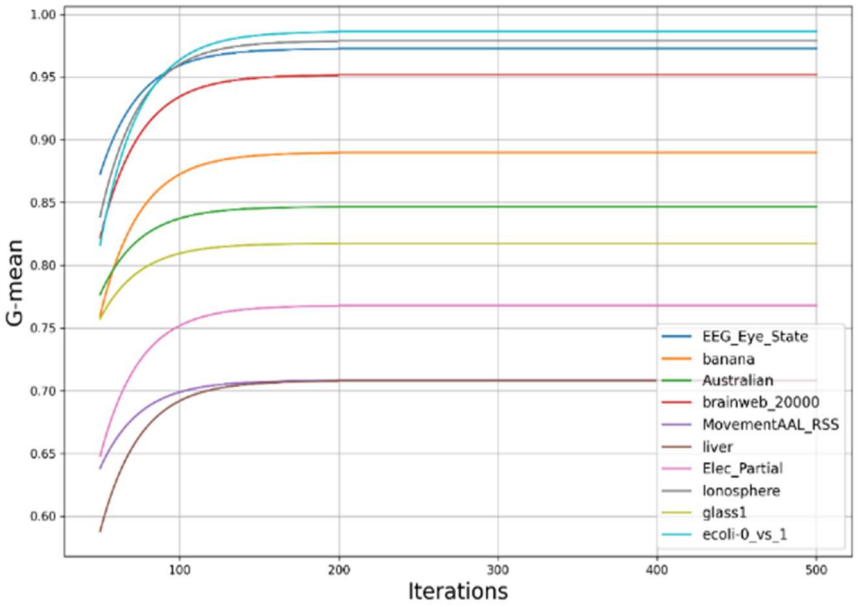


Figure 6. The variance of G-mean performance with increased iterations.

4.6. Ablation experiments

To make it clear whether the two designs both play significant roles in improving the quality of stacking, two groups of ablation experiments were also designed. First, we investigated the role of adopting multiple sampling techniques to promote ensemble diversity. Specifically, we compared the performance of adopting multiple sampling methods in stacking to that of only using a single one. The experimental results are illustrated in Table 7 in which a pairwise *t*-test at 5% significance level was used to record the number of wins/ties/losses throughout 40 datasets.

Table 7. Statistical comparison between adopting multiple sampling techniques and only a single sampling approach in MSFSS based on a pairwise *t*-test at 5% significance level.

Comparison	F-measure			G-mean		
	win	tie	lose	win	tie	lose
ALL vs. SMOTE	28	5	7	30	5	5
ALL vs. ADASYN	30	4	6	29	4	7
ALL vs. SMOTE-ENN	26	8	6	27	8	5
ALL vs. SMOTE-Tomek	25	9	6	26	10	4

The results in Table 7 show that in each group of experiment comparisons, the strategy of adopting multiple sampling techniques wins on significantly more datasets than that of using only one sampling approach. These results verify our assumption that disturbing the data-level method with multiple different kinds of sampling algorithms would be helpful for improving the diversity of base learners in stacking, further helping to improve the quality of stacking ensemble.

Next, we investigated the role of feature selection conducting on the meta feature space of stacking. Specifically, we specified using all original meta features to train the meta learner as the baseline, and investigated how much performance promotion happened on each dataset by adopting the WOA-based feature selection procedure. The performance improvement percentages of the F-measure and G-mean throughout all 40 datasets are presented in Figures 7 and 8, respectively.

