applied sciences

MDPI

Article

Strip Steel Defect Prediction Based on Improved Immune Particle Swarm Optimisation–Improved Synthetic Minority Oversampling Technique–Stacking

Zhi Fang [1], Fan Zhang [2], Su Yu [3] and Bintao Wang [2,*]

[1] School of Mechanical and Automotive Engineering, Shanghai University of Engineering Science, Shanghai 201620, China; 19117166659@163.com
[2] Engineering Training Center, Shanghai University of Engineering Science, Shanghai 201620, China; pdszhangfan@sues.edu.cn
[3] Information Office, Shanghai University of Engineering Science, Shanghai 201620, China; yusu@sues.edu.cn
* Correspondence: bitomw@sues.edu.cn

Abstract: A model framework for the prediction of defects in strip steel is proposed with the objective of enhancing the accuracy of defect detection. Initially, the data are balanced through the utilisation of the Improved Synthetic Minority Oversampling Technique (ISmote), which is based on clustering techniques. Subsequently, further enhancements are made to the inertia weights and learning factors of the immune particle swarm optimisation (IPSO), with additional optimisations in speed updates and population diversity. These enhancements are designed to address the issue of premature convergence at the early stages of the process and local optima at the later stages. Finally, a prediction model is then constructed based on stacking, with its hyperparameters optimised through the improved immune particle swarm optimisation (IIPSO). The results of the experimental trials demonstrate that the IIPSO-ISmote-Stacking model framework exhibits superior prediction performance when compared to other models. The Macro_Precision, Macro_Recall, and Macro_F1 values for this framework are 93.3%, 93.6%, and 92.2%, respectively.

Keywords: strip defect; stacking; immune particle swarm optimization; smote; multi-class prediction

check for updates

Citation: Fang, Z.; Zhang, F.; Yu, S.; Wang, B. Strip Steel Defect Prediction Based on Improved Immune Particle Swarm Optimisation–Improved Synthetic Minority Oversampling Technique–Stacking. *Appl. Sci.* **2024**, *14*, 5849. https://doi.org/10.3390/app14135849

Academic Editor: Andrea Carpinteri

Received: 9 May 2024
Revised: 23 June 2024
Accepted: 30 June 2024
Published: 4 July 2024

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

1. Introduction

Strip steel [1], a critical material in the industry, is employed extensively across diverse fields due to its notable mechanical properties and high surface quality. Nevertheless, during the manufacturing process, various defects may appear on strip steel, including scratches, pits, and inclusions. Such defects not only affect the aesthetic quality of the product but may also compromise its mechanical performance. Consequently, the prediction and identification of strip steel defects are of great significance for the improvement of product quality and production efficiency.

The traditional methods for detecting defects in strip steel, which often rely on manual inspection and simple automation techniques, are beset with challenges. These include high labour costs, subjectivity in defect assessment, and limited detection capabilities. In recent years, machine learning techniques have proven to be highly effective in defect detection, offering a solution for automating the process of strip steel defect detection. Among the various machine learning techniques, stacking [2] is an ensemble learning method that combines multiple base models into a meta-model in order to leverage the strengths of each individual model for accurate predictions. This method has been employed in a variety of industrial domains, including the prediction of the shear strength of reinforced concrete beams [3], the lifetime prediction of electronic devices [4], and the identification of inferior compositions in steel [5].

Nevertheless, the issue of data imbalance [6] has a negative impact on the application of machine learning to defect prediction. This imbalance can result in a biased learning

process, leading to models that perform well on the majority class but poorly on the minority class. To address this issue, the ISmote algorithm is employed to achieve data balance. Moreover, it is of paramount importance to optimise the hyperparameters of machine learning models in order to enhance predictive performance [7]. Conventional hyperparameter optimisation techniques, such as grid search and random search, frequently prove inadequate for efficiently navigating the intricate and high-dimensional hyperparameter space associated with stacking models. To address this issue, the IIPSO algorithm is employed to identify optimal hyperparameter settings for the model.

This paper presents a comprehensive model framework integrating the ISmote algorithm, the stacking model, and the IIPSO algorithm. The objective of this framework is to enhance the accuracy of defect prediction, thereby contributing to the advancement of quality control processes in the strip steel industry.

The primary contributions of this paper are as follows:

(1) At the stage of data processing, the K-means++ algorithm is employed to cluster the minority classes. Subsequently, the number of samples generated in each cluster is determined based on the density of each cluster. Based on the determined synthetic samples, the Smote algorithm is used to synthesise samples in the minority cluster. The ISmote algorithm can assist the model in avoiding bias towards the majority class, thereby enhancing its classification performance.

(2) The standard IPSO algorithm is prone to premature convergence. To address this challenge, IPSO is enhanced with a dynamic parameter adjustment mechanism and a mutation strategy. During the model training phase, the IIPSO algorithm is used to optimise the hyperparameters of the stacked model, thereby obtaining more accurate defect prediction results.

(3) The experimental evaluations were conducted on the strip defect dataset. The results demonstrate that the predictive model, enhanced by data balance and hyperparameter optimization, significantly outperforms all comparator models in terms of defect prediction.

The rest of this paper is structured as follows. Section 2 provides a concise overview of related work on the processing of class imbalance and the optimisation of model hyperparameters. Section 3 presents a detailed account of the enhancements to the Smote algorithm and IPSO algorithm, as well as the construction of the strip defect prediction model. Furthermore, Section 4 outlines the experimental design, after which the results are analysed. Finally, Section 5 presents the conclusion and outlines the prospective directions of this paper.

2. Related Work

The processing of imbalanced data is typically conducted during the preprocessing phase of the modelling process. Since its inception, the Smote algorithm [8] has been a common tool for addressing class imbalance problems in classification tasks. Over the years, various enhancements and evolutions of the Smote algorithm have been proposed in order to improve its effectiveness. Chen et al. [9] enhanced the Smote algorithm by optimising the interpolation process. This involved introducing Gaussian deviation to generate synthetic samples that were closer to the class boundaries. This improved the classifier accuracy in these regions. Arafa et al. [10] proposed the Reduced Noise–Smote (RN-Smote) algorithm, which integrates the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) to identify and remove noise within the oversampling process, thus ensuring cleaner training data. Chen et al. [11] proposed a novel parameter-free Smote algorithm that focuses on generating representative synthetic instances to augment the minority class's coverage. This approach expands the decision margin while minimizing noisy insertions. Sáez et al. [12] proposed an extension of the Smote approach, namely, the Iterative-Partitioning Filter (IPF), an ensemble-based noise filter that addresses issues arising from noisy and borderline instances in imbalanced datasets through an iterative process. Chen et al. [13] proposed the adaptive robust Smote (RSmote) algorithm, a novel

approach that categorises non-noisy minority samples dynamically into boundary and safe samples based on their relative density characteristics. RSmote then synthesises new samples specifically within these boundary and safe areas.

The aforementioned work primarily enhances the Smote algorithm's processing capabilities by mitigating the impact of noise. However, the issue of sample distribution is not addressed, resulting in the generation of an excess of samples in areas with a high sample density, which in turn leads to the occurrence of sample overlap. To address these issues, two improvements are made to the Smote algorithm. Firstly, the K-means++ algorithm is employed to cluster minority class samples. Subsequently, the precise number of samples to be generated for each cluster class is determined by calculating the sampling weight of each cluster class. This improves the consideration of the internal structure of the minority class. Secondly, interpolation is performed between the cluster centre and neighbouring points in order to make the generated samples more representative of the overall characteristics.

The field of intelligent optimisation algorithms has witnessed significant advancements through the integration of various theories and mechanisms. One noteworthy advancement in this field is the IPSO algorithm [14], which combines the principles of artificial immune system theory with particle swarm optimisation (PSO) [15]. Researchers have sought to enhance the performance of the algorithm by introducing more adaptable parameter adjustment mechanisms and dynamic strategies. Qiao et al. [16] proposed a methodology based on IIPSO with dynamic disturbance term (IPSO-DDT). This approach employs genetic algorithm matrix coding for particle representation and integrates an immune information processing mechanism. This combination preserves individual diversity, thereby mitigating premature convergence. In a further contribution to this field. Li et al. [17] proposed the IIPSO, which dynamically adjusts the scale of sub-populations and incorporates a concentration regulation mechanism, along with an adaptive search range adjustment for vaccination. These modifications serve to prevent the degradation of the population and to enhance both the convergence performance and the global search efficiency. Sun et al. [18] presented an innovative method combining IPSO with the Simulated Annealing (SA) algorithm to facilitate collaborative search operations. This approach effectively prevents IPSO from becoming trapped at local optima and ensures that the search continues towards the global optimum. Zhao et al. [19] proposed an IIPSO variant that utilises adaptive inertia weight to balance global exploration and local exploitation, significantly improving the algorithm's overall performance and convergence efficiency.

In view of the aforementioned ideas, the immune particle swarm algorithm has undergone a process of enhancement. The algorithm employs dynamically adjusted nonlinear inertial weights and learning factors, which adapt to the search requirements during the iteration process. This enhances the convergence accuracy and speed of the algorithm. Furthermore, the enhanced Bernoulli mapping is incorporated into the speed update, and a mutation mechanism is introduced to maintain population diversity. These strategies effectively suppress the issue of premature convergence and enhance the global optimisation capability.

3. Materials and Methods

3.1. Data Encoding

The dataset utilised in this study, which concerns strip defects, was sourced from the University of California, Irvine (UCI) machine learning repository. It consists of 34 feature variables related to strip steel defects. Of these, 27 are descriptive feature variables that represent various attributes of the strip steel, while the remaining are defect feature variables that indicate the presence of specific types of defects in the strip steel. To facilitate the analysis and interpretation of the dataset, a transformation was applied to the original seven defect feature variables. As illustrated in Figure 1, these defect feature variables were merged into a single, new feature variable, labelled "Defect". This new variable was then utilised as the ultimate target variable for subsequent prediction and classification.

