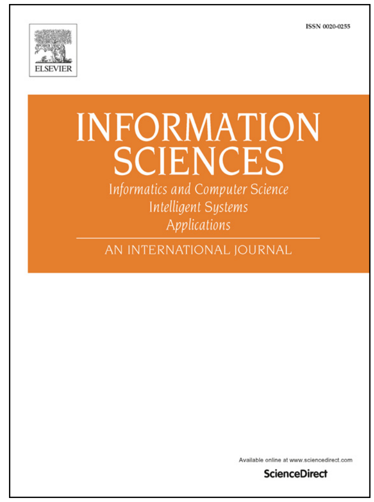
Journal Pre-proof



Multi-Class Imbalance Problem: A Multi-Objective Solution

Yi-Xiao He, Dan-Xuan Liu, Shen-Huan Lyu, Chao Qian and Zhi-Hua Zhou

PII: S0020-0255(24)01070-3
DOI: https://doi.org/10.1016/j.ins.2024.121156
Reference: INS 121156

To appear in: *Information Sciences*

Received date: 6 January 2024
Revised date: 1 July 2024
Accepted date: 3 July 2024

Please cite this article as: Y.-X. He, D.-X. Liu, S.-H. Lyu et al., Multi-Class Imbalance Problem: A Multi-Objective Solution, *Information Sciences*, 121156, doi: https://doi.org/10.1016/j.ins.2024.121156.

Highlights

- We explicitly consider the between-class trade-off issue in the multi-class imbalance problem.
- To find the many optimal trade-off solutions, we design an efficient multi-objective optimization method incorporating selective ensemble and varied downsampling rates.
- We further propose a margin-based objective modeling to tackle the many-class case, and analyze its optimization ability.
- Our methods successfully obtain diverse and highly competitive solutions within an acceptable running time.

Multi-Class Imbalance Problem: A Multi-Objective Solution

Yi-Xiao He[a,b,1], Dan-Xuan Liu[a,b,1], Shen-Huan Lyu[c,d,a,2], Chao Qian[a,b,1,*], Zhi-Hua Zhou[a,b,1]

[a]*National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, 210023, China*
[b]*School of Artificial Intelligence, Nanjing University, Nanjing, 210023, China*
[c]*Key Laboratory of Water Big Data Technology of Ministry of Water Resources, Hohai University, Nanjing, 211100, China*
[d]*College of Computer Science and Software Engineering, Hohai University, Nanjing, 211100, China*

Abstract

Multi-class imbalance problems are frequently encountered in real-world applications of machine learning. They have fundamentally complex trade-offs between classes. Existing literature tends to use a predetermined rebalancing strategy and mostly focuses on overall performance measures. However, in many real-world problems, the true level of imbalance and the relative importance between classes are unknown, making it difficult to predetermine the rebalancing strategy and the evaluation criterion. In this paper, we explicitly consider the between-class trade-off issue in the multi-class imbalance problem. We consider all the classes to be important and find a set of optimal trade-offs for the decision-maker to choose from. To reduce the computational cost of this process and make it a practical method, we seek the help of selective ensemble and multiple undersampling rates, and propose the Multi-class Multi-objective Selective Ensemble (MMSE) framework. We further equip the objective modeling with margins to reduce the number of objectives when the task has many classes. Experimental results show that our proposed methods successfully obtain diverse and highly competitive solutions within an acceptable running time.

*Corresponding author.
[1]{heyx,liudx,qianc,zhouzh}@lamda.nju.edu.cn
[2]lvsh@hhu.edu.cn

Keywords: class-imbalanced learning, ensemble method, multi-objective optimization

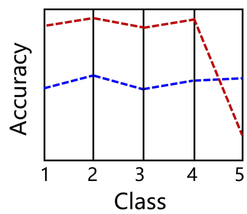
1. Introduction

Class imbalance is a problem frequently encountered in classification tasks [18]. The data collected can be naturally imbalanced, such as the number of patients with different diseases [22]. Abundant imbalanced learning methods have been developed to enhance the relative impact of the minority class in binary classification problems and achieved good results [17, 4, 5, 25]. However, multi-class imbalance problems are fundamentally more complex [35, 24].