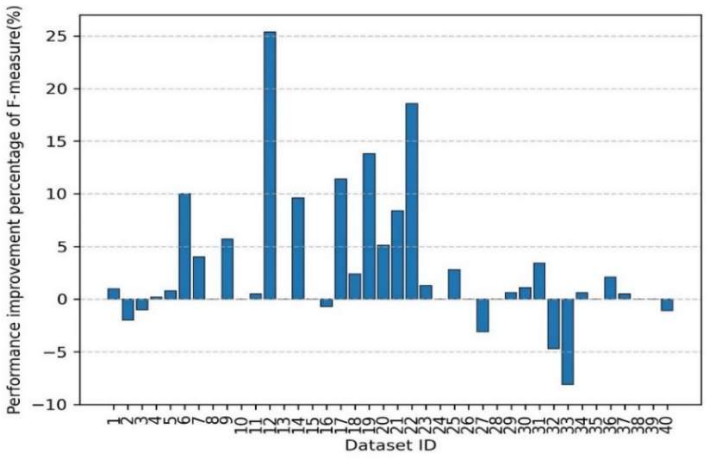


Figure 7. Performance improvement percentages of conducting the WOA-based feature selection procedure compared to directly training the meta learner on the original meta feature space in terms of the F-measure metric.

The results in these two figures show that on most datasets, selecting a sub-group of meta features by the WOA-based feature selection approach helps improving the classification performance in comparison to directly training the meta learner on the original meta feature space, further verifying the rationality of our design at the second layer of stacking. Specifically, feature selection improves F-measure performance on 23 datasets, and only declines on 7 datasets, while in terms of the G-mean metric, the feature selection produces better results on 25 datasets, and worse results on 8 datasets in comparison to training the meta learner on all meta features.

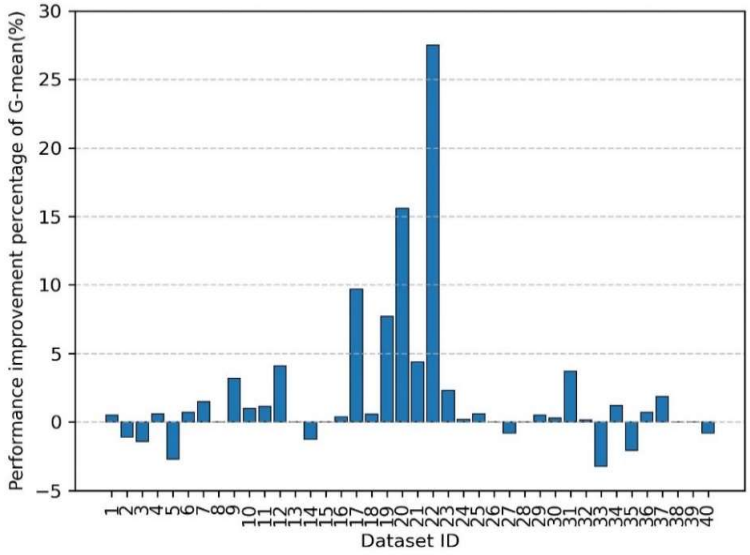


Figure 8. Performance improvement percentages of conducting the WOA-based feature selection procedure compared to directly training the meta learner on the original meta feature space in terms of the G-mean metric.

Obviously, the results of two groups of ablation experiments reflect that our proposed two-structure design for stacking is both helpful for improving the classification performance of stacking ensemble learning model on imbalanced data.

4.7. Comparison of running time

Finally, we compared the running time of the compared algorithms, and presented them in Table 8. It is not difficult to observe that in contrast to several other ensemble learning algorithms, the proposed MSFSS algorithm is generally more time-consuming. Actually, the increased time consumption mainly lies in the introduction of the WOA-based feature selection procedure. However, the size of the meta feature space only associates with two parameters, H and K, but is irrelevant to the original data. Therefore, the increased time consumption is acceptable.

Table 8. Running time comparison of various algorithms (seconds).

