A Partition Stacking Classification Framework With Oversampling for Quality Prediction of Aluminum Alloy Ingots

Shen Yan, Haifeng Guo, and Shixin Liu

***Abstract*—As the raw material of aluminum alloy production, the quality of aluminum alloy ingots has an important influence on the quality of final products. We study how to use machine learning models to predict ingot quality by production parameters. However, traditional machine learning models often ignore upstream and downstream relationships and the data imbalance in practical production. In this paper, to solve the above problems, knowledge about the aluminum alloy melting-casting process is applied and the relationship between process parameters of melting-casting process and ingots quality is modeled via a partition classification framework based on standard stacking. The data is divided into three parts according to different processes, and then sequentially input the framework in process order. The major difference between our method and the standard stacking is that, at the base-level, the classifiers' predictions for each part are fused with the next part as the new input. After that, the output of the last part is then fused with all the data for training at the meta-level. We carefully compare the performance of Synthetic Minority Oversampling Technique (SMOTE) variants, and an oversampling method called Polynomial Fitting SMOTE is used in the classification framework to deal with the imbalance problem. We apply our proposed method and obtain the best classification results on aluminum alloy melting-casting data. We also show the validity of the combination of Polynomial Fitting SMOTE and standard stacking on public datasets.**

***Key Words*—Aluminum alloy ingot, Quality prediction, Stacking, Polynomial Fitting SMOTE, Data partition.**

I. Introduction

ALUMINUM alloy is one of the most commonly used nonferrous materials in the industry. Aluminum alloys are light materials with a high strength-to-weight ratio combined with a relatively low melting point, good castability, and low casting shrinkage [1]. Because of these characteristics, they play a very important role in some high-technology fields such as aerospace, transportation, automobile, and in everyday industries such as food packaging [2]. Aluminum production and consumption keep growing steadily at 4-5% a year [3].

The quality of aluminum alloy products is always the most important factor for enterprises. The performance of aluminum alloy products is closely related to the quality of ingots, it can be said that the quality of ingots determines the final product [4]. If the ingots are not up to standard because of defects, then several remedial actions can be taken: conditioning, adjustment of normal operation, reapplication, or scrapping [5]. Any remedial measures result in extra costs.

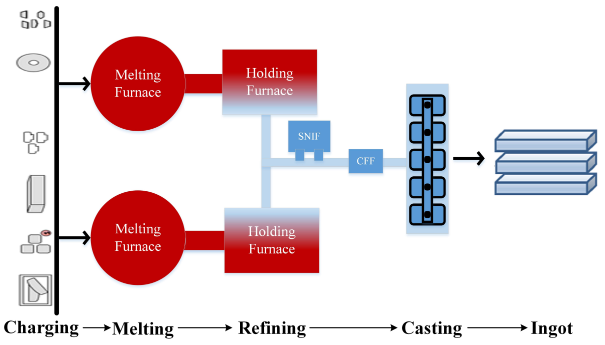

Fig. 1. Melting-casting flow chart of aluminum alloy.

The production of aluminum alloy ingots is a multi-stage process. As shown in Figure 1, the melting-casting process is divided into three stages. Raw materials are first melted in melting furnaces. The resulting liquid aluminum is transferred to holding furnaces for refining. Then liquid aluminum is processed through degassing and filtering devices (SNIF and CFF), cast in a casting machine, and finally formed into ingots. Therefore, the quality of ingots results from the combined effect of each stage. It is important to understand the impact of process parameters on quality. During the whole process, the temperature of the liquid aluminum can significantly affect the quality of the final product. If the pouring temperature is not sufficiently higher than the liquidus temperature, the fluidity of liquid aluminum deteriorates and defects such as porosity or insufficient filling are likely to occur. If the temperature of liquid aluminum is superheated over $800°C$, the oxidation reaction at the melt surface is promoted with the increase of the impurities or cracks. Besides, the dissolved hydrogen content in liquid aluminum should also be carefully controlled. Remarkably, hydrogen gas can be dissolved in the liquid state of aluminum. The dissolved hydrogen in the molten aluminum remaining after solidification can cause defects such as micro-porosity. Also, the dissolved hydrogen diffused into the bifilm gap can work as crack initiators [6]. According to the process knowledge, we consider three major equipments that are vitally important — melting furnace, holding furnace,

This work was supported in part by Liaoning Province Dr. Research Foundation of China under Grant 2024-BSLH-205.

S. Yan and H. Guo are with the School of Science, Shenyang University of Technology, Shenyang 110870, China (e-mail: yanshenneu@163.com, 3098600461@qq.com).

S. Liu is with the College of Information Science and Engineering, Northeast University, Shenyang 110819, China (e-mail: sxliu@mail.neu.edu.cn).

Corresponding Author: Shen Yan.

and casting machine.

It is important to build a relationship model of process parameters and the quality of ingots so that the ingot quality can be predicted prior to production by inputting a specific set of process parameters. Some empirical and numerical models can provide an understanding of the relationship between parameters and quality, but the accuracy is not high. They usually reflect the relationship between ingot quality and a single parameter, which cannot describe the comprehensive law of multiple process parameters on ingot quality.

In recent years, machine learning models have been used as an alternative way to successfully predict aluminum alloy properties. Ingots quality prediction is a typical binary classification problem of machine learning. Unlike general classification problems, aluminum alloy ingot data has the following characteristics due to the specificity of industrial production:

1) Data is imbalanced, that is a dataset in which qualified ingots have a much greater number of samples than unqualified ingots. Although unqualified ingots may cause great losses, the number of them is tiny. The class to which qualified ingots belong is called the majority class, while the class to which unqualified ingots belong is called the minority class [7]. The imbalance problem is common in the industry and particularly challenging to classify with traditional classification models.
2) The data comes from multiple processes and may be suitable for different classifiers. According to the production process, the upstream process has a strong influence on the downstream process, thus a single classifier may not reflect the upstream and downstream relations in practical production.

In this paper, for the above characteristics of aluminum alloy ingot data, we combine an oversampling approach with a novel stacking classification framework. Oversampling is a data-level technique to deal with data imbalance problems [8]. Polynomial Fitting Synthetic Minority Oversampling Technique (PFSMOTE) [9] performs well on multiple datasets. This method finds appropriate samples among minority samples in the training set and then adds synthetic samples using a polynomial fitting function. We use four fitting functions to generate samples and randomly add them to the dataset until the majority and minority samples reach the desired proportion. Then a stacking ensemble framework is constructed. Stacking [10] is a non-linear ensemble predictor in order to achieve higher prediction accuracy and reduce generalization error based on two levels, which are called base-level and meta-level respectively. We design a novel stacking framework that conforms to the production process to describe the upstream and downstream relations. It partitions the data by different process stages and only trains one part at a time. At base-level, the output of each part is fused with the process parameters of the next part and used together as the input for the next training. The output of the last part and all parameters are used as the input to meta-classifier training. Meta-classifier gives the final classification. Data partition not only increases the diversity of ensemble learning but also conforms to the production processes of aluminum alloy ingots. The experimental results show that our classification framework can predict the quality of aluminum alloy ingots accurately when the number of unqualified ingots is small. The contributions of this article are as follows:

1) In order to solve data imbalance problems, four fitting functions are combined to generate minority samples. It is experimentally found to be suitable for dealing with the problems in this paper.
2) A novel classification framework is proposed based on a standard stacking ensemble learning model that explains the melting-casting process.