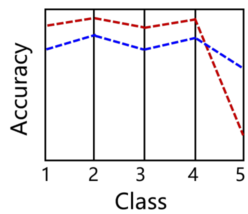
Firstly, in binary classification, even random guesses can achieve an accuracy of 50%, making the problem relatively easy. In contrast, in multi-class cases, vulnerable classes can perform extremely poorly. Secondly, in binary classification, the trade-offs are only between one small class and one big class, while in multi-class imbalance problems, the trade-offs are not only between small and big classes, but also between different small classes and between different big classes. Therefore, designing a rebalancing strategy for multi-class imbalance problems is more challenging. Finally, when it comes to model evaluation, it is hard to describe a multi-class classifier in one overall performance score.

In addition to multi-class classification being more complex than binary classification, another challenge we often face in real-world applications is that the ground-truth level of imbalance and the ground-truth relative importance of the classes are often unknown [48]. Note that under the traditional close-environment assumptions, we always know the targeted performance measure beforehand [49]. Nevertheless, in an open environment, it is not always possible to determine the relative importance of each class *a priori*. If we can provide the decision-maker with all the possible best trade-off performances of the model, it will greatly help them make decisions in an open environment.

Taking disease classification as an example, misdiagnosis of certain rare diseases (classes with a small number of samples) may cause serious problems, but meanwhile it is impossible to quantify the importance of each class. Figure 1 gives two examples of different trade-offs. In each example we assume that there are only two optimal trade-offs, in fact, there may be



(a) Assuming there are only two optimal trade-offs as shown in the figure, the decision-maker chooses the classifier shown in red.



(b) Assuming there are only two optimal trade-offs as shown in the figure, the decision-maker chooses the classifier shown in blue.

Figure 1: Different trade-offs of per-class accuracy. Different optimal trade-offs result in different choices by the decision-maker.

many more trade-offs in real applications. If the only two optimal trade-offs are as shown in Figure 1(a), the decision maker may choose the classifier shown in red because it *can distinguish at least the first four classes.* If the fifth class is indeed important, a separate inspection can be designed. If the only two optimal trade-offs are as shown in Figure 1(b), the decision-maker may choose the classifier shown in blue because it *achieves satisfactory performance on all classes.* The fundamental factor that affects the decision-maker's choice here is that the improvement of the fifth class has different effects on other classes. Only by presenting different optimal trade-offs to the decision-maker can she make better choices.

Therefore, when we cannot determine the importance of each class in advance, we hope to obtain diverse optimal trade-offs among classes for the decision-maker to choose from. To achieve this goal, we propose to model the multi-class imbalance problem as a multi-objective problem

$$\text{maximize}\,(M_1, M_2 \ldots, M_l)\ , \tag{1}$$

where l denotes the number of classes, M_i is the model's performance on the i-th class. Given that solutions excelling in different objectives are incomparable, multi-objective problems usually have multiple optimal solutions [50, 29, 44]. These optimal trade-off solutions are referred to as *Pareto-optimal solutions* (or the *Pareto front* in the objective space). It is assumed that revealing the Pareto front will better equip the decision-maker to make the final choice among these trade-offs.

In the process of searching for multiple optimal solutions on the Pareto front, we need to generate a large number of solutions, each emphasizing different classes. This process can lead to significant model training overhead. Therefore, reducing this overhead is essential for transforming our goal into a practical learning algorithm. To address this issue, we propose the Multi-class Multi-objective Selective Ensemble (MMSE) framework. It encompasses three fundamental points. 1) We incorporate **selective ensemble** into the multi-objective modeling. In this way, we don't have to repeatedly train the entire model, but instead obtain different models through different combinations of base learners. 2) We use **undersampled** datasets to train base learners, which improves training efficiency. Meanwhile, the model obtained by ensembling multiple base learners can cover more training samples, which avoids the problem of information loss. 3) We undersample the dataset with **different undersampling ratios**. Different undersampling ratios for each class represent different rebalancing strategies. By combining base learners that have heterogeneous emphases over classes, we can obtain a variety of ensemble models with more diverse choices in performance across different classes.