Dataset	BalancedBagging	RUSBoost	Stacking	SIRUS	SADA	NUS-SE	MSFSS
EEG_Eye_State	0.7891	0.6383	243.6555	230.2293	370.1172	233.6021	356.2872
banana	0.0793	0.2058	8.0224	7.3567	10.3903	7.3250	11.1298
Australian	0.0481	0.1408	0.9951	0.9127	1.2501	0.8552	3.7041
brainweb_20000	0.3011	0.8021	40.4165	36.0129	81.4095	33.9393	55.4272
MovementAAL_RSS	0.2346	0.3938	99.4214	80.6455	131.9176	101.9191	103.2567
liver	0.0381	0.1324	0.9226	1.0635	0.8770	0.5912	4.4089
Elec_Partial	0.3249	0.4588	60.5196	48.8534	83.7458	52.1630	68.8820
Ionosphere	0.0461	0.1552	15.6837	14.6712	26.4039	20.0584	32.6041
glass1	0.0302	0.1145	0.9230	0.5733	0.5674	0.3908	2.9899
ecoli-0_vs_1	0.0287	0.1076	0.4007	0.4741	0.5277	0.3551	2.8188
wisconsin	0.0332	0.1249	0.6046	0.7440	0.9424	0.6745	3.7607
diabetes	0.0530	0.1269	1.1904	0.8789	1.2320	0.8630	3.6307
iris0	0.0282	0.0059	0.5952	0.4395	0.6160	0.4315	1.8153
bupa	0.0307	0.1190	1.4230	0.5237	0.6051	0.4285	3.1481
wine	0.0292	0.1135	0.4607	0.5257	0.5833	0.3888	2.9973
titanic	0.0362	0.1453	2.5668	2.1130	5.1227	2.0455	6.6073
German	0.0396	0.1299	1.6343	1.1685	2.2895	1.2955	4.5573
phoneme	0.1046	0.2395	11.3943	6.7378	21.6974	7.1643	19.9138
sober	0.0292	0.1120	0.5004	0.4723	0.4999	0.3174	3.1059
ILPD	0.0367	0.1284	0.7489	0.6924	1.0654	0.6249	3.8425
glass-0-1-2-3_vs_4-5-6	0.0282	0.1180	0.4270	0.4841	0.6487	0.4741	3.3287
blood	0.0317	0.1120	0.7826	0.7321	1.2558	0.7281	3.7056
ecoli1	0.0287	0.1116	0.4647	0.5872	0.6586	0.4106	2.9710
skin	1.3888	4.1476	936.7546	251.7978	4097.1584	1281.0152	3206.3131
usps	0.4761	0.9354	11.1066	8.0909	21.5724	9.5094	22.5322
appendicitis	0.0282	0.0124	1.1195	0.4265	3.0255	2.1963	5.2636
new-thyroid1	0.0287	0.1135	0.5356	0.5059	0.6051	0.4027	3.0385
ecoli2	0.0292	0.1125	0.4696	0.5019	0.6309	0.4146	3.1154
musk	1.3928	1.5580	46.5965	15.2224	101.1228	31.9450	73.3401
segment0	0.0481	0.1825	7.3661	5.9441	13.9834	8.3408	16.7243
glass-3-5_vs_1-2-6-7	0.0287	0.1145	0.5917	0.4761	0.5833	0.3769	3.2647
glass	0.0367	0.1319	0.5594	0.7876	0.8094	0.5039	3.8430
page-blocks0	0.0639	0.2123	7.4296	4.7497	44.3867	4.6446	49.8388
ecoli-0-4-6_vs_5	0.0312	0.0114	0.4836	0.4662	3.0772	0.3591	5.2641
HTRU2	0.1701	0.5039	26.3178	11.2594	271.1350	11.8090	148.6880
vowel0	0.0332	0.1314	0.8670	0.7559	1.3868	0.6487	3.8877
ecoli-0-1-4-7_vs_2-3-5-6	0.0287	0.0198	0.5991	0.4880	0.5713	0.3948	3.0519
ecoli4	0.0277	0.1145	0.9161	0.5436	3.2776	2.1586	5.3697
dermatology-6	0.0317	0.1155	0.5748	0.6031	0.7638	0.4146	3.1997
wilt	0.0897	0.2877	6.7493	5.0379	28.6223	2.7104	31.0440