The rest of this paper is organized as follows. In section II, we review the related work. In Section III, details are given on how the four fitting functions apply. A novel stacking algorithm is proposed to improve the classification accuracy of unqualified ingots. The results of the comparison experiment are presented in Section IV. Finally, Section V concludes this paper.

II. Related Work

A. *Machine Learning in Aluminum Alloy Production*

Machine learning encompasses a broad range of algorithms for modeling based on experiences and making accurate predictions. Machine learning tasks can be categorized as supervised, unsupervised, and reinforcement learning. Among them, supervised learning is widely used in aluminum alloy production, while unsupervised learning and reinforcement learning are rarely studied. Supervised learning models generally form predictions through a learned mapping $f(x)$, which produces an output y for each input x (or a probability distribution over y given x) [11]. Supervised learning tasks in aluminum alloy production can be roughly divided into two categories: regression [12] and classification [13]. Many different forms of mapping f exist, such as Artificial Neural Network [14, 15], Extreme Learning Machine [16] are used for regression and Logistic Regression (LR) [17], k-Nearest Neighbors (kNN) [13], Decision Tree (DT), Support Vector Machine (SVM) [18] are used for classification. As machine learning paradigms, ensemble learning [6, 19] and deep learning [20] are also increasingly being utilized to improve the positive effect. Deep learning often requires large amounts of data. It is not appropriate for our problem. We consider ensemble learning to handle the binary classification task.

In ensemble learning, the combination of different weak classifiers can provide complementary information [21]. It is known that this kind of solution can be used for the improvement of the overall classification from the perspective of accuracy as well as generalization [22].

To generate an ensemble model, two important things should be contemplated. First, sufficient diversity must be introduced in ensemble learning. The second is to determine the output of each single classifier and consider how to integrate these outputs into one [23].

Diversity is the degree to which classifiers make different decisions on one problem. Empirical studies have shown that a larger diversity results in a better recall for the minority but harms the majority classes. When the accuracy is not high enough, it increases the probability of classifying minority samples [23, 24]. In ensemble learning, diversity can be achieved from several aspects: 1) Using different training datasets to train weak classifiers. 2) Using different training parameters for different weak classifiers. 3) Using different weak classifiers. In some common algorithms, such as bagging [25] and boosting [26], their weak classifiers are usually DTs. Such ensemble models are also called homogeneous. Each tree is trained by selecting different samples or different features. After training, different model parameters are obtained for each classifier. For another method to improve diversity, the most common model is stacking.

Stacking is a well-known heterogeneous ensemble learning model. Compared with bagging and boosting, stacking does not manipulate the training set directly. Diversity is achieved because different learning algorithms make different errors in the same dataset. There are no uniform criteria for the selection of classifiers. What kind of models to combine, or even different architectures of the same model is used only for a specific application [10, 27, 28].

When stacking is used in a classification problem, the output of a single classifier can be a label, probability distribution, or entropy [29]. As for how to combine the outputs of single classifiers, there are a lot of studies [30, 31, 32, 33]. Many papers apply stacking to different applications [34, 35, 36, 37, 38]. We need to redesign the stacking framework to be more compatible with the aluminum ingot data.

B. *Imbalance Problem*

The degree of imbalance makes the problem more complex and hinders learning [7]. Traditional classifiers are often unable to deal with imbalance problems [39, 40]. This main issue occurs since the learning process of most classification algorithms is always biased toward the majority class samples, and minority ones are underrepresented [24, 41]. Resampling is one of the common methods to deal with imbalance problems. Depending on the mechanism, resampling methods can be divided into three categories, which are undersampling methods, oversampling methods and hybrids methods [7, 8, 24, 41]. For concrete problems, we need to choose the most appropriate one. There are some empirical guidelines on how to select resampling methods. When there are hundreds of minority samples in the dataset, undersampling is superior to oversampling in terms of computational time. When there are only a few dozen minority samples, oversampling is found to be a better choice [42, 43]. Considering that the sample size of the aluminum alloy dataset is not big and the number of minority samples is also small, oversampling is more suitable for our problem.

Regarding oversampling, SMOTE is a powerful method that has shown a great deal of success in various applications [44, 45]. It creates synthetic data based on the feature space similarities between existing minority samples. A synthetic sample x is a point along the line segment joining x_{min} under consideration and a randomly selected nearest neighbor x_{kNN} It is generated according to (1),

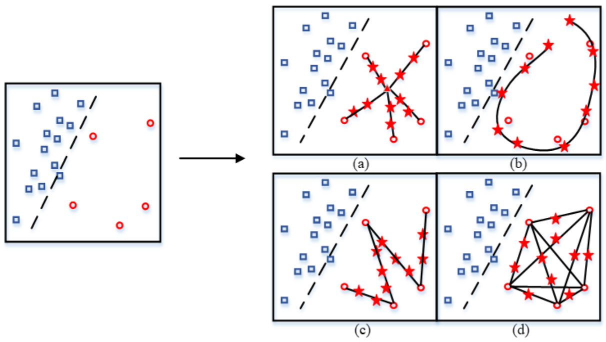

Fig. 2. Four oversampling strategies of Polynomial Fitting SMOTE. (a) Star topology. (b)Polynomial curve topology. (c) Bus topology. (d) Mesh topology.

$$x = x_{min} + \delta(x_{kNN} - x_{min}) \tag{1}$$

where $\delta \in [0, 1]$ is a random number. The number of synthetic samples is determined by the proportion of the majority and minority samples. Then a certain number of nearest neighbors should be selected according to this proportion. Since SMOTE was published in 2002, many variants have been proposed [46].

III. Proposed Method

In this section, an oversampling technique and data partition are combined with a standard stacking classification framework. Four kinds of sample generation methods based on PFSMOTE are used in combination. Data partition allows data from different processes to be trained separately. It makes the training more in line with the production process and improves the diversity of stacking ensemble learning.