Pastry	Z_Scratch	K_Scratch	Stains	Dirtiness	Bumps	Other_Faults	Type Encoding	Defect
1	0	0	0	0	0	0	1000000	1
0	1	0	0	0	0	0	0100000	2
0	0	1	0	0	0	0	0010000	3
0	0	0	1	0	0	0	0001000	4
0	0	0	0	1	0	0	0000100	5
0	0	0	0	0	1	0	0000010	6
0	0	0	0	0	0	1	0000001	7

Figure 1. Result of data encoding.

3.2. Improved Smote Algorithm

In 2002, Chawla proposed the Smote algorithm, a technological approach designed to address the issue of data imbalance in classification tasks. The algorithm synthesises samples from underrepresented classes, thereby compensating for their numerical disadvantages. Consequently, the risk of model overfitting is reduced, thereby enhancing the overall performance and reliability of the model. While the Smote algorithm enhances the representativeness of the minority class, it may also produce redundant data and noise in the process, further increasing the imbalance of positive and negative samples. In response to the aforementioned issues, we have proposed an improved version of the Smote algorithm (ISmote), comprising three distinct stages: clustering, the determination of the sampling number, and oversampling.

K-means++ [20] is an optimised version of the K-means algorithm. The algorithm employs a probabilistic strategy to select initial clustering centres, thereby ensuring that they can more effectively cover the entire data space, reducing the algorithm's sensitivity to initial conditions, and improving the quality of the resulting clustering. In this study, firstly, the K-means++ algorithm is employed to cluster the original minority class samples. Secondly, the synthesised number of samples for each cluster class is determined by calculating the sparsity of the respective cluster group. In this process, the number of samples to be synthesised for each cluster is typically calculated based on the cardinality of each cluster. However, this approach only considers the number of samples in each cluster, without accounting for the distribution of samples in space. As illustrated in Figure 2, two clusters with the same cardinality exhibit disparate sample distributions. Consequently, the synthesized number of samples for clustering is determined by calculating the sparsity of each cluster group. The higher the sparsity of the cluster group, the greater the sampling weight, and thus the greater the number of samples that must be synthesised. Conversely, clusters with a lower sparsity require a smaller number of synthetic samples. By considering the sparsity of cluster, it is possible to generate more samples in areas with sparse samples. This helps the model to better learn the characteristics of these areas, thereby improving its generalisation ability.

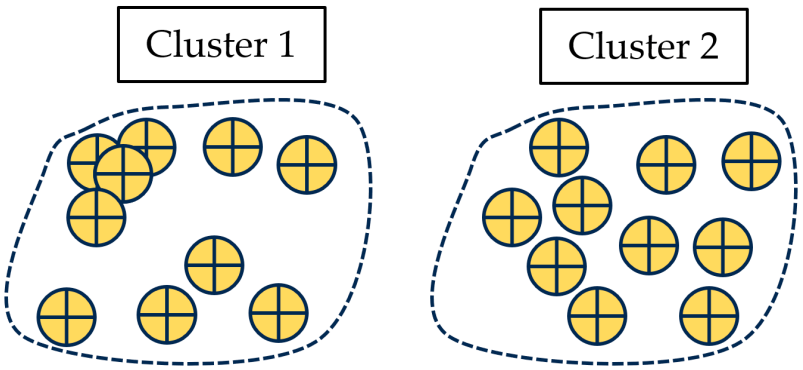


Figure 2. Cluster distribution.

Once the requisite number of samples for the synthesis of each cluster has been determined, the Smote algorithm is employed to oversample in accordance with the determined number of synthesized samples. Concurrently, the interpolation methodology inherent to the original Smote algorithm is improved, and instead, the new sample is synthesized between the centre of the cluster and neighbor points, as shown in Figure 3. The pseudo of the ISmote algorithm is given in Algorithm 1.

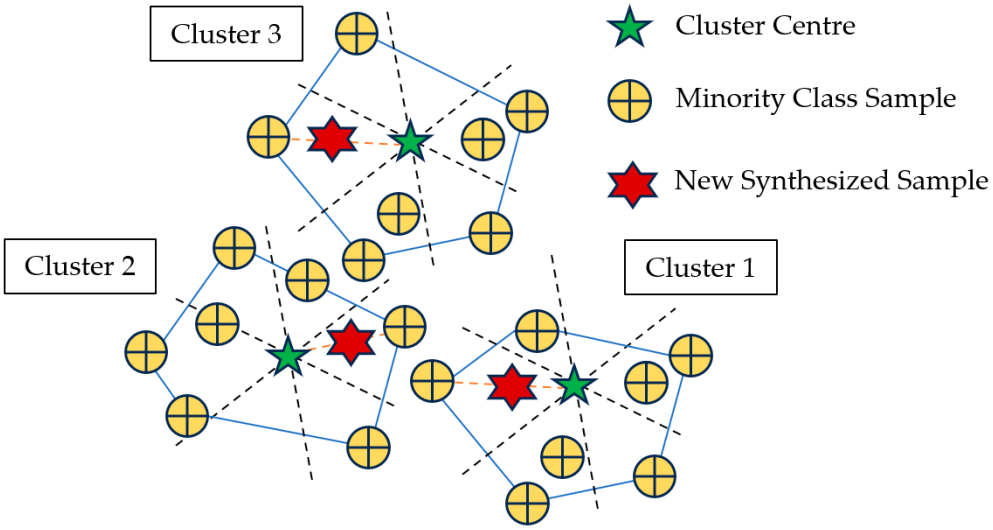


Figure 3. Process of generating the synthetic samples.

Algorithm 1: ISmote Algorithm

Input: original minority class sample N, number of samples to be generated n, nearest neighbor number knn, number of clusters k//k value ranges from 5% to 10% of the number of minority class samples
Output: the date set obtained after oversampling S
//Step 1: cluster the original samples.
1. clusters ← K-means++ (N, k)
2. selected clusters ← $\{C_1, C_2, ..., C_k\}$
//Step 2: determine the number of samples to be synthesized
3. for f ∈ selected clusters do
4. density factor(f) ← $mean(local\ density(f))$//average of the local density
5. sparsity factor(f) ← $1/Density\ factors$//sparsity of each cluster
6. end for
7. sparsity sum ← $\sum_{f \in selected\ clusters} sparsity\ factor(f)$
8. sampling weight ← $sparsity\ factor(f)/sparsity\ sum$
9. generated samples ← $\|n \times sampling\ weight(f)\|$
//Step 3: Smote oversampling
10. S ← $generated\ samples \cup \{Smote(f, numberofsamples, knn)\}$
11. Return S

3.3. Improved Immune Particle Swarm Optimization

The IPSO builds upon the foundation of the PSO, yet it still exhibits certain limitations, such as premature convergence and susceptibility to local optimality. A number of scholars have put forward a range of improvement strategies as a result of their continuous research. Through continued practice and implementation, these strategies enhance the optimisation performance of the IPSO. Nevertheless, they are not without limitations. Consequently, this paper presents an improved IPSO based on multi-strategy fusion. This approach has been designed to address the issue of premature convergence at the early stages and local optimality at the later stages.

3.3.1. Immune Particle Swarm Optimization

In the 1990s, scholars Kennedy and Eberhart introduced the concept of PSO. The fundamental premise of this concept is to simulate a flock of birds by establishing a set of massless particles. Each particle represents a bird within the flock and emulates the process of foraging. The particles independently search the problem space to identify the optimal solution, with the best solution found regarded as the individual extreme value. Subsequently, the individual extreme values are compared to those of other particles, with the optimal individual extreme value obtained serving as the local optimal solution at the current stage. Concurrently, each particle modifies its velocity and position in accordance with its current optimization status and that of other particles. Ultimately, a globally optimal solution is achieved through the continuous updating and iteration of the particles. Assuming there are D particles in a D-dimensional search space, where each particle represents a potential solution, then the position of the i^{th} particle can be denoted as $x_{id} = \{x_{i1}, x_{i2}, \ldots, x_{iD}\}$, and the velocity of the i^{th} particle can be denoted as $v_{id} = \{v_{i1}, v_{i2}, \ldots, v_{iD}\}$. These particles update their velocity and position using the following formula:

$$v_{id}^{k+1} = wv_{id}^{k} + c_1 r_1 (p_{id,pbest}^{k} - x_{id}^{k}) + c_2 r_2 (p_{d,gbest}^{k} - x_{id}^{k}) \quad (1)$$

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1} \quad (2)$$

where i represents the particle index; k is the current number of iterations; w is the inertia weight; c_1 and c_2 are learning factors; r_1 and r_2 are two random numbers in the range of (0, 1) to ensure the diversity of the group; $p_{id,pbest}^{k}$ is the historical best position of particle i in dimension d during iteration k; and $p_{d,gbest}^{k}$ is the historical best position of the group in dimension d during iteration k.

The IPSO employs an ingenious integration of the principles of the immune system into the foundational structure of the PSO. This approach utilises the concept of immune memory to ensure the preservation of superior solutions encountered during previous searches. The population is diversified through the use of cloning and mutation operations.

3.3.2. Dynamic Adjustment of Inertia Weight

The inertia weight w determines the extent to which the particle's historical velocity influences the current velocity update. In the early stages of the iteration, a larger inertia weight enables particles to extricate themselves from local minimum points, thereby fostering a broader global search. Conversely, a smaller inertia weight in the later stages of the iteration aids particles in executing precise local searches. Accordingly, a nonlinear decreasing strategy of inertia weight is adopted, and two random numbers, α, and β, which obey a uniform distribution from -0.1 to 0.1, are added. Consequently, the inertia weight is subjected to a certain degree of randomness at each iteration, as given in Equation (3).

$$w = (w_{max} - w_{min}) \times cos((\frac{t - T_{max}}{T_{max}})^2 \times \frac{\pi}{2}) + w_{min} + \alpha\beta \quad (3)$$

where t represents the iterations; T_{max} represents the maximum number of iterations; and w_{max} and w_{min} represent the maximum and minimum values of inertia weight, respectively.