5. Conclusions

In this study, an improved stacking ensemble learning algorithm called MSFSS was proposed to address an imbalanced data classification problem. The algorithm first considered to enhance ensemble diversity by integrating multiple sampling strategies with multiple supervised learning algorithms. Then, it introduced a WOA-based feature selection procedure to improve the quality of the meta feature space, further producing a better meta classifier. In experiments, we first compared the proposed MSFSS algorithm with several baselines and state-of-the-art stacking variants which are specifically designed for solving class imbalance problems. Specifically, on 40 benchmarked datasets, the MSFSS acquired best results in terms of the F-measure metric on 27 datasets, and best results in terms of the G-mean metric on 26 datasets, indicating its superiority. By statistical analysis, we observed that the proposed MSFSS significantly outperformed its several competitors, except stacking and SADA, in the context of the F-measure, and the stacking in terms of the G-mean metric. Also, the results of ablation experiments illustrated the effectiveness and necessity of a two-module design in MSFSS. Additionally, we compared the running time of various algorithms, and observed that the proposed MSFSS algorithm seemed to be more time-consuming than several others. However, profiting from the faster convergence speed of the WOA, the increment of running time of the MSFSS was totally acceptable. Therefore, we do not think that this factor would limit the use of the MSFSS algorithm in various real-world applications.

In future work, we plan to integrate more novel and high-quality sampling strategies and classification algorithms into the MSFSS algorithm framework to investigate whether its performance could be further improved. Also, we wish to explore whether there exist some more efficient optimization algorithms that could further enhance the quality and efficiency of meta feature selection.

Author contributions

Shuxiang Wang: Conceptualization, Data curation, Investigation, Methodology, Writing-original draft; Changbin Shao: Formal analysis, Methodology, Writing-original draft; Sen Xu: Validation, Visualization, Funding acquisition; Xibei Yang: Funding acquisition, Supervision, Writing-review & editing; Hualong Yu: Conceptualization, Methodology, Supervision, Funding acquisition, Writing-review & editing. All authors have read and approved the final version of the manuscript for publication.

Use of AI tools declaration

The authors declare that they have not used artificial intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant Nos. 62176107, 62076111, and 62076215, the Open Project of Jiangsu Key Laboratory of Media Design and Software Technology (Jiangnan University), and the Qinglan Project of Jiangsu Province.

Conflict of interests

The authors declare that there are no conflicts of interest.