A. *Polynomial Fitting SMOTE*

Different from SMOTE, PFSMOTE refers to four fairly different oversampling strategies controlled by the topology parameter of the technique: Star, Polynomial curve, Bus, and Mesh topologies. All operations are performed on minority samples. New synthetic samples are generated according to the above four different topologies. The algorithm is visualized in Figure 2. Blue circles represent the majority samples, red circles represent the minority samples, and red stars represent the synthetic minority samples. The numbers of majority samples and minority samples are m and n respectively. $m - n$ synthetic minority samples are needed to achieve a balance between majority and minority classes.

1) Star topology: As shown in Figure 2(a), we average each process parameter of unqualified ingot samples and use x (red triangle) to represent the mean value. Then we fit n linear functions

$$\begin{cases} y_1 = x_{min}^1 + \delta \times (\bar{x} - x_{min}^1) \\ \qquad\cdots \\ y_n = x_{min}^n + \delta \times (\bar{x} - x_{min}^n) \end{cases} \tag{2}$$

for connecting each unqualified ingot sample $x^i_{min}, i = 1, 2, ..., n$ and the mean value.

$l = ((m-n))/n$ new synthetic unqualified ingot samples are generated on each line. It is realized by adjusting the size of δ according to l, where $\delta = (\frac{1}{l+1}, \frac{2}{l+1}, ..., \frac{l}{l+1})$.

2) Polynomial curve topology: As shown in Figure 2(b), we fit unqualified ingot samples using a polynomial curve y of p degree as follows:

$$y = \omega_0 x^p + \omega_1 x^{p-1} + \omega_2 x^{p-2} + \cdots + \omega_n \tag{3}$$

where $\{\omega_0, \omega_1, ..., \omega_n\}$ are coefficients. In practice, we fit each feature of unqualified ingot samples using (3) with $p = 3$. As a result, we have d polynomial curves, where d is the number of features. $l = m-n$ new points are generated on each curve. It is realized by applying l random numbers as inputs of (3). After d operations, l new synthetic unqualified ingot samples are generated.

3) Bus topology: As shown in Figure 2(c), we fit $n-1$ linear functions

$$\begin{cases} y_1 = x^1_{min} + \delta \times (x^2_{min} - x^1_{min}) \\ \quad\cdots \\ y_{n-1} = x^{n-1}_{min} + \delta \times (x^n_{min} - x^{n-1}_{min}) \end{cases} \tag{4}$$

for connecting two succeeding unqualified ingot samples. Therefore, $l = ((m-n))/((n-1))$ new synthetic unqualified ingot samples are generated on each line. It is realized by adjusting the size of δ according to l in the same way as Star topology.

4) Mesh topology: As shown in Figure 2(d), we fit $(n(n-1))/2$ linear functions for connecting one unqualified ingot sample to all others. Therefore, $l = (2(m-n))/(n(n-1))$ new synthetic unqualified ingot samples are generated on each line. It is realized in the same way as Star topology.

Four ways generate four different types of minority samples. These samples are randomly added to the training set.

B. Oversampling Partition Stacking

We propose a new stacking framework that is based on standard stacking. It also makes predictions using two levels, base-level and meta-level. Typically, there are multiple classifiers at base-level and one at meta-level. In order to make the classification framework more compatible with melting-casting process and utilize the characteristics of the heterogeneous ensemble, raw data is input stacking model in the upstream and downstream order of practical production, i.e. melting data, holding data, and casting data. At the base-level, the classifiers only train melting data first. The results that the base-classifiers predict are fused with holding data. In the same way, the new results are then fused with casting data. Once again, base-classifiers train and make predictions. After that, the last results are fused with all data for training at the meta-level. A meta-classifier is applied to generate the final prediction. Before each training, PFSMOTE is needed to balance the training data.

Specific algorithm pseudocode is shown in Algorithm 1 and specific details of the framework are depicted in Figure 3. We use $\{BC_1, BC_2, ..., BC_T\}$ and MC to represent a set of base-classifiers and a meta-classifier respectively, where T is the number of base-classifiers. Raw data is divided into a training set D_{tr} and a test set D_{te}. Same as standard stacking, in the learning process, the operations on training and test data are almost identical, with the only difference being that base-classifiers' predictions for the test set are averaged during cross-validation. The training set is split into k folds, $k-1$ folds of which are utilized for training, and the remaining one fold is used for validation. At the same time, the test set is also predicted. After k successive rounds, all samples of the training set are predicted, and the mean value of k test set's outputs is also calculated.

In Step 1 of Algorithm 1, we divide the training set and the test set into three parts. For the sake of brevity, we take the training set as an example. Partition D_{tr} according to different production processes, $D_{tr} = \left\{D^{pr}_{tr}\right\}, pr = 1, 2, 3,$ where pr denotes different processes, hence $D^1_{tr}, D^2_{tr}, D^3_{tr}$ denote melting data, holding data and casting data respectively. According to the framework of stacking, input data into the ensemble framework in turn. At base-level, input D^1_{tr} to base-classifiers $\{BC_1, BC_2, ..., BC_T\}$ firstly. The output is denoted by P^1_{tr}. It means that this part of the process has an influence on the quality of ingots after the melting is over.

We select the probability distribution as the output of each classifier, hence P^1_{tr} can be denoted as

$$P^{pr}_{tr} = \left[P^{pr}_{BC_{t,tr}}, \left(1 - P^{pr}_{BC_{t,tr}}\right)\right], t = 1, 2, ..., T \tag{5}$$

with $pr = 1$. Let $P^{pr}_{BC_{t,tr}}$ be the probability that the t-th classifier for each training sample in the training set of melting data belongs to a certain class. Since it is a binary classification, we use $1 - P^{pr}_{BC_{t,tr}}$ to represent the probability that belongs to the other class. We have M training samples. Therefore, the dimension of matrix P^1_{tr} is $M \times (2T)$.

Then concatenate P^1_{tr} to D^2_{tr} as a whole input for the next training. It comes to the refining stage at this point. Using base-classifiers to train $\left\{D^2_{tr}, P^1_{tr}\right\}$ as we do in the previous step. After that, a matrix P^2_{tr} is predicted and can likewise be denoted by (5) with $pr = 2$. By that analogy, we train $\left\{D^3_{tr}, P^2_{tr}\right\}$ and get P^2_{tr}. It means ingots quality is the cumulative result of each process. At meta-level, concatenate P^2_{tr} to D_{tr}, then train D_{tr}, P^3_{tr}. Once again, all production processes are fully considered to avoid missing useful information. This process is shown in Step 2 of Algorithm 1. The quality prediction of the test set transformed with the training set is the final result.