The change curve of inertia weight is shown in Figure 4. It can be observed that the inertia weight, following nonlinear treatment, exhibits a gradual decrease in waves slowly in the early stages. This is conducive to enhancing the capability of global searches. In contrast, in the later stages, the decrease gradually accelerates, which helps to carry out more adequate local searches.

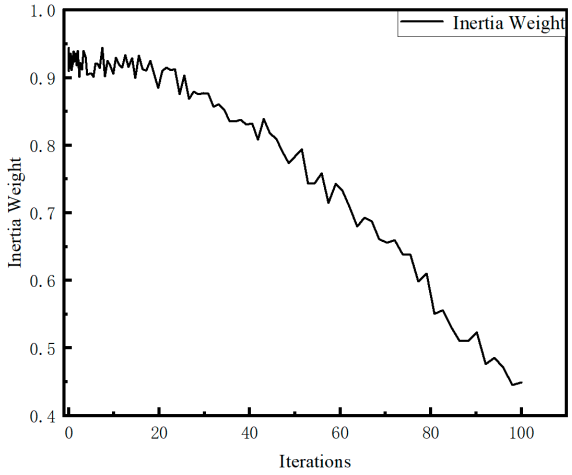


Figure 4. Change curve of inertia weight.

3.3.3. Dynamic Adjustment of Learning Factors

The learning factors c_1 and c_2 in IPSO influence the self-learning capability of particles, and their capability to extract insights from the superior particles within the population. Generally, the values of c_1 and c_2 are fixed, which may not be sufficient to accommodate the complex and variable requirements. Consequently, a nonlinear strategy based on the inertial weight to optimise the learning factors is employed, as given in Equation (4).

$$\begin{cases} c_1(w) = (c_{max} - c_{min}) \times sin(\frac{(e^w-1)^2}{2}) \\ c_2(w) = 2 - c_1(w) \\ -\eta \le c_{k+1}(i+1) - c_k(i) \le \eta \end{cases} \tag{4}$$

where c_{max} and c_{min} represent the maximum and minimum values of learning factors, respectively; and η represents the acceleration rate, which can inhibit the disparity between the iterative learning factors from becoming excessively large.

The change curve of the learning factors is shown in Figure 5. In the early stages of the iteration process, the larger inertia weight w and individual learning factor c_1, in conjunction with the smaller social learning factor c_2, enhance the global exploration capability of the particles. This prevents the algorithm from becoming trapped in a local optimum. In the later stages of the iteration process, the smaller inertia weight w and individual learning factor c_1, in conjunction with the larger social learning factor c_2, enhance the refined search ability of particles and accelerate the convergence of the algorithm to the global optimal solution.

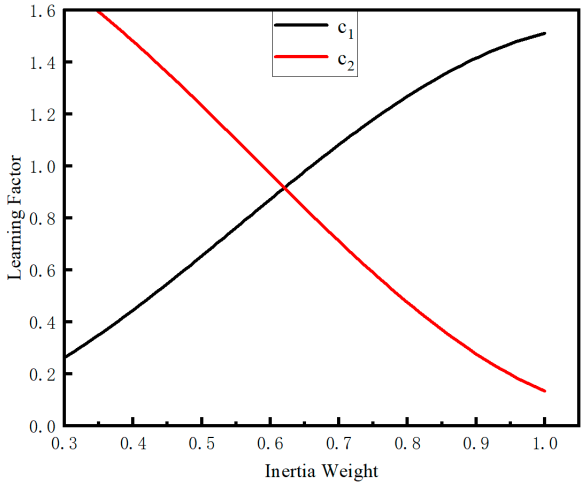


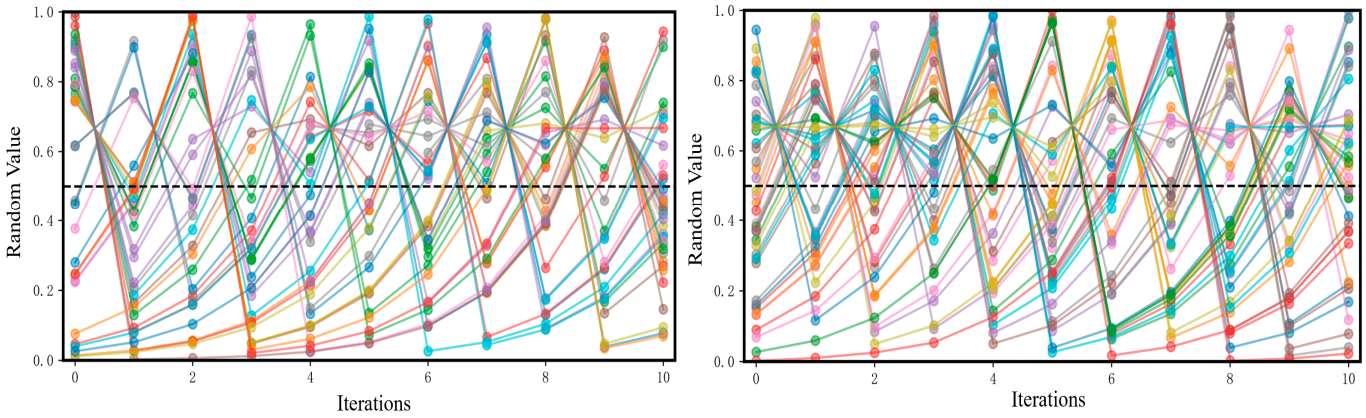
Figure 5. Change curve of learning factors.

3.3.4. Incorporation of Chaos Factors into Particle Velocity

Bernoulli mapping [21], a one-dimensional iterative mapping, encounters the issue of unstable periodic points throughout its iteration process. In order to broaden the mapping space while preserving the benefits of uniform distribution, a random variable $rand(0,1)/T_{max}$ is added into the original mapping equation, as given in Equation (5). The contrast effect is shown in Figure 6, where it can be observed that Bernoulli mapping with random variables improves its traversal ability in space and can generate more uniform random sequences.

$$R_{t+1} = \begin{cases} \frac{R_t}{1-\alpha} + \frac{rand(0,1)}{T_{max}} & 0 \le R_t \le 1-\alpha \\ \frac{R_t-1+\alpha}{\alpha} + \frac{rand(0,1)}{T_{max}} & 1-\alpha < R_t \le 1 \end{cases} \tag{5}$$

where t represents the number of iterations; T_{max} represents the maximum number of iterations; R_t represents the mapping value; and α represents the mapping parameter, set to 0.5.



(**a**) (**b**)

Figure 6. Contrast effect of improved Bernoulli algorithm. (**a**) Distribution before treatment; (**b**) distribution after treatment.

The conventional random numbers r_1 and r_2 in Equation (1) do not guarantee search diversity. The random numbers are replaced with the chaos factors produced by the improved Bernoulli mapping, as given in Equation (6). The random perturbations induced by the chaos factors can enhance the randomness of particle's motion, thereby facilitating the particle's escape from the local optimal solution.

$$v_{id}^{k+1} = wv_{id}^{k} + c_1 R(p_{id,pbest}^{k} - x_{id}^{k}) + c_2(1-R)(p_{d,gbest}^{k} - x_{id}^{k}) \tag{6}$$

3.3.5. Optimization of Particle Swarm

To some extent, enhancing learning factors and particle positions exhibits a superimposition effect. While this accelerates the algorithm's convergence speed, it simultaneously diminishes particle diversity. In light of Kong's proposal of the adaptive mutation probability [22], the adaptive mutation probability α is adopted to maintain the richness of the particle swarm, as given in Equation (7). As illustrated in Figure 7, the value of α increases nonlinearly with the number of iterations, resulting in a reduction in the probability that the random number β from 0 to 1 is greater than α. This limits the update of the optimal particles, preventing them from being assimilated by other particles and ensuring the diversity of the particle swarm. The process of particle optimisation is shown in Figure 8.

$$a = \sqrt{e^{1.2\times\frac{t}{Tmax}}} - 0.8 \tag{7}$$

where t represents the number of iterations, and T_{max} represents the maximum number of iterations.

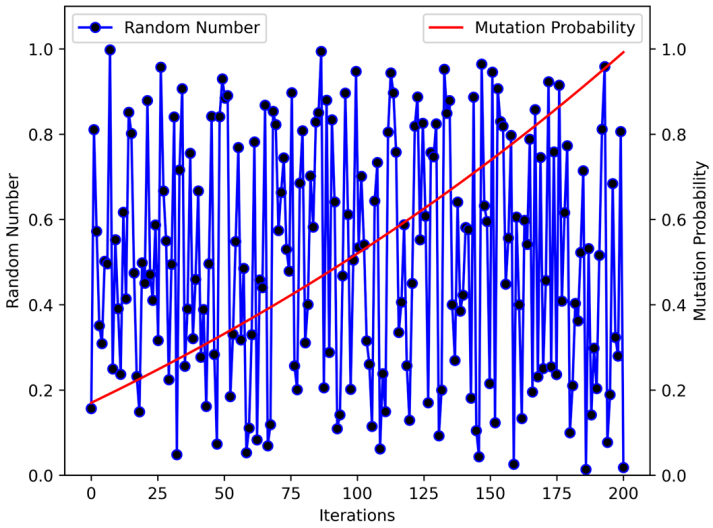


Figure 7. Comparison of mutation probability and random number.

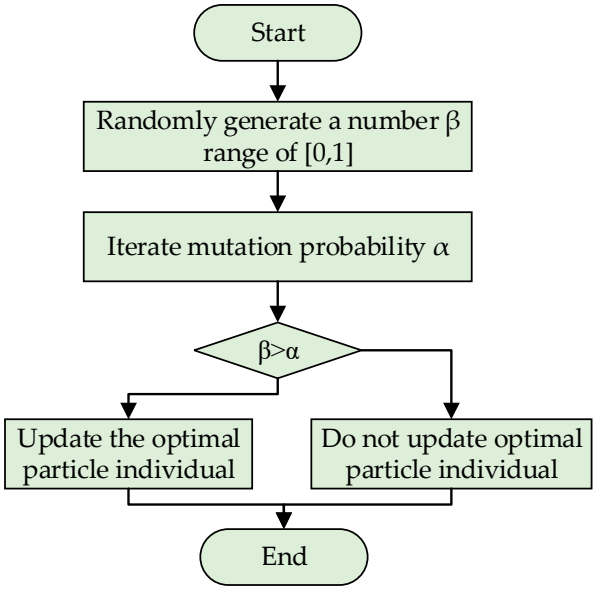


Figure 8. Process of particle optimization.