References

1. P. Branco, L. Torgo, R. P. Ribeiro, A survey of predictive modeling on imbalanced domains, *ACM Comput. Surv. (CSUR)*, **49** (2016), 1–50. https://doi.org/10.1145/2907070
2. K. Oksuz, B. C. Cam, S. Kalkan, E. Akbas, Imbalance problems in object detection: A review, *IEEE T. Pattern Anal.*, **43** (2021), 3388–3415. https://doi.org/10.1109/TPAMI.2020.2981890
3. M. Ghorbani, A. Kazi, M. S. Baghshah, H. R. Rabiee, N. Navab, RA-GCN: Graph convolutional network for disease prediction problems with imbalanced data, *Med. Image Anal.*, **75** (2022), 102272. https://doi.org/10.1016/j.media.2021.102272
4. Y. C. Wang, C. H Cheng, A multiple combined method for rebalancing medical data with class imbalances, *Comput. Biol. Med.*, **134** (2021), 104527. https://doi.org/10.1016/j.compbiomed.2021.104527
5. A. Abdelkhalek, M. Mashaly, Addressing the class imbalance problem in network intrusion detection systems using data resampling and deep learning, *J. Supercomput.*, **79** (2023), 10611–10644. https://doi.org/10.1007/s11227-023-05073-x
6. Z. Li, K. Kamnitsas, B. Glocker, Analyzing overfitting under class imbalance in neural networks for image segmentation, *IEEE T. Med. Imaging*, **40** (2021), 1065–1077. https://doi.org/10.1109/TMI.2020.3046692
7. V. Rupapara, F. Rustam, H. F. Shahzad, A. Mehmood, I. Ashraf, G. S. Choi, Impact of SMOTE on imbalanced text features for toxic comments classification using RVVC model, *IEEE Access*, **9** (2021), 78621–78634. https://doi.org/10.1109/ACCESS.2021.3083638
8. W. Zheng, Y. Xun, X. Wu, Z. Deng, X. Chen, Y. Sui, A comparative study of class rebalancing methods for security bug report classification, *IEEE T. Reliab.*, **70** (2021), 1658–1670. https://doi.org/10.1109/TR.2021.3118026
9. J. Kuang, G. Xu, T. Tao, Q. Wu, Class-imbalance adversarial transfer learning network for cross-domain fault diagnosis with imbalanced data, *IEEE T. Instrum. Meas.*, **71** (2021), 1–11. https://doi.org/10.1109/TIM.2021.3136175
10. M. Qian, Y. F. Li, A weakly supervised learning-based oversampling framework for class-imbalanced fault diagnosis, *IEEE T. Reliab.*, **71** (2022), 429–442. https://doi.org/10.1109/TR.2021.3138448
11. Y. Aydın, Ü. Işıkdağ, G. Bekdaş, S. M. Nigdeli, Z. W. Geem, Use of machine learning techniques in soil classification, *Sustainability*, **15** (2023), 2374. https://doi.org/10.3390/su15032374
12. M. Asgari, W. Yang, M. Farnaghi, Spatiotemporal data partitioning for distributed random forest algorithm: Air quality prediction using imbalanced big spatiotemporal data on spark distributed framework, *Environ. Technol. Inno.,* **27** (2022), 102776. https://doi.org/10.1016/j.eti.2022.102776
13. L. Dou, F. Yang, L. Xu, Q. Zou, A comprehensive review of the imbalance classification of protein post-translational modifications, *Brief. Bioinform.*, **22** (2021), bbab089. https://doi.org/10.1093/bib/bbab089
14. S. Y. Bae, J. Lee, J. Jeong, C. Lim, J. Choi, Effective data-balancing methods for class-imbalanced genotoxicity datasets using machine learning algorithms and molecular fingerprints, *Comput. Toxicol.*, **20** (2021), 100178. https://doi.org/10.1016/j.comtox.2021.100178

15. G. H. Fu, Y. J. Wu, M. J. Zong, J. Pan, Hellinger distance-based stable sparse feature selection for high-dimensional class-imbalanced data, *BMC Bioinformatics*, **21** (2020), 121. https://doi.org/10.1186/s12859-020-3411-3
16. N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, SMOTE: Synthetic minority over-sampling technique, *J. Artif. Intell. Res.*, **16** (2002), 321–357. https://doi.org/10.1613/jair.953
17. G. E. A. P. A. Batista, R. C. Prati, M. C. Monard, A study of the behavior of several methods for balancing machine learning training data, *ACM SIGKDD Explor. Newslett.*, **6** (2004), 20–29. https://doi.org/10.1145/1007730.1007735
18. H. He, Y. Bai, E. A. Garcia, S. Li, *ADASYN: Adaptive synthetic sampling approach for imbalanced learning*, In: 2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence), IEEE Press, 2008. https://doi.org/10.1109/IJCNN.2008.4633969
19. M. Kubat, S. Matwin, *Addressing the curse of imbalanced training sets: one-sided selection*, In: International Conference of Machine Learning, Morgan Kaufmann, 1997.
20. M. A. Tahir, J. Kittler, F. Yan, Inverse random under sampling for class imbalance problem and its application to multi-label classification, *Pattern Recogn.*, **45** (2012), 3738–3750. https://doi.org/10.1016/j.patcog.2012.03.014
21. A. Zhang, H. Yu, Z. Huan, X. Yang, S. Zheng, S. Gao, SMOTE-R*k*NN: A hybrid re-sampling method based on SMOTE and reverse *k*-nearest neighbors, *Inform. Sci.*, **595** (2022), 70–88. https://doi.org/10.1016/j.ins.2022.02.038
22. R. Batuwita, V. Palade, FSVM-CIL: Fuzzy support vector machines for class imbalance learning, *IEEE T. Fuzzy Syst.*, **18** (2010), 558–571. https://doi.org/10.1109/TFUZZ.2010.2042721
23. H. Yu, C. Sun, X. Yang, S. Zheng, H Zou, Fuzzy support vector machine with relative density information for classifying imbalanced data, *IEEE T. Fuzzy Syst.*, **27** (2019), 2353–2367. https://doi.org/10.1109/TFUZZ.2019.2898371
24. H. Yu, C. Mu, C. Sun, W. Yang, X. Yang, X. Zuo, Support vector machine-based optimized decision threshold adjustment strategy for classifying imbalanced data, *Knowl.-Based Syst.,* **76** (2015), 67–78. https://doi.org/10.1016/j.knosys.2014.12.007
25. H. Yu, C. Sun, X. Yang, W. Yang, J. Shen, Y. Qi, ODOC-ELM: Optimal decision outputs compensation-based extreme learning machine for classifying imbalanced data, *Knowl.-Based Syst.,* **92** (2016), 55–70. https://doi.org/10.1016/j.knosys.2015.10.012
26. J. Laurikkala, *Improving identification of difficult small classes by balancing class distribution*, In: Artificial Intelligence in Medicine: 8th Conference on Artificial Intelligence in Medicine in Europe, AIME 2001 Cascais, Portugal, Springer Berlin Heidelberg, 2001. https://doi.org/10.1007/3-540-48229-6_9
27. F. S. Hanifah, H. Wijayanto, A. Kurnia, Smotebagging algorithm for imbalanced dataset in logistic regression analysis (case: Credit of bank X), *Appl. Math. Sci.*, **9** (2015), 6857–6865. http://dx.doi.org/10.12988/ams.2015.58562
28. C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, A. Napolitano, *RUSBoost: Improving classification performance when training data is skewed*, In: 19th international conference on pattern recognition, IEEE, 2008. https://doi.org/10.1002/abio.370040210
29. Y. Zhang, G. Liu, W. Luan, C. Yan, C. Jiang, *An approach to class imbalance problem based on Stacking and inverse random under sampling methods*, In: 2018 IEEE 15th international conference on networking, sensing and control (ICNSC), IEEE, 2018. https://doi.org/10.1002/abio.370040210