Based on the partition stacking framework, PFSMOTE is carried out to balance the samples before each training. As shown in Step 2 and Step 3 of Algorithm 1, PFSMOTE(ts) is oversampling for the current training set ts. After oversampling, balanced data is input into base-classifiers for training. At this time, the classifiers' ability to classify unqualified ingots is enhanced. In the same way, oversampling is also applied at the meta-level, so that the meta-classifier can better learn unqualified ingots. At base-level, k-fold cross-validation is performed on each part of the data, and each base-classifier is trained. Therefore, PFSMOTE is applied $3 \times k \times T$ times at base-level and only one time at the meta-level.

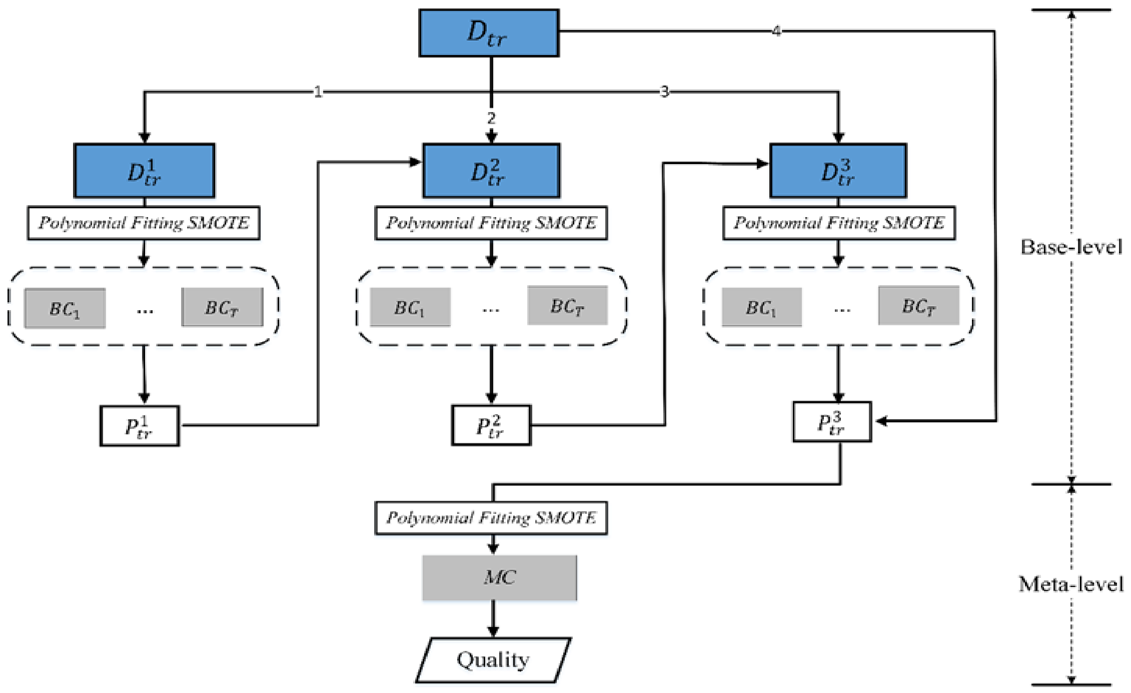

Fig. 3. Training details of partition stacking.

Algorithm 1 Oversampling partition stacking

```
Input: training set $D_{tr}$, test set $D_{te}$, base-classifiers $BC_1, BC_2, ..., BC_T$, meta-classifier $MC$
Output: quality prediction results
  Begin
  Step 1: Dataset partition
  According to the production process, divide the training set and test set into three parts.
  $D_{tr} = \{D_{tr}^{pr}\}_{pr=1}^{3}, D_{te} = \{D_{te}^{pr}\}_{pr=1}^{3}$.
  Step 2: Training base-classifiers
  Current training set $ts = \{D_{tr}^1\}$.
  for $pr \longleftarrow$ 1 to 3 do
    for $t \longleftarrow$ 1 to $T$ do
      Learn a base-classifier $BC_t$ using PFSMOTE($ts$).
    end for
    if $pr \neq 3$ then
      Construct new current training set and test set $\begin{cases} ts = \{D_{tr}^{pr+1}, P_{tr}^{pr}\} \\ test\ set = \{D_{te}^{pr+1}, P_{te}^{pr}\} \end{cases}$,
      where $\begin{cases} P_{tr}^{pr} = \{P_{BC_{t,tr}}^{pr}, 1 - P_{BC_{t,tr}}^{pr}\}, t = 1, 2, ..., T \\ P_{te}^{pr} = \{P_{BC_{t,te}}^{pr}, 1 - P_{BC_{t,te}}^{pr}\}, t = 1, 2, ..., T \end{cases}$
    else
      Construct new current training set and test set $\begin{cases} ts = \{D_{tr}, P_{tr}^{pr}\} \\ test\ set = \{D_{te}, P_{te}^{pr}\} \end{cases}$ for meta-classifier.
    end if
  end for
  Step 3: Learn meta-classifiers
  Learn meta-classifier $MC$ using PFSMOTE($ts$)
  return $MC(test\ set)$
  End
```

In terms of the framework we proposed, single classifier selection is based on two principles: low complexity and high diversity between different classifiers [23, 45]. Each classifier should be as unique as possible, that is, each classifier has a different classification boundary from each other [36]. It requires that each classifier has different misclassified samples

TABLE I Selected process parameters/features of raw data

Process Nodes	Process Parameters/Features
Melting	Additive Temperature, Master Alloy Temperature, Melting Temperature
Refining	Refining Times, Total Refining Time, Refining Temperature, Ar2 Flow, Cl2 Flow
Casting	Holder Actual Temperature, SNIF 1 Actual Temperature, CFF Preheat Time, CFF Temperature, Total Casting Time, Actual Number of Ingots, Yield
Quality Detection	Appearance Quality

so that they can complement each other when integrating the results. Such a set of classifiers is usually unstable to use since they can generate sufficiently different predictive results even for small perturbations in their training parameters. Generally, the higher the number of base-classifiers, the better the results we achieve. However, it is also prone to overfitting [22]. We chose three kinds of classifiers for base-level, namely kNN, DT, and LR. It can be said that these three kinds of classification algorithms are very general and simple. Their classification mechanism is completely different. At the meta-level, we use SVM as a meta-classifier. SVM is a powerful classifier with a wide range of applications [45].

IV. Experimental results

In this section, we evaluate the performance of the proposed method. We start with the data collection and briefly introduce data preprocessing. Experimental setup and several evaluation metrics are also explained. Then we show a series of experiments. In order to demonstrate the performance of the combination of standard stacking framework and oversampling, we first test on several public datasets. Then, the comparison experiments are conducted on real-world aluminum alloy ingot data. Data partition further improves the performance of the model compared to the previous experiment. Multiple comparative experiments verify the effectiveness of the proposed method.