3.4. Model Framework

Stacking is an ensemble learning technique employed in machine learning with the objective of enhancing the overall performance of models by integrating prediction results from various stages. Initially, a number of base models are selected to create an initial layer of the learning set. Subsequently, each of the aforementioned models is trained individually using the available data. Secondly, the predicted outcomes generated by these base models are utilised as new input features, while the original labels are retained in their original form. This process results in the formation of a novel dataset, which is subsequently trained by the meta-learner. Finally, the meta-learner integrates the predictive information provided by each base model to generate a comprehensive prediction outcome. The stacking framework is depicted in Figure 9.

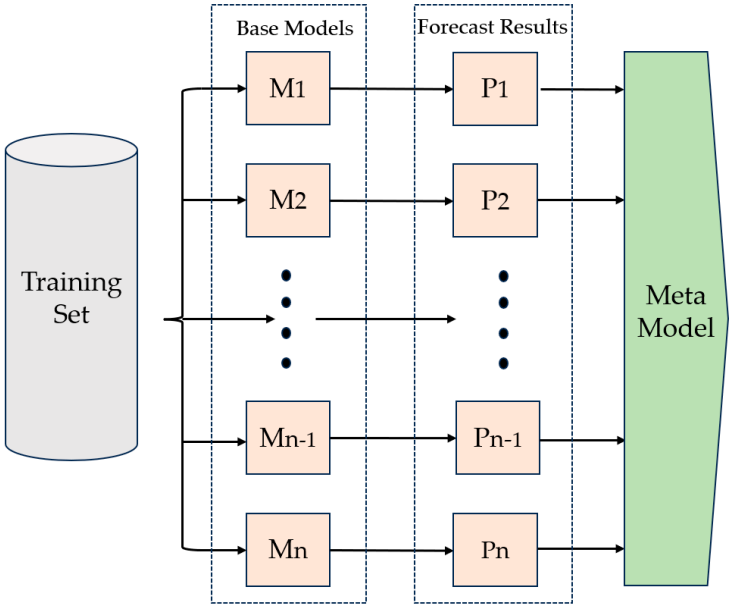


Figure 9. Stacking framework.

Gradient Boosting Decision Tree (GBDT) [23], eXtreme Gradient Boosting (XGBoost) [24], Categorical Boosting (CatBoost) [25], and Light Gradient Boosting Machine (LightGBM) [26] were selected as the base models. These models, all rooted in the gradient boosting algorithm, are designed to offer distinct advantages in various aspects, such as feature engineering, the optimisation of loss functions, and overfitting prevention. At the same time, in order to prevent overfitting, the output of the base model is the classification probability, rather than the direct output of the category label. The classification probability provides the confidence of each base classifier for each category, which is a more informative representation than simply providing the final category label. This study is a multi-classification task, and the output of all base models is superimposed to produce a $n \times 28$ classification probability matrix, as shown in Table 1. The 28 columns of the matrix represent the classification probability of the four classifiers. In the selection of the meta model, the output of the base model is linearly related to the final classification label. Therefore, the Softmax regression model [27], which is more suitable for linear relationships and easy to interpret, is selected.

Table 1. Classification probability.

XGBoost-Defect1	XGBoost-Defect1	XGBoost-Defect1	...	LightGBM-Defect6	LightGBM-Defect7
0.2	0.2	0.1	...	0.1	0.3
⋮	⋮	⋮	⋮	⋮	⋮
0.3	0.1	0.1	...	0.1	0.1
0.4	0.3	0.2	...	0.3	0.2

This study presents a comprehensive framework for the prediction of defects in strip steel. The framework is structured as follows:

Firstly, at the data processing stage, the ISmote algorithm is employed to address the issue of data imbalance. This ensures that the training data are balanced and that the model does not become biased toward the majority class.
Secondly, the model based on stacking is trained using the balanced dataset obtained from the data processing stage. In order to enhance the performance of the model further, the hyperparameters of the model are optimised by utilising the IIPSO algorithm.
Finally, the optimised model is employed to accurately identify strip steel defects.

The defect prediction model framework is shown in Figure 10.

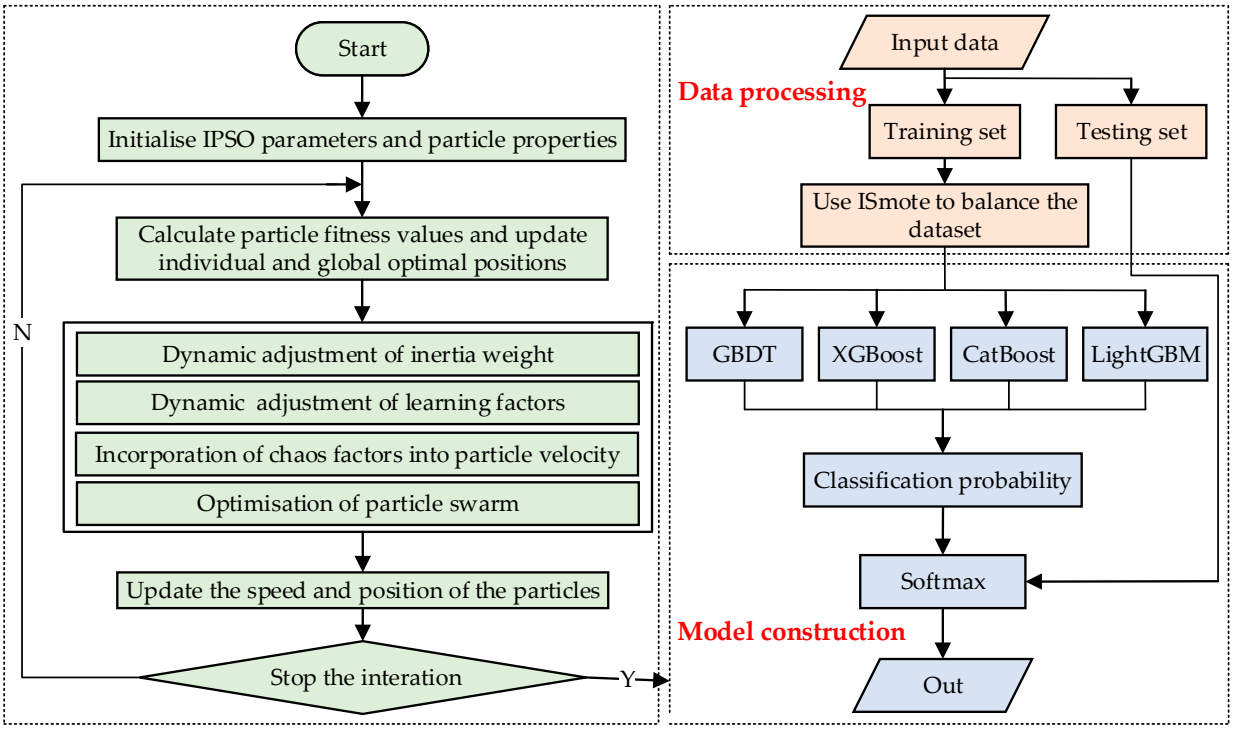


Figure 10. Model framework.

4. Experiment

4.1. Evaluation Metrics

In order to evaluate the performance of a model in a binary classification prediction task, four metrics are employed: Precision, Recall, F1, and G-mean. These metrics reveal the model's predictive performance for different sample types, offering evaluations from a range of perspectives, as given in Equations (8)–(11). In addition, AUC, which stands for Area Under the Receiver Operating Characteristic Curve (ROC), is a performance measurement for classification problems at various threshold settings, as given in Equation (12). Each of these metrics is defined on a scale from 0 to 1, with higher values indicating a more effective model performance.

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

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

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{10}$$

$$G - mean = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}} \tag{11}$$

$$AUC = \frac{1}{2}\left(1 + \frac{TP}{TP + FN} - \frac{FP}{FP + TN}\right) \tag{12}$$

where TP, FP, and FN represent the number of true positives, false positives, and false negatives, respectively.

The prediction of defects in strip steel is a multi-classification task. If the binary classification evaluation index is directly applied, the model performance evaluation is susceptible to bias due to the number of dominant category samples. To avoid this bias, a macro-average strategy is employed when appraising the performance of the multi-class

model. This strategy involves the computation of evaluation metrics for each category independently, followed by the calculation of the arithmetic mean of these metrics. This approach provides a comprehensive measurement of the overall model performance, as given in Equations (13)–(16).

$$Macro_Precision = \frac{1}{N}\sum_{i=1}^{N} Precision_i \tag{13}$$

$$Macro_Recall = \frac{1}{N}\sum_{i=1}^{N} Recall_i \tag{14}$$

$$Marco_F1 = \frac{2 \times Macro_Precision \times Macro_Recall}{Macro_Precision + Macro_Recall} \tag{15}$$

$$Macro_AUC = \frac{1}{N}\sum_{i=1}^{N} AUC_i \tag{16}$$

4.2. ISmote Algorithm Performance Test

In order to verify the efficacy of the ISmote algorithm in practical applications, comparative experiments were conducted on unbalanced data sets with the Smote algorithm, Borderline-Smote [28] algorithm, SVMSmote [29] algorithm, ADASYN [30] algorithm, RSmote algorithm, and Smote-IPF algorithm. The experimental results are shown in Figure 11. While the Smote algorithm alleviates the issue of data imbalance to a certain extent, the newly synthesised samples that invade the majority class area result in the production of noise samples and the overlapping of these with the original minority class samples. Three algorithms, Borderline-Smote, SVMSmote, and ADASYN, all concentrate on the boundary region for sample generation. Nevertheless, the newly synthesised samples still exhibit some degree of overlap with the majority of classes, particularly in the boundary region, where the issue of noisy samples is particularly pronounced. Furthermore, although the RSmote algorithm and the Smote-IPF algorithm are capable of avoiding the generation of noise to a certain extent, the newly synthesised samples lack sufficient diversity and are unable to fully expand the coverage of the minority class in space. In contrast, the ISmote algorithm not only alleviates the imbalance problem but also effectively prevents the newly synthesised samples from invading the majority class area, significantly reducing the impact of noise. Moreover, the newly synthesised samples exhibit a high degree of diversity, thereby providing the classifier with a greater quantity of classification information.