With straightforward objective modeling where the performance of each class is modeled as an objective, we propose $\mathrm{MMSE}_{\mathrm{class}}$. However, scalability is another issue that must be taken into consideration. When the number of classes increases, the optimization problem becomes difficult because most of the generated solutions are incomparable. Considering this, we further propose a margin-based version called $\mathrm{MMSE}_{\mathrm{margin}}$. It optimizes common performance measures by optimizing label-wise and instance-wise margins. It not only reduces the number of objectives to 3 but also proves to be able to optimize common performance measures.

Our contributions are summarized as follows:

- We explore the multi-class imbalance problems from a new perspective, specifically when it is difficult to determine trade-offs between classes *a priori*.
- We model the problem as a multi-objective problem, where the performance of each class is optimized as a separate objective. But more importantly, in order to improve efficiency and make the method practical, we incorporate undersampling and selective ensemble, and develop the MMSE framework.
- Considering the scalability issue when the number of classes increases, we further propose a variant of objective modeling that equips with

margins, and analyze its optimization ability.

- We show in the experiments that both $\text{MMSE}_{\text{class}}$ and $\text{MMSE}_{\text{margin}}$ not only achieve better performance on common performance measures, but also provide a variety of trade-offs between classes, and within an acceptable running time.

The rest of this paper is organized as follows. We start by introducing the related work in Section 2. In Section 3, we first demonstrate the problem settings, then introduce the proposed MMSE framework in detail. Theoretical analysis is provided in Section 4. In Section 5, experimental results are reported. Finally, we conclude our work in Section 6.

2. Related work

The most fundamental idea for solving class-imbalanced learning problems is rebalancing. The methods can be roughly categorized into the following three types. a) Sampling methods. These methods include random sampling, synthetic sampling [4, 16], and evolutionary-based sampling methods [11, 32]. b) Re-weighting methods. They are closely related to cost-sensitive learning where instances in small classes have higher misclassification costs [26]. c) Hybrid methods. They combine multiple techniques, such as integrating sampling methods in each boosting round [5, 33], ensembling multiple base learners trained on different balanced training sets [6, 25, 10].

Ensemble methods naturally have applications in solving class imbalance problems, because they can combine the strengths of multiple learners to achieve better performance [42, 41, 43, 38]. A highly representative approach is EasyEnsemble [25]. It combines undersampling with ensemble to achieve effective rebalancing while avoiding information loss. In addition, selective ensemble methods aim to use some base learners to achieve better results than a complete ensemble [19], and can also be applied to handle class imbalance problems [9].

It is worth noting that, many of the imbalanced-learning methods were originally proposed for binary problems, and the binary imbalanced classification has been studied more thoroughly [35, 9]. Although many learning methods are applicable to multi-class imbalance problems, they are generally direct extensions of the binary rebalancing strategies, without considering the complex trade-offs among multiple classes [37, 19]. Usually, a learner