A. Dataset and Preprocessing

We take AA-5182 aluminum alloy as the main object of our study. 204 samples are collected in an aluminum casting plant in China. Each sample represents the entire melting-casting process. With the help of process knowledge and relevant experts, some features that are not useful should be removed. The selected features are shown in Table I. Raw data is dirty and needs to be cleaned before modeling. We correct the outlier using the mean or mode instead, fill the respective feature mean values into the missing values, and delete duplicate values. So far, we have obtained the production process data of aluminum alloy in the melting-casting stage. There are 204 rows (samples) and 16 columns (features). In 204 samples of aluminum alloy ingots, only 19 samples have quality problems. Imbalanced datasets can be described by three principal characteristics: imbalance rate (IR), number of samples, and number of features. IR is the proportion of majority and minority samples. There are 185 qualified ingots in our dataset, thus IR= 185 : 19 = 9.74. Dataset description is shown in Table II.

TABLE II Dataset description

IR	Number of samples	Number of features
9.74	204	16

TABLE III Confusion matrix

	Positive Class	Negative Class
Positive Class	True Positive	False Negative
Negative Class	False Positive	True Negative

B. Experimental Setup

All classifier parameters use the default values in scikit-learn which is a machine learning toolkit written in Python [46]. In order to fully consider the effects of randomness and persuasiveness of the model, classification performance is evaluated by five-folds stratified cross-validation and repeated five times using different random generator seeds. Different from the general cross-validation method, stratified cross-validation divides data according to the proportion of different classes. It ensures that every fold is also imbalanced. Using four folds as the training set and the remaining one fold as a test set. We take the average value of different evaluation metrics as the final experimental evaluation.

C. Evaluation Metrics

The selected evaluation metrics must be able to reflect the overall predictive performance of different classes. In Table III, the confusion matrix demonstrates the results of incorrectly and correctly classified samples of each class in binary classification. Here TP, TN, FP, and FN separately represent the number of minority samples that are classified as minority class, the number of minority samples that are classified as majority class, the number of majority samples that are classified as minority class, and the number of majority samples that are classified as majority class.

Accuracy is one of the prevailing evaluation metrics and is defined as the correct prediction sample size divided by the total testing sample size, as is shown in (6).

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{6}$$

However, accuracy ignores the effect of imbalanced samples and only reflects the overall prediction accuracy of the dataset. As a result, it is likely to generate deceptive high accuracy in imbalanced data. To address this drawback, additional three comprehensive evaluation metrics apply to validate this study [24]. They are F-measure, Area Under Roc Curve (ROC-AUC), and G-Mean. These evaluation metrics can well reflect the performance of the classifiers and each of them has a different focus.

F-measure: It can be shown in (7) and viewed as a weighted average of model accuracy and recall rates. Here, precision and recall are calculated as (8) and (9) respectively.

$$F - measure(F1) = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{7}$$

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

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

G-Mean: Geometric mean is a comprehensive evaluation method constructed with sensitivity and specificity, which is presented as (10). Sensitivity is equal to recall, and specificity can be calculated in (11). The higher G-Mean shows the balance between classes is reasonable and has outstanding performance in the binary classification model.

$$G - Mean = \sqrt{sensitivity \times specificity} \tag{10}$$

$$specificity = \frac{TN}{TN + FP} \tag{11}$$

ROC-AUC: It is an extensively used evaluation metric obtained from the Receiver Operating Characteristic (ROC) curve representing the area under the ROC curve.

In general, the four metrics are between 0 and 1. The larger they are, the better their classification performance is. ROC-AUC and G-Mean focus more on the minority class samples.

D. Performance Comparison

We test the effectiveness of adding PFSMOTE to a standard stacking classification framework to deal with data imbalance problems on several public datasets. As shown in Table IV, 15 datasets are collected in the Knowledge Extraction from Evolutionary Learning (KEEL) repository [47]. The datasets are sorted by IR from small to large. In order to show the wide range of applications of the model as much as possible, IR values are well dispersed. The number of features of these datasets also ranges widely. Most datasets have about the same number of minority class samples as ours. In Table IV, YES and NO indicate the use or non-use of PFSMOTE. The four metrics mentioned above are used to evaluate the feasibility of our method.

Experimental results are getting better for 13 of the 15 datasets, except for ecoli-0_vs_1 and dermatology-6. By comparing the values of four evaluation metrics, we can see that there is no significant change in classification ability. For the rest of the datasets - newthyroid2, glass6, cleveland-0_vs_4, and glass4, all evaluation metrics are improved. For other datasets, ROC-AUC and G-Mean increase while Accuracy and F1 decrease. That means the model sacrifices the classification accuracy of majority class samples to gain the classification accuracy of minority class samples. It is worth mentioning that in glass-0-1-6_vs_2 and glass2, although the overall accuracy decreases a lot, it can be seen that the model does not identify minority class samples without oversampling. However, our method can predict correctly. Although the ability is limited, we still think it is effective.

Next, we evaluate our proposed method on aluminum alloy ingot data. The experimental results are shown in the Table V. We start with single classifiers for classification. Considering that the difference between the proposed method and the standard stacking comprises two techniques, one is data partition according to the process, and the other is oversampling using PFSMOTE. We also compare the methods that use one alone. They are partition-stacking and PFSMOTE-stacking. Among single classifiers, kNN has the highest classification accuracy, reaching 0.9402. It also has the highest F1 value. For minority samples, DT performs best on ROC-AUC and G-Mean values. However, the performance of the other three classifiers is not satisfactory. It can be seen that most classifiers fail when handling the imbalance problem. When using a standard stacking ensemble framework, the results show that the accuracy and F1 perform better than single classifiers, while the values of ROC-AUC and G-Mean are a little better than kNN, SVM, and LR, but not as good as DT. It also reflects the fact that ensemble learning can improve the overall classification performance from the perspective of accuracy and sacrifice the classification accuracy of minority classes at the same time. When only data partition is used, accuracy and F1 value reach the best of the whole comparison experiment. Accordingly, compared with standard stacking, the classification effect of minority decreases. Similarly, when oversampling is used alone, the classification effect for the minority class will be better, while the classification effect for the majority class will be slightly reduced. When using both techniques, ROC-AUC and G-Mean values reach the best of the whole comparison experiment. For quality prediction, these evaluation metrics need to be given priority, while in fact, accuracy and F1 values are still within the acceptable range.

From the perspective of the diversity of ensemble learning, our method uses different minority class samples for oversampling during each training to obtain a lot of synthetic samples. In addition, data partition leads to greater diversity. Therefore, the final classification effect is the result of the joint action of the two techniques. We believe that our method is more suitable for the quality prediction of aluminum alloy ingots.