In order to conduct a more objective evaluation, a series of comparative experiments were conducted on five imbalanced datasets sourced from the public KEEL database. The detailed information is presented in Table 2. In the experimental session, the Random Forest (RF) [31] algorithm and the K-Nearest Neighbor (KNN) [32] algorithm were employed to train data processed by various sampling methods. This study employs key performance metrics, including the F1 score, G-mean, and AUC to assess the performance variations of different sampling algorithms when applied to the same classifier.

As demonstrated in Tables 3 and 4, an analysis of the results reveals that the ISmote algorithm outperforms other sampling methods in terms of performance. In particular, the ISmote algorithm achieved the highest F1 score, G-mean, and AUC across multiple datasets for both KNN and RF classifiers. This indicates that the ISmote algorithm significantly enhances the generalisation ability of classifiers by increasing the diversity of minority class samples, which effectively enhances the overall classification performance of the model, particularly when dealing with highly imbalanced datasets. Nevertheless, it should be noted that the ISmote algorithm does not consistently outperform other sampling methods across all datasets. The distinctive characteristics of each dataset, including its features and feature spaces, can influence the effectiveness of any sampling method.

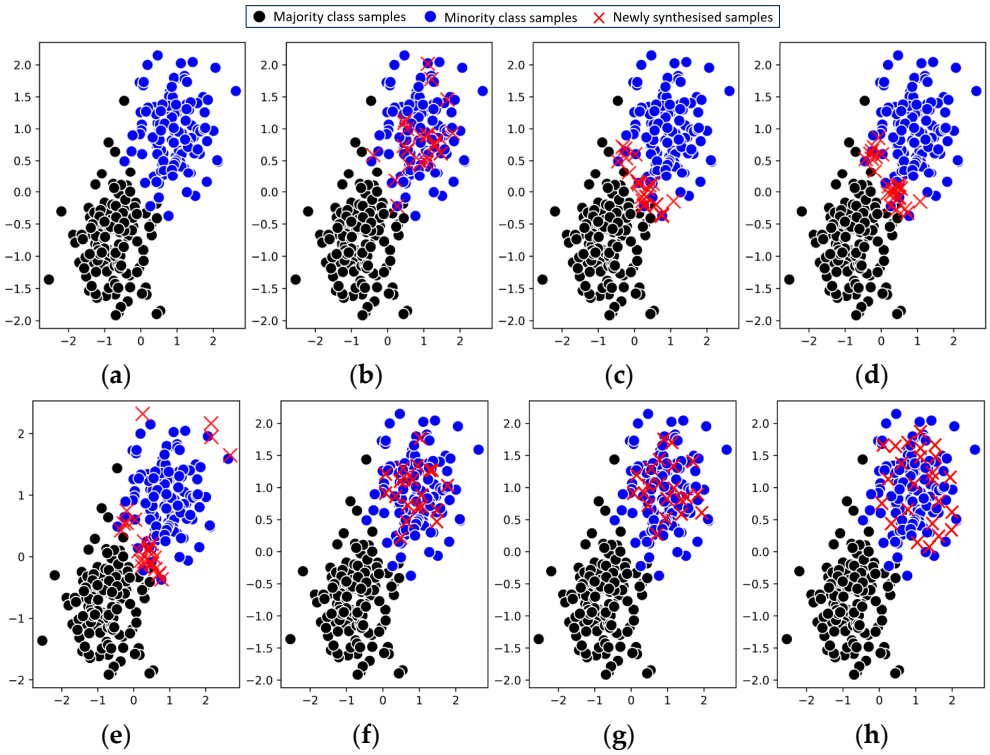


Figure 11. Effect of synthesising samples. (**a**) Original data; (**b**) Smote algorithm; (**c**) Borderline-Smote algorithm; (**d**) SVMSmote algorithm; (**e**) ADASYN algorithm; (**f**) Smote-IPF algorithm; (**g**) RSmote algorithm; and (**h**) ISmote algorithm.

Table 2. Information on public datasets.

Dataset	Sample Number	Feature Dimension	Imbalance Rate/%
WIS	683	9	1.86
LED	443	7	10.97
ECO	336	7	15.80
YEA5	1484	8	32.73
YEA6	1484	8	41.40

Table 3. Performance of each algorithm on KNN.

Dataset	Index	Smote	Borderline-Smote	SVMSmote	ADASYN	Smote-IPF	RSmote	ISmote
WIS	F1	0.945	0.944	0.931	0.952	0.949	0.961	0.971
	G-mean	0.971	0.973	0.964	0.973	0.968	0.963	0.984
	AUC	0.971	0.973	0.964	0.973	0.968	0.963	0.984
LED	F1	0.601	0.592	0.526	0.620	0.711	0.538	0.761
	G-mean	0.657	0.745	0.654	0.810	0.792	0.621	0.885
	AUC	0.715	0.810	0.705	0.841	0.837	0.717	0.892
ECO	F1	0.828	0.792	0.691	0.785	0.795	0.853	0.844
	G-mean	0.949	0.946	0.931	0.939	0.949	0.941	0.950
	AUC	0.949	0.939	0.936	0.943	0.950	0.942	0.953
YEA5	F1	0.639	0.626	0.523	0.613	0.634	0.651	0.671
	G-mean	0.932	0.935	0.921	0.929	0.933	0.935	0.930
	AUC	0.931	0.936	0.919	0.938	0.941	0.929	0.945
YEA6	F1	0.366	0.414	0.319	0.343	0.402	0.482	0.534
	G-mean	0.721	0.777	0.764	0.688	0.712	0.681	0.812
	AUC	0.815	0.784	0.792	0.824	0.831	0.824	0.851

Table 4. Performance of each algorithm on RF.

Dataset	Index	Smote	Borderline-Smote	SVMSmote	ADASYN	Smote-IPF	RSmote	ISmote
WIS	F1	0.950	0.935	0.941	0.928	0.951	0.944	0.959
	G-mean	0.970	0.962	0.961	0.971	0.959	0.960	0.976
	AUC	0.971	0.970	0.969	0.964	0.959	0.961	0.979
LED	F1	0.691	0.742	0.661	0.702	0.796	0.811	0.858
	G-mean	0.901	0.866	0.862	0.859	0.892	0.901	0.921
	AUC	0.919	0.881	0.871	0.879	0.920	0.912	0.931
ECO	F1	0.829	0.831	0.801	0.821	0.826	0.869	0.882
	G-mean	0.931	0.934	0.925	0.921	0.933	0.939	0.949
	AUC	0.939	0.942	0.929	0.931	0.921	0.940	0.951
YEA5	F1	0.514	0.508	0.455	0.520	0.529	0.525	0.565
	G-mean	0.931	0.933	0.896	0.938	0.911	0.939	0.922
	AUC	0.932	0.940	0.914	0.941	0.921	0.945	0.939
YEA6	F1	0.442	0.435	0.392	0.329	0.407	0.436	0.527
	G-mean	0.813	0.820	0.793	0.804	0.792	0.752	0.872
	AUC	0.839	0.822	0.841	0.822	0.791	0.841	0.852

4.3. IIPSO Algorithm Performance Test

In order to verify the performance of the IIPSO algorithm, it was compared with the IPSO algorithm, the bald eagle search algorithm (BES) [33], the sparrow search algorithm (SSA) [34], the IIPSO-W [35] algorithm, and the IIPSO-L algorithm in the experiment. In a simulation experiment, five benchmark test functions were found for testing. The definition, value range, and theoretical optimal of the aforementioned test functions are listed in Table 5.

Table 5. Standard test functions.

Functional Form	Value Range	Theoretical Optimal
$f_1(x) = \sum_{i=1}^{30} x_i^2$	$[-100, 100]$	0
$f_2(x) = \sum_{i=1}^{30} (x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-5.12, 5.12]$	0
$f_3(x_1, x_2) = \frac{\sin\sqrt{x_1^2+x_2^2}-0.5}{(1+0.001(x_1^2+x_2^2))^2} + 0.5$	$[-5, 5]$	0
$f_4(x) = -\sum_{i=1}^{30} (x_i \sin(\sqrt{\lvert x_i \rvert}))$	$[-500, 500]$	0
$f_5(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\frac{x_i}{\sqrt{i}} + 1$	$[-600, 600]$	0

The value of the mean and standard deviation of search results were recorded to evaluate the performance of different algorithms in terms of search accuracy and stability. The results of function verification testing are shown in Table 6. The mean value of IIPSO is the smallest among all algorithms compared, indicating that it has superior precision in seeking the optimal solution. With regard to stability, the standard deviation of IIPSO for the optimisation results of function f_3 is slightly higher than that of IIPSO-L. However, IIPSO has the lowest standard deviation for optimising other functions. This indicates that IIPSO has a high degree of stability in the convergence path when solving unipolar and multipolar problems. Furthermore, the time complexity of the algorithm, that is to say, the response time, is also taken into account. The response time of IIPSO is demonstrably shorter than that of IPSO, thereby demonstrating that the enhancement of IPSO has effectively enhanced the efficiency of the algorithm. Concurrently, the response time of IIPSO is the shortest when compared with other algorithms.