30. Y. Pristyanto, A. F. Nugraha, I. Pratama, A. Dahlan, L. A. Wirasakti, *Dual approach to handling imbalanced class in datasets using oversampling and ensemble learning techniques*, In: 2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM), IEEE, 2021. https://doi.org/10.1002/abio.370040210
31. Z. Seng, S. A. Kareem, K. D. Varathan, A neighborhood undersampling stacked ensemble (NUS-SE) in imbalanced classification, *Exp. Syst. Appl.*, **168** (2021), 114246. https://doi.org/10.1016/j.eswa.2020.114246
32. D. H. Wolpert, Stacked generalization, *Neural Networks,* **5** (1992), 241–259. https://doi.org/10.1016/S0893-6080(05)80023-1
33. Y. Shi, R. Eberhart, *A modified particle swarm optimizer*, In: Proceedings of 1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360), IEEE, 1998, 69–73. https://doi.org/10.1109/icec.1998.699146
34. K. V. Price, *Differential evolution: A fast and simple numerical optimizer*, In: Proceedings of North American fuzzy information processing, IEEE, 1996, 524–527. https://doi.org/10.1109/nafips.1996.534790
35. E. Cuevas, M. Cienfuegos, D. Zaldívar, M. Pérez-Cisneros, A swarm optimization algorithm inspired in the behavior of the social-spider, *Exp. Syst. Appl.*, **40** (2013), 6374–6384. https://doi.org/10.1016/j.eswa.2013.05.041
36. S. Mirjalili, A. Lewis, The whale optimization algorithm, *Adv. Eng. Soft.*, **95** (2016), 51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008
37. E. Cuevas, A. Rodríguez, M. Perez, J. Murillo-Olmos, B. Morales-Castañeda, A. Alejo-Reyes, et al., Optimal evaluation of re-opening policies for COVID-19 through the use of metaheuristic schemes, *Appl. Math. Model.*, **121** (2023), 506–523. https://doi.org/10.1016/j.apm.2023.05.012
38. M. H. Nadimi-Shahraki, S. Taghian, S. Mirjalili, L. Abualigah, M. Abd Elaziz, D. Oliva, EWOA-OPF: Effective whale optimization algorithm to solve optimal power flow problem, *Electronics*, **10** (2021), 2975. https://doi.org/10.1007/978-981-16-9447-9_20
39. R. Kundu, S. Chattopadhyay, E. Cuevas, R. Sarkar, AltWOA: Altruistic Whale Optimization Algorithm for feature selection on microarray datasets, *Comput. Biol. Med.*, **144** (2022), 105349. https://doi.org/10.1016/j.compbiomed.2022.105349
40. M. S. Santos, P. H. Abreu, N. Japkowicz, A. Fernández, C. Soares, S. Wilk, et al., On the joint-effect of class imbalance and overlap: a critical review, *Artif. Intell. Rev.*, **55** (2022), 6207–6275. https://doi.org/10.1007/s10462-022-10150-3
41. S. K. Pandey, A. K. Tripathi, An empirical study toward dealing with noise and class imbalance issues in software defect prediction, *Soft Comput.*, **25** (2021), 13465–13492. https://doi.org/10.1007/s00500-021-06096-3
42. L. Breiman, Bagging predictors, *Mach. Learn.*, **24** (1996), 123–140. https://doi.org/10.1007/BF00058655
43. R E. Schapire, The strength of weak learnability, *Mach. Learn.*, **5** (1990), 197–227. https://doi.org/10.1007/BF00116037
44. A. Krogh, J. Vedelsby, Neural network ensembles, cross validation, and active learning, *Adv. Neural Inform. Proces. Syst.*, **7** (1995), 231–238. Available from: http://papers.nips.cc/paper/1001-neural-network-ensembles-cross-validation-and-active-learning.