Then we compare some advanced oversampling methods based on SMOTE algorithm and use them respectively in our classification framework. They are SMOTE-TomekLinks [48], Assembled SMOTE [49], ProWSyn [50], SMOTE-IPF [51], SMOBD [52], G-SMOTE [53], CCR [54]. These oversampling methods can be roughly divided into two categories, one of which focuses on selecting minority class samples before generating synthetic samples, and the other focuses on removing synthetic noise after generating synthetic samples. Although the respective corresponding classifiers are different in the papers that propose these methods, they all have the ability to solve imbalance problems in different fields. The experimental results are shown in the Table VI. PFSMOTE works best in the minority class. In our framework, ROC-AUC and G-Mean values of PFSMOTE are higher than those of other oversampling methods, and Accuracy and F1 values are not the highest, but they are perfectly acceptable.

The common behavior of using the Star, Polynomial curve, Bus, and Mesh topologies is that each of them generates samples along line segments between relatively far samples of the minority class, thus, the generated samples are more scattered in the minority class's manifold than using other

TABLE IV Comparison of the effectiveness of combined oversampling and stacking method on public datasets

Name	IR	Number of samplers	Number of features	PFSMOTE	Accuracy	F1	ROC-AUC	G-Mean
ecoli-0_vs_1	1.86	220	7	NO	**0.9873**	**0.9904**	**0.9846**	**0.9844**
				YES	0.9846	0.9884	0.9812	0.981
glass0	2.06	214	9	NO	**0.8354**	**0.8776**	0.8132	0.8052
				YES	0.8234	0.8573	**0.8372**	**0.8346**
vehicle1	2.9	846	18	NO	**0.8005**	**0.8703**	0.7037	0.6726
				YES	0.7993	0.8557	**0.7962**	**0.7955**
glass-0-1-2-3_vs_4-5-6	3.2	214	9	NO	**0.9467**	**0.9644**	0.9417	0.9396
				YES	0.9456	0.9635	**0.9452**	**0.9436**
newthyroid2	5.14	215	5	NO	0.9888	0.9934	0.9818	0.9807
				YES	**0.9898**	**0.9939**	**0.9893**	**0.9888**
glass6	6.38	214	9	NO	0.9636	0.9794	0.8792	0.8649
				YES	**0.9721**	**0.984**	**0.9251**	**0.9174**
ecoli3	8.6	336	7	NO	**0.9303**	**0.9615**	0.7818	0.7438
				YES	0.8964	0.9393	**0.8765**	**0.8725**
vowel0	9.88	988	13	NO	**0.999**	**0.9994**	**0.9944**	**0.9944**
				YES	0.9988	0.9993	0.9993	0.9993
glass-0-1-6_vs_2	10.29	192	9	NO	**0.9116**	**0.9537**	0.5	0
				YES	0.7855	0.8717	**0.6762**	**0.6246**
ecoli-0-1_vs_5	11	240	6	NO	**0.9767**	**0.9875**	0.8691	0.8539
				YES	0.9653	0.9836	**0.9145**	**0.9068**
glass2	11.59	214	9	NO	**0.9208**	**0.9587**	0.5	0
				YES	0.779	0.8673	**0.6895**	**0.6565**
cleveland-0_vs_4	12.62	177	13	NO	0.9377	0.9669	0.6565	0.4895
				YES	**0.949**	**0.9723**	**0.8377**	**0.7804**
ecoli-0-1-4-6_vs_5	13	280	6	NO	**0.9779**	**0.9882**	0.8542	0.8369
				YES	0.9714	0.9845	**0.9108**	**0.9037**
glass4	15.47	214	9	NO	0.9684	0.9835	0.7495	0.6382
				YES	**0.9721**	**0.9849**	**0.9072**	**0.8787**
dermatology-6	16.9	358	34	NO	**1**	**1**	**1**	**1**
				YES	0.9683	0.9826	0.9145	0.9073

TABLE V Experimental results of proposed method and different combinations in classification framework

	Accuracy	F1	ROC-AUC	G-Mean
KNN	0.9402	0.968	0.704	0.5894
DT	0.9324	0.9624	0.8161	0.7892
LR	0.9118	0.9515	0.7	0.594
SVM	0.9373	0.9666	0.6739	0.5165
Standard stacking	0.9472	0.9716	0.7423	0.6552
PFSMOTE-stacking	0.9412	0.9675	0.8377	0.8054
Partition-stacking	**0.95**	**0.9734**	0.7406	0.5966
Proposed method	0.9362	0.9644	**0.8467**	**0.8274**

TABLE VI Experimental results between different oversampling methods

	Accuracy	F1	ROC-AUC	G-Mean
SMOTE_TomekLinks	0.9157	0.9531	0.7774	0.7319
Assembled_SMOTE	0.9088	0.9485	0.7887	0.7523
ProWSyn	0.9058	0.9469	0.7943	0.768
SMOTE-IPF	0.9166	0.9538	0.7612	0.6926
SMOBD	0.9099	0.9498	0.7591	0.7097
CCR	0.9354	0.9643	0.8044	0.7769
G_SMOTE	**0.9372**	**0.9654**	0.8055	0.7636
PFSMOTE	0.9362	0.9644	**0.8467**	**0.8274**

SMOTE variants. It has the advantage of making minority samples distributed at different locations throughout space more like a whole, avoiding treating samples far away from the center of synthetic samples as noise.

Finally, the proposed method is compared with several popular classification frameworks combining resampling and ensemble learning. In ensemble learning, two frameworks are used - bagging and boosting. Random Forest (RF) can be thought of as a special kind of bagging. The methods we compare can be divided into two categories according to different resampling rules. BalancedBagging [22] allows to undersample each subset of data before training each classifier of bagging. Here, the weak classifiers are DTs. BalancedRF [55] is similar to BalancedBagging, but replaces bagging with RF. RUSBoost [56] randomly undersample the dataset before performing a boosting iteration. A specific method that uses AdaBoost as weak classifiers in the bagging framework is called EasyEnsemble [57]. It allows to bag AdaBoost learners which are trained on balanced bootstrap samples. All of the above algorithms are based on undersampling method combined with different ensemble learning models. SMOTEBoost [58] and SMOTEBagging [59] use the oversampling method when training subsets. To be more comparative, we change the SMOTE of the original algorithm to the PFSMOTE used by our proposed method. The experimental results are shown in Table VII.

The experimental results show that SMOTEBagging has the best performance on Accuracy and F1 values and the proposed method has the best performance on ROC-AUC and G-Mean values. It can be seen that the combination of oversampling and ensemble learning can achieve better results on different evaluation metrics. It also explains that when the sample size of the minority class is not large enough, the oversampling method can achieve better performance. The proposed method performs best in the minority class we are most interested in, and the overall accuracy does not drop much.