Table 1: Definition of popular multi-class performance measures

Measure	Formulation	Note
Average Accuracy	$\text{Avg. Acc}(h)=\frac{1}{l}\sum_{y=1}^{l}\frac{1}{\lvert D_y\rvert}\sum_{i\in D_y}[\![h(\boldsymbol{x}_i)=y]\!]$	The average of per-class accuracy.
G-mean	$\text{G-mean}(h)=\left\{\prod_{y=1}^{l}\left(\frac{1}{\lvert D_y\rvert}\sum_{i\in D_y}[\![h(\boldsymbol{x}_i)=y]\!]\right)\right\}^{\frac{1}{l}}$	The geometric mean of per-class accuracy.
macro-F1	$\text{macro-F1}(h)=\frac{1}{l}\sum_{y=1}^{l}\frac{2\sum_{i\in D_y}[\![h(\boldsymbol{x}_i)=y]\!]}{\lvert D_y\rvert+\sum_{i\in D_y}[\![h(\boldsymbol{x}_i)=y]\!]}$	F-measure averaging on each class.
micro-F1	$\text{micro-F1}(h)=\frac{2\sum_{j=1}^{l}\sum_{i\in D_y}[\![h(\boldsymbol{x}_i)=y]\!]}{\lvert D\rvert+\sum_{j=1}^{l}\sum_{i\in D_y}[\![h(\boldsymbol{x}_i)=y]\!]}$	F-measure averaging on the prediction matrix.
macro-AUC	$\text{macro-AUC}\ (f)=\frac{1}{l}\sum_{y=1}^{l}\frac{\mathcal{S}_{\text{macro}}^{y}}{\lvert D_y\rvert\lvert D\backslash D_y\rvert}$ $\mathcal{S}_{\text{macro}}^{y}=\left\{(a,b)\in D_y\times\{D\backslash D_y\}\mid f^{(y)}(\boldsymbol{x}_a)\geq f^{(y)}(\boldsymbol{x}_b)\right\}$	AUC averaging on each class. $\mathcal{S}_{\text{macro}}$ is the set of correctly ordered instance pairs considering whether the instance belongs to the corresponding class.
MAUC [14]	$\text{MAUC}\ (f)=\frac{2}{l(l-1)}\sum_{i<j}\hat{A}(i,j)$ $\hat{A}(i,j)=[\hat{A}(i\mid j)+\hat{A}(j\mid i)]/2$ $\hat{A}(i\mid j)=\frac{1}{\lvert D_i\rvert\lvert D_j\rvert}\left\{(a,b)\in D_i\times D_j\mid f^{(j)}(\boldsymbol{x}_a)\geq f^{(j)}(\boldsymbol{x}_b)\right\}$	AUC averaging on each pair of classes. $\hat{A}(i\mid j)$ is the correctly ordered instance pairs of the i-th and j-th class based on the predicted probabilities on the i-th class.

is trained based on a pre-determined rebalancing strategy, and then the results on a series of evaluation criteria, such as F1, G-mean, and MAUC, are reported [43]. Table 1 summarizes six performance measures commonly used in multi-class imbalance studies. However, few studies have been conducted when the evaluation criteria and the relative importance of classes are unknown beforehand.

In this paper, we consider the performance of different classes as multiple objectives. Recently, many methods have been proposed to optimize multiple objectives simultaneously while training models [23, 40, 39, 46], such as simultaneously optimizing accuracy and regularization, or considering objectives related to specific tasks such as feature selection. However, they did not consider the trade-offs among classes. Instead, we directly model the performance of each class as an objective, and our goal is to provide different trade-offs between classes for the decision-maker to make choices. This is a clear difference that makes this paper a different study from existing literature. Although the idea of modeling each class as an objective is simple, making its optimization practical requires exquisite design, which is the focus of our work.

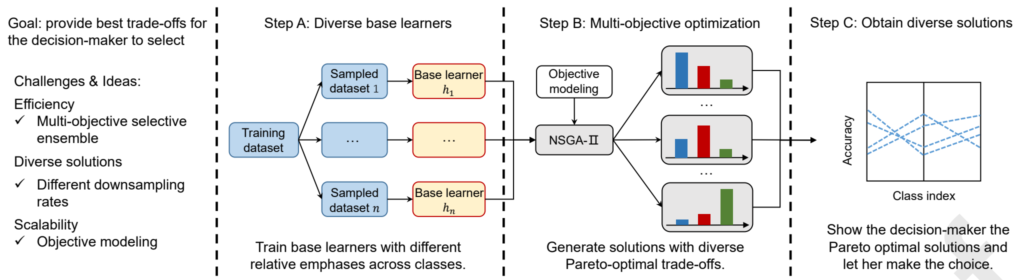


Figure 2: An illustration of our proposed MMSE framework.

3. The proposed approach

3.1. Problem description

Given the multi-class predictor $f : \mathbb{R}^d \to \mathbb{R}^l$, where $f^{(j)}(\boldsymbol{x})$ denotes the predicted probability of instance $\boldsymbol{x}$ on the j-th class. Let $h(\boldsymbol{x}) = \arg\max_j f^{(j)}(\boldsymbol{x})$ denote the predicted class. Let D denote a dataset sampled i.i.d. from distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} = \mathbb{R}^d$ is the feature space and $\mathcal{Y} \in \{1, 2, \ldots, l\}$ is the label space. Let D_y denote the set of sample indices with label y. $\mathbb{1}_{[\cdot]}$ is the indicator function, which returns 1 if $\cdot$ is true and 0 otherwise.