45. S. Zhang, X. Li, M. Zong, X. Zhu, R. Wang, Efficient kNN classification with different numbers of nearest neighbors, *IEEE T. Neur. Net. Learn.*, **29** (2018), 1774–1785. https://doi.org/10.1109/TNNLS.2017.2673241
46. J R. Quinlan, Induction of decision trees, *Mach. Learn.*, **1** (1986), 81–106. https://doi.org/10.1023/A:1022643204877
47. C. Cortes, V. Vapnik, Support-vector networks, *Mach. Learn.*, **20** (1995), 273–297. https://doi.org/10.1007/BF00994018
48. T. Bayes, An essay towards solving a problem in the doctrine of chances, *MD Comput. Comput. Med. Pract.*, **8** (1991), 376–418. https://doi.org/10.1002/abio.370040210
49. A. Tharwat, T. Gaber, A. Ibrahim, A. E. Hassanien, Linear discriminant analysis: A detailed tutorial, *AI Commun.*, **30** (2017), 169–190. https://doi.org/10.3233/AIC-170729
50. X. Su, X. Yan, C. L. Tsai, Linear regression, *WIRES Comput. Stat.*, **4** (2012), 275–294. https://doi.org/10.1002/wics.1198
51. C. Blake, E. Keogh, C. J. Merz, *UCI repository of machine learning databases*, Department of Information and Computer Science, University of California, Irvine, CA, USA, 1998. Available from: http://www.ics.uci.edu/mlearn /MLRepository.html.
52. I. Triguero, S. González, J. M. Moyano, S. García, J. Alcalá-Fdez, J. Luengo, et al., KEEL 3.0: An open source software for multi-stage analysis in data mining international, *J. Comput. Intell. Syst.*, **10** (2017), 1238–1249. https://doi.org/10.2991/ijcis.10.1.82
53. J. Demsar, Statistical comparisons of classifiers over multiple data sets, *J. Mach. Learn. Res.*, **7** (2006), 1–30. Available from: http://jmlr.org/papers/v7/demsar06a.html.
54. S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power, *Inform. Sci.*, **180** (2010), 2044–2064. https://doi.org/10.1016/j.ins.2009.12.010

AIMS Press