TABLE VII Experimental results of proposed methods and other methods of combining resampling and ensemble learning

	Accuracy	F1	ROC-AUC	G-Mean
BalancedBagging	0.807	0.8827	0.7848	0.7745
BalancedRF	0.7409	0.8347	0.7614	0.7461
RUSBoost	0.794	0.8715	0.8313	0.8236
EasyEnsemble	0.7844	0.8659	0.8019	0.7916
SMOTEBoost	0.8519	0.9137	0.7567	0.7174
SMOTEBagging	**0.9393**	**0.9664**	0.826	0.7824
Proposed method	0.9362	0.9644	**0.8467**	**0.8274**

V. Conclusion

This paper presents a method for quality prediction of aluminum alloy ingots. We cross-use data mining and machine learning techniques to build a relationship model of process parameters and quality. Based on the stacking classification framework of ensemble learning, the data partition is done according to the production process when training classifiers. In order to solve the imbalance problem, PFSMOTE is used to adjust the proportion of different classes during each training. Comparative experiments show that the novel learning model is the most suitable for quality prediction and can avoid the losses caused by the detection after the formation of ingots.

References

[1] R. Branco, F. Berto, and A. Kotousov, "Special issue on "mechanical behaviour of aluminium alloys"," *Applied Sciences*, vol. 8, no. 10, p. 1854, 2018.

[2] D. Ashkenazi, "How aluminum changed the world: A metallurgical revolution through technological and cultural perspectives," *Technological Forecasting and Social Change*, vol. 143, pp. 101–113, 2019.

[3] G. Galevsky, V. Rudneva, and V. Aleksandrov, "Current state of the world and domestic aluminium production and consumption," in *IOP Conference Series: Materials Science and Engineering*, vol. 411, no. 1. IOP Publishing, 2018, p. 012017.

[4] B. Lela, I. Duplančić, and J. Prgin, "Possibility of grain size prediction in aa5754 aluminium ingots using neural networks," *International Journal of Cast Metals Research*, vol. 21, no. 5, pp. 357–363, 2008.

[5] K. D. Schnelle and R. S. Mah, "Product quality management using a real-time expert system," *ISIJ International*, vol. 34, no. 10, pp. 815–821, 1994.

[6] M.-J. Kim, J. P. Yun, J.-B.-R. Yang, S.-J. Choi, and D. Kim, "Prediction of the temperature of liquid aluminum and the dissolved hydrogen content in liquid aluminum with a machine learning approach," *Metals*, vol. 10, no. 3, p. 330, 2020.

[7] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[8] N. A. Azhar, M. S. M. Pozi, A. M. Din, and A. Jatowt, "An investigation of smote based methods for imbalanced datasets with data complexity analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 7, pp. 6651–6672, 2022.

[9] S. Gazzah and N. E. B. Amara, "New oversampling approaches based on polynomial fitting for imbalanced data sets," in *2008 The Eighth IAPR International Workshop on Document Analysis Systems*. IEEE, 2008, pp. 677–684.

[10] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

[11] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[12] J. Wang, A. Yousefzadi Nobakht, J. D. Blanks, D. Shin, S. Lee, A. Shyam, H. Rezayat, and S. Shin, "Machine learning for thermal transport analysis of aluminum alloys with precipitate morphology," *Advanced Theory and Simulations*, vol. 2, no. 4, p. 1800196, 2019.

[13] T. L. Galvao, G. Novell-Leruth, A. Kuznetsova, J. Tedim, and J. R. Gomes, "Elucidating structure–property relationships in aluminum alloy corrosion inhibitors by machine learning," *The Journal of Physical Chemistry C*, vol. 124, no. 10, pp. 5624–5635, 2020.

[14] X.-W. Yang, J.-C. Zhu, Z.-S. Nong, H. Dong, Z.-H. Lai, L. Ying, and F.-W. Liu, "Prediction of mechanical properties of a357 alloy using artificial neural network," *Transactions of Nonferrous Metals Society of China*, vol. 23, no. 3, pp. 788–795, 2013.

[15] I. Ghosh, S. K. Das, and N. Chakraborty, "An artificial neural network model to characterize porosity defects during solidification of a356 aluminum alloy," *Neural Computing and Applications*, vol. 25, pp. 653–662, 2014.

[16] M. Ulas, O. Aydur, T. Gurgenc, and C. Ozel, "Surface roughness prediction of machined aluminum alloy with wire electrical discharge machining by different machine learning algorithms," *Journal of Materials Research and Technology*, vol. 9, no. 6, pp. 12 512–12 524, 2020.

[17] I. Kruglov, O. Sergeev, A. Yanilkin, and A. R. Oganov, "Energy-free machine learning force field for aluminum," *Scientific Reports*, vol. 7, no. 1, p. 8512, 2017.

[18] D. J. Pasadas, H. G. Ramos, B. Feng, P. Baskaran, and A. L. Ribeiro, "Defect classification with svm and wideband excitation in multilayer aluminum plates," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 1, pp. 241–248, 2019.

[19] H. Liao, B. Zhao, X. Suo, and Q. Wang, "Prediction models for macro shrinkage of aluminum alloys based on machine learning algorithms," *Materials Today Communications*, vol. 21, p. 100715, 2019.

[20] W. Du, H. Shen, J. Fu, G. Zhang, and Q. He, "Approaches for improvement of the x-ray image defect detection of automobile casting aluminum parts based on deep learning," *Ndt & E International*, vol. 107, p. 102144, 2019.

[21] Z. Li, X.-Y. Jing, X. Zhu, H. Zhang, B. Xu, and S. Ying, "Heterogeneous defect prediction with two-stage ensemble learning," *Automated Software Engineering*, vol. 26, pp. 599–651, 2019.

[22] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 463–484, 2011.

[23] A. Mohammed and R. Kora, "A comprehensive review on ensemble deep learning: Opportunities and challenges," *Journal of King Saud University-Computer and Information Sciences*, vol. 35, no. 2, pp. 757–774, 2023.

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

[25] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, pp. 123–140, 1996.

[26] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *European Conference on Computational Learning Theory*. Springer, 1995, pp. 23–37.

[27] M. P. Sesmero, A. I. Ledezma, and A. Sanchis, "Generating ensembles of heterogeneous classifiers using stacked generalization," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 5, no. 1, pp. 21–34, 2015.

[28] W. Jiang, Z. Chen, Y. Xiang, D. Shao, L. Ma, and J. Zhang, "Ssem: A novel self-adaptive stacking ensemble model for classification," *IEEE Access*, vol. 7, pp. 120 337–120 349, 2019.

[29] S. Džeroski and B. Ženko, "Is combining classifiers with stacking better than selecting the best one?" *Machine Learning*, vol. 54, pp. 255–273, 2004.