The simulation diagram and evolution process curves of five test functions are shown in Figure 12. It demonstrates that IIPSO significantly outperforms other algorithms with regard to convergence speed when testing functions f_1~f_2. For the test function f_3~f_5, IPSO, IIPSO-L, and IIPSO-W all have faster convergence speeds, but IIPSO still maintains high competitiveness. With regard to the ability to escape from local optima, it can be observed that SSA, BES, and IPSO exhibit certain limitations, often becoming trapped in local optima without the ability to escape. Nevertheless, even though IIPSO occasionally falls into the local optima, it can swiftly move out and locate the global optimum, thereby exhibiting superior performance.

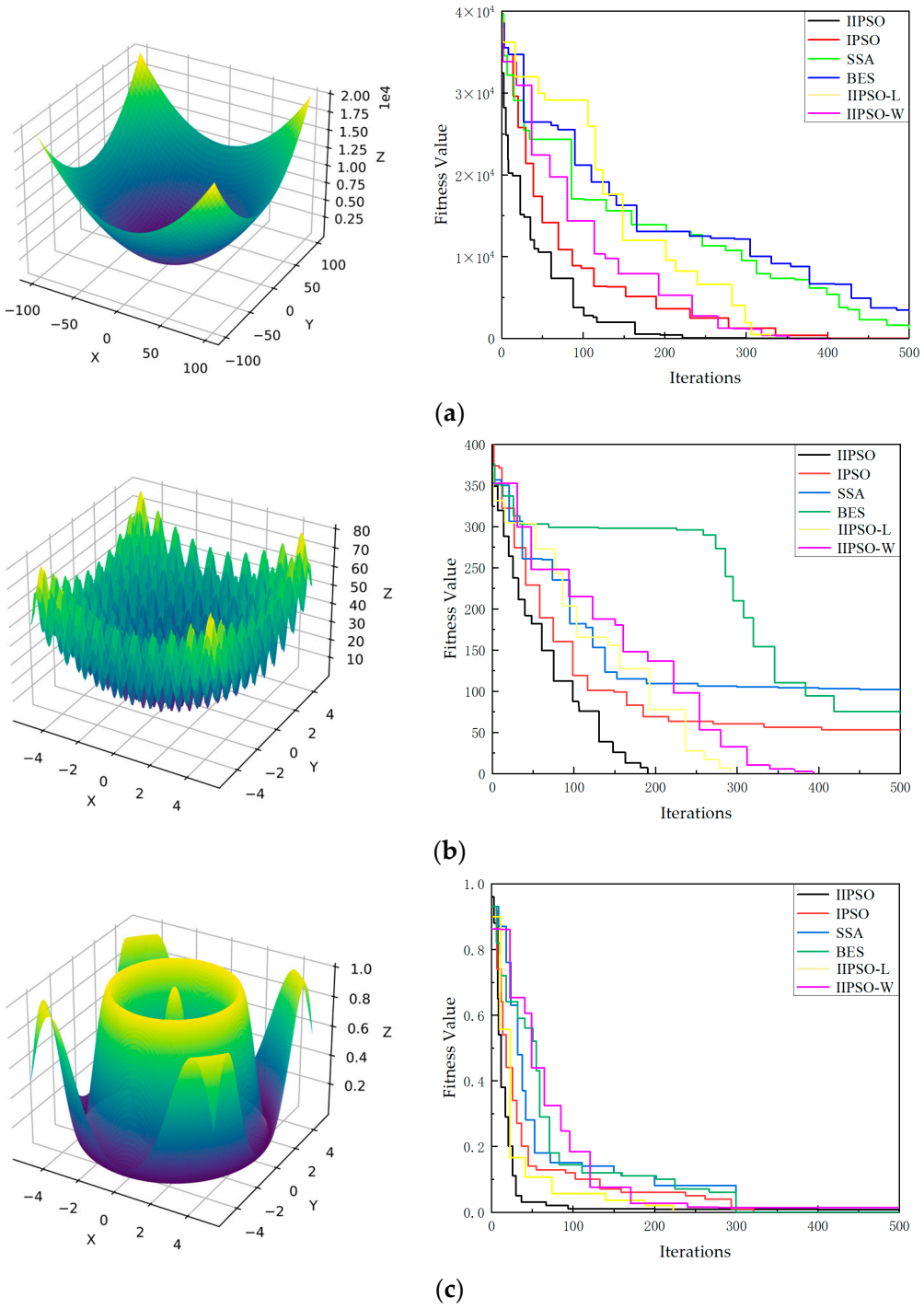


Figure 12. *Cont.*

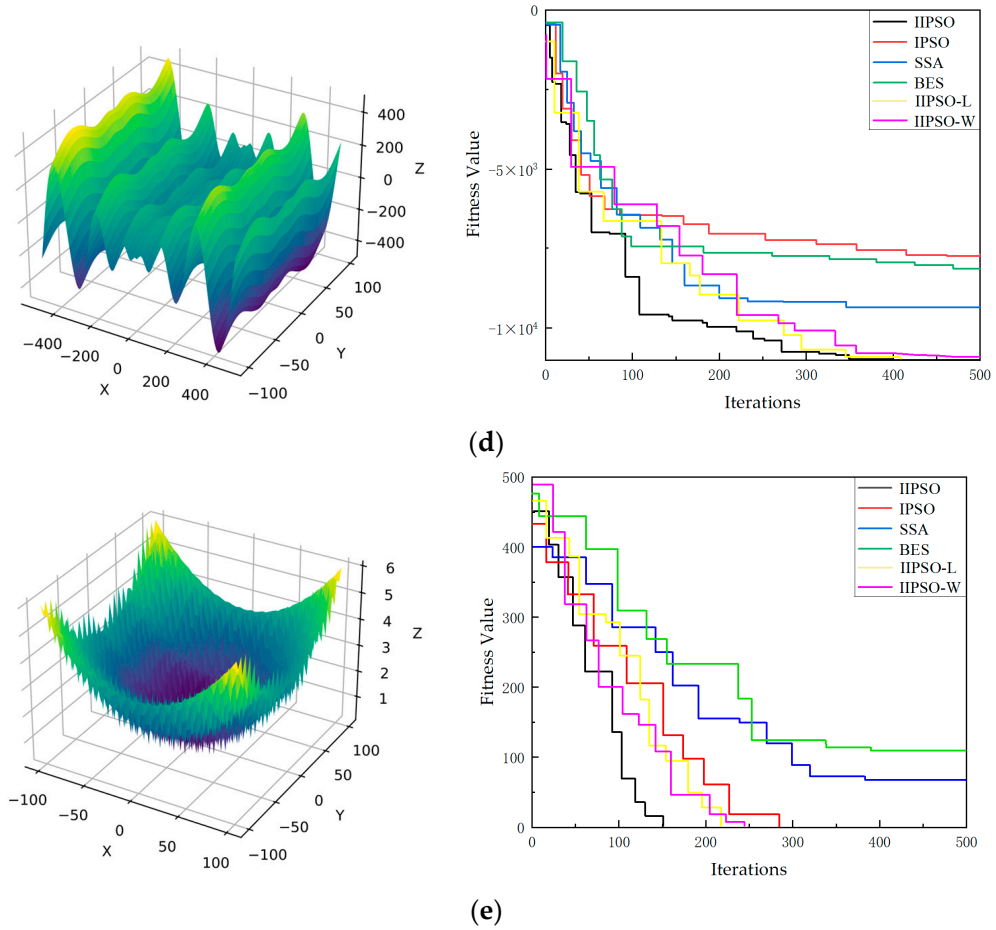


Figure 12. Evolution curve of test functions: (**a**) fitness value of f_1 function; (**b**) fitness value of f_2 function; (**c**) fitness value of f_3 function; (**d**) fitness value of f_4 function; and (**e**) fitness value of f_5 function.

Table 6. Simulation results of standard function.

Function	Index	IIPSO	IPSO	BES	SSA	IIPSO-L	IIPSO-W
f_1	Mean	6.66×10^{-34}	3.30×10^{-2}	3.59×10^{-3}	1.31×10^{-8}	2.50×10^{-29}	1.57×10^{-1}
	Standard Deviation	1.34×10^{-33}	6.15×10^{-3}	4.36×10^{-3}	7.15×10^{-1}	7.46×10^{-29}	1.78×10^{-1}
	Response time/S	$2.76 \times 10^{+0}$	$2.97 \times 10^{+0}$	$4.88 \times 10^{+0}$	$4.24 \times 10^{+0}$	$3.77 \times 10^{+0}$	$3.97 \times 10^{+0}$
f_2	Mean	8.88×10^{-16}	$2.78 \times 10^{+0}$	1.72×10^{-2}	$6.53 \times 10^{+1}$	$2.40 \times 10^{+0}$	3.49×10^{-2}
	Standard Deviation	1.63×10^{-10}	$5.21 \times 10^{+0}$	1.93×10^{-2}	$6.12 \times 10^{+0}$	8.69×10^{-1}	2.16×10^{2}
	Response time/S	$4.08 \times 10^{+0}$	$5.23 \times 10^{+0}$	$6.45 \times 10^{+0}$	$8.98 \times 10^{+0}$	$4.42 \times 10^{+0}$	$4.94 \times 10^{+0}$
f_3	Mean	1.75×10^{-26}	3.00×10^{-12}	6.22×10^{-8}	9.75×10^{-12}	1.13×10^{-28}	5.42×10^{-15}
	Standard Deviation	2.43×10^{-26}	9.24×10^{-11}	8.40×10^{-8}	6.24×10^{-12}	7.55×10^{-27}	3.62×10^{-15}
	Response time/S	$1.87 \times 10^{+0}$	$2.22 \times 10^{+0}$	$3.44 \times 10^{+0}$	$3.78 \times 10^{+0}$	$2.01 \times 10^{+0}$	$2.44 \times 10^{+0}$
f_4	Mean	8.60×10^{-13}	5.20×10^{-1}	6.99×10^{-5}	4.58×10^{-6}	1.89×10^{-9}	4.94×10^{-7}
	Standard Deviation	0.57×10^{-12}	$4.90 \times 10^{+0}$	7.83×10^{-6}	4.28×10^{-6}	2.35×10^{-9}	3.92×10^{-7}
	Response time/S	$7.61 \times 10^{+0}$	$10.2 \times 10^{+0}$	$8.41 \times 10^{+0}$	$9.59 \times 10^{+0}$	$8.01 \times 10^{+0}$	$9.01 \times 10^{+0}$
f_5	Mean	1.74×10^{-18}	3.22×10^{-10}	1.32×10^{-5}	4.82×10^{-7}	2.66×10^{-15}	3.21×10^{-12}
	Standard Deviation	3.84×10^{-18}	5.06×10^{-10}	2.33×10^{-5}	2.09×10^{-7}	4.21×10^{-15}	3.47×10^{-12}
	Response time/S	$11.42 \times 10^{+0}$	$13.88 \times 10^{+0}$	$14.21 \times 10^{+0}$	$14.88 \times 10^{+0}$	$13.2 \times 10^{+0}$	$12.59 \times 10^{+0}$