[30] K. M. Ting and I. H. Witten, "Issues in stacked generalization," *Journal of Artificial Intelligence Research*, vol. 10, pp. 271–289, 1999.

[31] M. Koopialipoor, P. G. Asteris, A. S. Mohammed, D. E. Alexakis, A. Mamou, and D. J. Armaghani, "Introducing stacking machine learning approaches for the prediction of rock deformation," *Transportation Geotechnics*, vol. 34, p. 100756, 2022.

[32] Y. Chen, M.-L. Wong, and H. Li, "Applying ant colony optimization to configuring stacking ensembles for data mining," *Expert Systems with Applications*, vol. 41, no. 6, pp. 2688–2702, 2014.

[33] I. D. Mienye and Y. Sun, "A survey of ensemble learning: Concepts, algorithms, applications, and prospects," *IEEE Access*, vol. 10, pp. 99 129–99 149, 2022.

[34] B. M. Haddad, S. Yang, L. J. Karam, J. Ye, N. S. Patel, and M. W. Braun, "Multifeature, sparse-based approach for defects detection and classification in semiconductor units," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 1, pp. 145–159, 2016.

[35] Z. Hu, H. Qiu, Z. Su, M. Shen, and Z. Chen, "A stacking ensemble model to predict daily number of hospital admissions for cardiovascular diseases," *IEEE Access*, vol. 8, pp. 138 719–138 729, 2020.

[36] M. Jiang, J. Liu, L. Zhang, and C. Liu, "An improved stacking framework for stock index prediction by leveraging tree-based ensemble models and deep learning algorithms," *Physica A: Statistical Mechanics and its Applications*, vol. 541, p. 122272, 2020.

[37] J. Moon, S. Jung, J. Rew, S. Rho, and E. Hwang, "Combination of short-term load forecasting models based on a stacking ensemble approach," *Energy and Buildings*, vol. 216, p. 109921, 2020.

[38] W. Sun and B. Trevor, "A stacking ensemble learning framework for annual river ice breakup dates," *Journal of Hydrology*, vol. 561, pp. 636–650, 2018.

[39] H. Pham-The, G. Casañola-Martin, T. Garrigues, M. Bermejo, I. González-Álvarez, N. Nguyen-Hai, M. Á. Cabrera-Pérez, and H. Le-Thi-Thu, "Exploring different strategies for imbalanced adme data problem: case study on caco-2 permeability modeling," *Molecular Diversity*, vol. 20, pp. 93–109, 2016.

[40] G. M. Weiss and F. Provost, "Learning when training data are costly: The effect of class distribution on tree induction," *Journal of Artificial Intelligence Research*, vol. 19, pp. 315–354, 2003.

[41] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 2016.

[42] K. Napierala and J. Stefanowski, "Types of minority class examples and their influence on learning classifiers from imbalanced data," *Journal of Intelligent Information Systems*, vol. 46, pp. 563–597, 2016.

[43] O. Loyola-González, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, and M. García-Borroto, "Study of the impact of resampling methods for contrast pattern based classifiers in imbalanced databases," *Neurocomputing*, vol. 175, pp. 935–947, 2016.

[44] G. Kovács, "An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets," *Applied Soft Computing*, vol. 83, p. 105662, 2019.

[45] P. Priore, B. Ponte, J. Puente, and A. Gómez, "Learning-based scheduling of flexible manufacturing systems using ensemble methods," *Computers & Industrial Engineering*, vol. 126, pp. 282–291, 2018.

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[47] J. Alcalá-Fdez, L. Sanchez, S. Garcia, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas *et al.*, "Keel: a software tool to assess evolutionary algorithms for data mining problems," *Soft Computing*, vol. 13, pp. 307–318, 2009.

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

[49] B. Zhou, C. Yang, H. Guo, and J. Hu, "A quasi-linear svm combined with assembled smote for imbalanced data classification," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2013, pp. 1–7.

[50] S. Barua, M. M. Islam, and K. Murase, "Prowsyn: Proximity weighted synthetic oversampling technique for imbalanced data set learning," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2013, pp. 317–328.

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

[52] Q. Cao and S. Wang, "Applying over-sampling technique based on data density and cost-sensitive svm to imbalanced learning," in *2011 International Conference on Information Management, Innovation Management and Industrial Engineering*, vol. 2. IEEE, 2011, pp. 543–548.

[53] T. Sandhan and J. Y. Choi, "Handling imbalanced datasets by partially guided hybrid sampling for pattern recognition," in *2014 22nd International Conference on Pattern Recognition*. IEEE, 2014, pp. 1449–1453.

[54] M. Koziarski and M. Woźniak, "Ccr: A combined cleaning and resampling algorithm for imbalanced data classification," *International Journal of Applied Mathematics and Computer Science*, vol. 27, no. 4, 2017.

[55] T. M. Khoshgoftaar, M. Golawala, and J. Van Hulse, "An empirical study of learning from imbalanced data using random forest," in *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, vol. 2. IEEE, 2007, pp. 310–317.

[56] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "Rusboost: A hybrid approach to alleviating class imbalance," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and*

Humans, vol. 40, no. 1, pp. 185–197, 2009.

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

[58] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: Improving prediction of the minority class in boosting," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2003, pp. 107–119.

[59] S. Wang and X. Yao, "Diversity analysis on imbalanced data sets by using ensemble models," in *2009 IEEE Symposium on Computational Intelligence and Data Mining*. IEEE, 2009, pp. 324–331.

Shen Yan received his B.S. and M.S. degrees in 2014 and 2017, from the School of Science and the School of Information Science and Engineering, Shenyang University of Technology, Shenyang, China, respectively, Ph. D. degree in Systems Engineering from Northeastern University, Shenyang, China in 2024. He is currently a lecturer of the School of Science, Shenyang University of Technology, Shenyang, China. His research focuses on data mining, machine learning, and deep learning.

Haifeng Guo received the B.S. degree from the College of Mathematics and Computer Science, Chongqing University of International Business and Economics, Chongqing, China, in 2024. She is currently pursuing the M.S. degree with the College of Science, Shenyang University of Technology, Shenyang, China. Her research focuses on reinforcement learning, control systems, and deep learning.

Shixin Liu received his B.S. degree in Mechanical Engineering from Southwest Jiaotong University, Sichuan, China in 1990, M.S. degree and Ph. D. degree in Systems Engineering from Northeastern University, Shenyang, China in 1993, and 2000. He is currently a Professor of the College of Information Science and Engineering, Northeastern University, Shenyang, China. His research interests are in intelligent optimization algorithm, project management, and the theory and method of planning and scheduling. He has over 100 publications including 1 book.