4.4. Result of Defect Classification Prediction

In order to verify that the framework in this paper has a better predictive efficacy with respect to strip steel defects, a simulation experiment was conducted. In the experiment, a total of ten prediction models were constructed, namely, SVM (version 1.0.2), RF (version 1.0.2), GBDT (version 1.0.2), XGBoost (version 2.1.0), CatBoost (version 1.2.5), LightGBM (version 4.4.0), Stacking, ISmote_Stacking, IPSO_ISmote_Stacking, and IIPSO_ISmote_Stacking. These were trained and evaluated independently under the same experimental conditions. As

shown in Figure 13, in terms of response speed, the IIPSO_ISmote_Stacking model exhibits a distinct advantage, with a response time of 0.256 s, which is superior to the 0.284 s and 0.323 s observed in the IPSO_ISmote_Stacking model and the stacking model. The optimisation process results in a reduction in the response time of the model when processing tasks. Concurrently, although the stacking model incorporates multiple prediction models internally, it maintains a high response speed, indicating that the model can still provide feedback on prediction results in a timely manner despite increasing integration complexity. Additionally, due to the complexity of the inherent structure of the stacking model, the CPU usage and memory usage during operation are slightly higher than those of the single model. However, these values remain within a reasonable range.

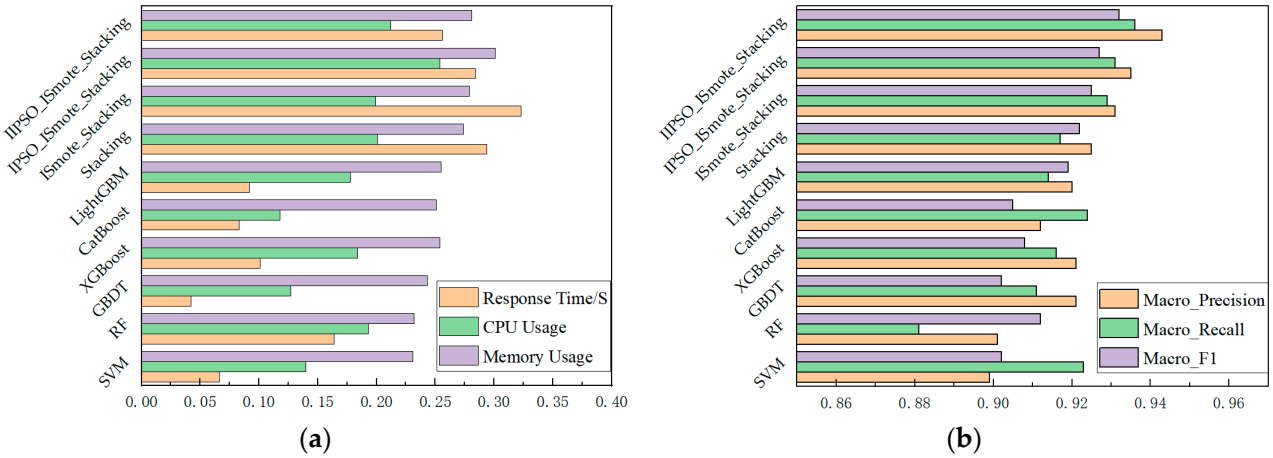


Figure 13. Comparison of the prediction: (**a**) response time and system resource usage; (**b**) prediction performance.

From the perspective of evaluation metrics, it can be observed that the stacking model demonstrates a significantly enhanced prediction performance in comparison to a single model. Following the implementation of data balancing through the ISmote algorithm and hyperparameter optimisation through the IIPSO algorithm, the performance of the stacking model was significantly enhanced. The value of Macro_Precision, Macro_Recall, and Macro_F1 scores reached 0.943, 0.936, and 0.932, respectively, with each indicator value attaining the optimal level. These results demonstrate that the optimised stacking ensemble model exhibits enhanced prediction accuracy. Secondly, the results demonstrate that the ISmote algorithm is capable of effectively enhancing the model's classification performance, as well as the IIPSO algorithm's ability to identify optimal hyperparameters. Furthermore, the four single models, GBDT, XGBoost, CatBoost, and LightGBM, which serve as the fundamental components of the stacking integrated model, also demonstrate high predictive capabilities. The performance indicators of these models remain at a high level. This indicates that they are suitable as base models in the stacking ensemble framework.

In order to verify the prediction performance of the IIPSO_ISmote_Stacking model for seven strip defects, the ROC curves for each defect were plotted. As shown in Figure 14, all ROC curves are situated above the diagonal reference line, and the value of Macro_AUC is 0.928, which signifies an exemplary predictive performance of the model. In particular, the values of AUC for Defect 1, Defect 3, and Defect 5 in Figure 15 are close to 0.930, indicating that the model has high performance in these categories. Although the values of AUC for the remaining strip steel defects are relatively low, they nevertheless remain at a high level. Consequently, the IIPSO_ISmote_Stacking model demonstrates excellent performance in the application of strip steel defect prediction.

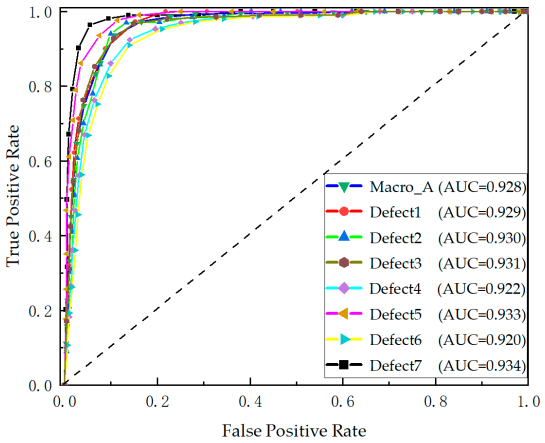


Figure 14. ROC curves of the IIPSO_ISmote_Stacking model.

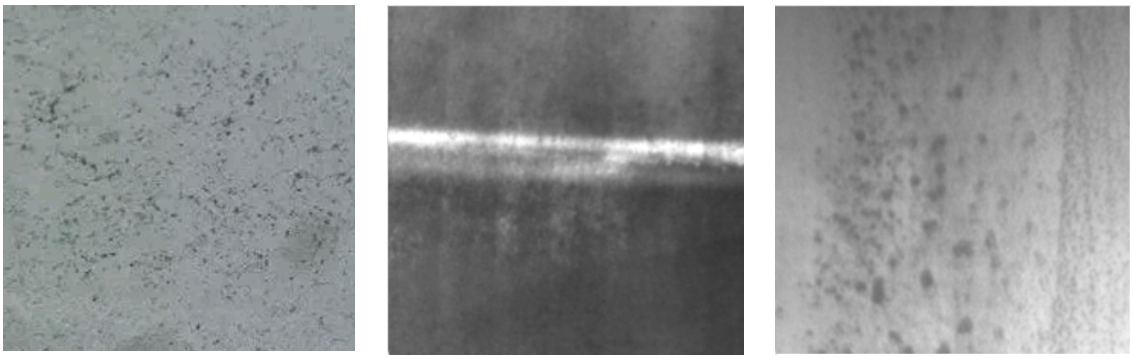


Figure 15. Identified strip steel defects: (**a**) Defect 1; (**b**) Defect 3; and (**c**) Defect 5.

5. Conclusions

This study proposes a novel framework for the prediction of strip steel defects based on a stacking ensemble approach. The framework addresses the issues of class imbalance and overfitting, thereby achieving accurate prediction outcomes. The model framework consists of three key components, which collectively enhance prediction performance.

Firstly, the strategy of K-means++ clustering combined with density estimation was employed to enhance the Smote algorithm. These enhancements significantly reduced the production of noise and provided more classification information. At the data processing stage, the ISmote algorithm was utilised to address the issue of data imbalance.

Secondly, a dynamic parameter adjustment combined with a mutation strategy was employed to enhance the IPSO algorithm. These enhancements facilitated the convergence of the IPSO algorithm, while also preserving the diversity of the solution space, thus mitigating the premature convergence issue commonly encountered in complex optimisation tasks.

Finally, GBDT, XGBoost, CatBoost, and LightGBM were selected to construct a strip steel defect prediction model based on the stacking approach. The IIPSO algorithm was employed for hyperparameter optimisation with the objective of enhancing the model's prediction accuracy and generalisation ability.

The experimental results demonstrate that the proposed model framework exhibits high accuracy and feasibility for application in the prediction of defects in strip steel. Furthermore, it can be concluded that the IIPSO and ISmote algorithms enhanced the performance of the model. Concurrently, the model framework exhibits a high predictive performance for the categories Defect 1, Defect 3, and Defect 5. This implies that the detection outcomes for these defects are more dependable, thereby reducing misjudgements and omissions and enhancing the overall detection accuracy. In conclusion, the proposed

framework represents a novel solution for similar classification problems. Further work could be conducted to examine the potential for additional algorithmic enhancements and to extend the proposed method to a more diverse range of application domains.

The process of feature selection was not carried out in this study. Nevertheless, it is evident that feature selection techniques are of significant value in the removal of superfluous information and the refinement of pivotal features, thereby optimising the performance of the model. In subsequent studies, a range of feature selection techniques may be employed in order to enhance model performance by reducing the feature set.

Author Contributions: Conceptualization and methodology, Z.F.; data curation and formal analysis, Z.F.; experiments and analysis, Z.F.; investigation, Z.F.; validation and visualization, Z.F.; writing—original draft preparation, Z.F.; writing—review and editing, Z.F. and S.Y.; resources and supervision, F.Z.; funding acquisition, S.Y. and B.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially supported by the National Science and Technology Program during the Twelfth Five-year Plan Period (2015BAF10B00) and the Science and Technology Innovation Plan of Shanghai Science and Technology Commission (17511110204).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset utilised in this study, which concerns strip defects, was sourced from the University of California, Irvine (UCI) machine learning repository.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wang, R.; Liang, F.; Mou, X.; Chen, L.; Yu, X.; Peng, Z.; Chen, H. Development of an improved YOLOv7-based model for detecting defects on strip steel surfaces. *Coatings* **2023**, *13*, 536. [CrossRef]
2. Guan, H.; Xu, H.; Cai, L. Requirement Dependency Extraction Based on Improved Stacking Ensemble Machine Learning. *Mathematics* **2024**, *12*, 1272. [CrossRef]
3. Shu, J.; Yu, H.; Liu, G.; Yang, H.; Chen, Y.; Duan, Y. BO-Stacking: A novel shear strength prediction model of RC beams with stirrups based on Bayesian Optimization and model Stacking. *Structures* **2023**, *58*, 105593. [CrossRef]
4. Wang, F.; Yang, Y.; Huang, T.; Xu, Y. Lifetime prediction of electronic devices based on the P-Stacking machine learning model. *Microelectron. Reliab.* **2023**, *146*, 115027. [CrossRef]
5. Khoshkroodi, A.; Parvini Sani, H.; Aajami, M. Stacking Ensemble-Based Machine Learning Model for Predicting Deterioration Components of Steel W-Section Beams. *Buildings* **2024**, *14*, 240. [CrossRef]
6. Thabtah, F.; Hammoud, S.; Kamalov, F.; Gonsalves, A. Data imbalance in classification: Experimental evaluation. *Inf. Sci.* **2020**, *513*, 429–441. [CrossRef]
7. Das, U.K.; Tey, K.S.; Seyedmahmoudian, M.; Mekhilef, S.; Idris, M.Y.I.; Van Deventer, W.; Horan, B.; Stojcevski, A. Forecasting of photovoltaic power generation and model optimization: A review. *Renew. Sustain. Energy Rev.* **2018**, *81*, 912–928. [CrossRef]
8. Zhang, Y.; Deng, L.; Wei, B. Imbalanced Data Classification Based on Improved Random-SMOTE and Feature Standard Deviation. *Mathematics* **2024**, *12*, 1709. [CrossRef]
9. Chen, Y.; Zou, J.; Liu, L.; Hu, C. Improved Oversampling Algorithm for Imbalanced Data Based on K-Nearest Neighbor and Interpolation Process Optimization. *Symmetry* **2024**, *16*, 273. [CrossRef]
10. Arafa, A.; El-Fishawy, N.; Badawy, M.; Radad, M. RN-SMOTE: Reduced Noise SMOTE based on DBSCAN for enhancing imbalanced data classification. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 5059–5074. [CrossRef]
11. Chen, Q.; Zhang, Z.-L.; Huang, W.-P.; Wu, J.; Luo, X.-G. PF-SMOTE: A novel parameter-free SMOTE for imbalanced datasets. *Neurocomputing* **2022**, *498*, 75–88. [CrossRef]
12. Sáez, J.A.; Luengo, J.; Stefanowski, J.; Herrera, F. SMOTE–IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. *Inf. Sci.* **2015**, *291*, 184–203. [CrossRef]
13. Chen, B.Y.; Xia, S.Y.; Chen, Z.; Wang, B.; Wang, G. RSMOTE: A self-adaptive robust SMOTE for imbalanced problems with label noise. *Inf. Sci.* **2021**, *553*, 397–428. [CrossRef]
14. Lai, X.; Tu, Y.; Yan, B.; Wu, L.; Liu, X. A Method for Predicting Ground Pressure in Meihuajing Coal Mine Based on Improved BP Neural Network by Immune Algorithm-Particle Swarm Optimization. *Processes* **2024**, *12*, 147. [CrossRef]
15. Wang, Y.; Jiao, R.; Wei, T.; Guo, Z.; Ben, Y. A Method for Predicting Inertial Navigation System Positioning Errors Using a Back Propagation Neural Network Based on a Particle Swarm Optimization Algorithm. *Sensors* **2024**, *24*, 3722. [CrossRef] [PubMed]

16. Qiao, P.; Sun, C. Research on hybrid flow-shop scheduling problem based on improved immune particle swarm optimization. In Proceedings of the 2011 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), Dengfeng, China, 8–10 August 2011; pp. 4240–4243.
17. Li, H.; Wang, S.; Chen, Q.; Gong, M.; Chen, L. IPSMT: Multi-objective optimization of multipath transmission strategy based on improved immune particle swarm algorithm in wireless sensor networks. *Appl. Soft Comput.* **2022**, *121*, 108705. [CrossRef]
18. Sun, J.; Che, Y.; Yang, T.; Zhang, J.; Cai, Y. Location and Capacity Determination Method of Electric Vehicle Charging Station Based on Simulated Annealing Immune Particle Swarm Optimization. *Energy Eng.* **2023**, *120*, 367–384. [CrossRef]
19. Zhao, D.; Feng, S.; Cao, Y.; Yu, F.; Guan, Q.; Li, J.; Zhang, G.; Xu, T. Study on the classification method of rice leaf blast levels based on fusion features and adaptive-weight immune particle swarm optimization extreme learning machine algorithm. *Front. Plant Sci.* **2022**, *13*, 879668. [CrossRef] [PubMed]
20. Meng, J.; Yu, Z.; Cai, Y.; Wang, X. K-Means++ clustering algorithm in categorization of glass cultural relics. *Appl. Sci.* **2023**, *13*, 4736. [CrossRef]
21. Hu, G.; Zhong, J.; Wei, G. SaCHBA_PDN: Modified honey badger algorithm with multi-strategy for UAV path planning. *Expert Syst. Appl.* **2023**, *223*, 119941. [CrossRef]
22. Kong, W.; Chen, J.; Song, Y.; Fang, Z.; Yang, X.; Zhang, H. Sobel edge detection algorithm with adaptive threshold based on improved genetic algorithm for image processing. *Int. J. Adv. Comput. Sci. Appl.* **2023**, *14*, 2. [CrossRef]
23. Cheng, T.; Chen, G. Prediction of mechanical properties of hot-rolled strip steel based on PCA-GBDT method. *J. Phys. Conf. Ser.* **2021**, *1774*, 012002. [CrossRef]
24. Azmi, S.S.; Baliga, S. An overview of boosting decision tree algorithms utilizing AdaBoost and XGBoost boosting strategies. *Int. Res. J. Eng. Technol.* **2020**, *7*, 6867–6870.
25. Saber, M.; Boulmaiz, T.; Guermoui, M.; Abdrabo, K.I.; Kantoush, S.A.; Sumi, T.; Boutaghane, H.; Nohara, D.; Mabrouk, E. Examining LightGBM and CatBoost models for wadi flash flood susceptibility prediction. *Geocarto Int.* **2022**, *37*, 7462–7487. [CrossRef]
26. Tang, M.; Zhao, Q.; Ding, S.X.; Wu, H.; Li, L.; Long, W.; Huang, B. An improved lightGBM algorithm for online fault detection of wind turbine gearboxes. *Energies* **2020**, *13*, 807. [CrossRef]
27. Abdullayeva, F.J. Advanced persistent threat attack detection method in cloud computing based on autoencoder and softmax regression algorithm. *Array* **2021**, *10*, 100067. [CrossRef]
28. Smiti, S.; Soui, M. Bankruptcy prediction using deep learning approach based on borderline SMOTE. *Inf. Syst. Front.* **2020**, *22*, 1067–1083. [CrossRef]
29. Bagui, S.S.; Mink, D.; Bagui, S.C.; Subramaniam, S. Determining Resampling Ratios Using BSMOTE and SVM-SMOTE for Identifying Rare Attacks in Imbalanced Cybersecurity Data. *Computers* **2023**, *12*, 204. [CrossRef]
30. Chen, Z.; Zhou, L.; Yu, W. ADASYN− Random forest based intrusion detection model. In Proceedings of the 2021 4th International Conference on Signal Processing and Machine Learning, Beijing, China, 18–20 August 2021; pp. 152–159.
31. Wang, J.; Ma, S.; Jiao, P.; Ji, L.; Sun, X.; Lu, H. Analyzing the Risk Factors of Traffic Accident Severity Using a Combination of Random Forest and Association Rules. *Appl. Sci.* **2023**, *13*, 8559. [CrossRef]
32. Uddin, S.; Haque, I.; Lu, H.; Moni, M.A.; Gide, E. Comparative performance analysis of K-nearest neighbour (KNN) algorithm and its different variants for disease prediction. *Sci. Rep.* **2022**, *12*, 6256. [CrossRef]
33. Chhabra, A.; Hussien, A.G.; Hashim, F.A. Improved bald eagle search algorithm for global optimization and feature selection. *Alex. Eng. J.* **2023**, *68*, 141–180. [CrossRef]
34. Gharehchopogh, F.S.; Namazi, M.; Ebrahimi, L.; Abdollahzadeh, B. Advances in sparrow search algorithm: A comprehensive survey. *Arch. Comput. Methods Eng.* **2023**, *30*, 427–455. [CrossRef] [PubMed]
35. Wang, T.; Lan, Y.; Guo, Y. Application of Improved Immune Particle Swarm Algorithm in TDOA Positioning. *Radio Eng.* **2023**, *53*, 1199–1206.

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