In this paper, we consider the problem where the decision-maker's evaluation criterion is not revealed until she sees the best possible trade-off solutions. We consider the following two scenarios of the evaluation process.

Scenario I: After the Pareto front is revealed, the decision-maker decides on a certain overall performance measure. The solution that has the best validation performance on this measure is chosen and the corresponding test performance is reported. We consider the measures in Table 1 to be the possible preferences of the decision-maker.

Scenario II: This scenario covers a broader context, in which the decision-maker may choose any solution on the Pareto front presented to her. Unlike scenario I, the decision-making process here may be a high-level consideration of the trade-offs between classes, which is hard to represent explicitly.

In this paper, we propose a multi-objective selective ensemble method that can deal with the two scenarios simultaneously. Our method not only achieves good performance in common overall performance measures, but also generates diverse trade-off solutions between classes.

3.2. The multi-class multi-objective selective ensemble framework

We present our framework MMSE, as illustrated in Figure 2. It incorporates selective ensemble in the multi-objective optimization to enhance training and storage efficiency, and employs undersampling with different ratios to help generate diverse solutions.

Multi-objective optimization. To explicitly consider different trade-offs between classes, we use the validation accuracy of each class as an objective, and the multi-objective problem is formulated as

$$\boldsymbol{g}(h, V) = \left(\frac{1}{|V_1|} \sum_{i \in V_1} \mathbb{1}_{[h(\boldsymbol{x}_i)=1]}, \ldots, \frac{1}{|V_l|} \sum_{i \in V_l} \mathbb{1}_{[h(\boldsymbol{x}_i)=l]} \right) , \tag{2}$$

where V denotes the validation set, and V_i denotes the subset of samples belonging to the i-th class. Usually, the solution to this multi-objective optimization problem contains many optimal classifiers h, which have their different advantages in different classes.

Selective ensemble. Let $F_{\boldsymbol{s}}$ denote a selective ensemble with selector vector $\boldsymbol{s} \in \{0, 1\}^n$, where $s_t = 1$ means that the base learner f_t is incorporated in the ensemble. If we consider soft voting to combine the base learners, the predicted probability of ensemble $F_{\boldsymbol{s}}$ on an instance $\boldsymbol{x}$ is

$$F_{\boldsymbol{s}}(\boldsymbol{x}) = \frac{1}{|\boldsymbol{s}|} \sum_{t=1}^{n} s_t f_t(\boldsymbol{x}) ,$$

where $|\boldsymbol{s}| = \sum_{t=1}^{n} s_t$ represents the ensemble size. And let

$$H_{\boldsymbol{s}}(\boldsymbol{x}) = \arg\max_j F_{\boldsymbol{s}}^{(j)}(\boldsymbol{x}) ,$$

denote the predicted class. In this way, the multi-objective optimization problem becomes a search on the selector vector, i.e.,

$$\boldsymbol{g}(\boldsymbol{s}, V) = \left(\frac{1}{|V_1|} \sum_{i \in V_1} \mathbb{1}_{[H_{\boldsymbol{s}}(\boldsymbol{x}_i)=1]}, \ldots, \frac{1}{|V_l|} \sum_{i \in V_l} \mathbb{1}_{[H_{\boldsymbol{s}}(\boldsymbol{x}_i)=l]}, -|\boldsymbol{s}| \right) . \tag{3}$$

Combining selective ensemble with multi-objective optimization leads to greatly reduced time and storage consumption. Without the design of incorporating selective ensemble, to find these solutions using a multi-objective evolutionary algorithm, we have to search many (usually thousands of) rebalancing settings. Since for each setting we have to train a classifier, in total, we need to train thousands of classifiers from scratch. In contrast, using the framework we proposed, we only need to search thousands of combinations.

Generating base learners. When generating the base learners, we construct multiple undersampled subsets from the training set Tr. Undersampling is an efficient way to obtain rebalanced datasets with low training overhead. Compared to it, oversampling has a higher training cost and may also cause overfitting. The only weakness of undersampling is the possibility of discarding useful samples. But this disadvantage can be compensated for by ensembling multiple undersampled datasets, which avoids information loss [25]. Based on this idea, we use each subset to train a separate classifier, and the final prediction is made by combining the predictions of all the classifiers. But there is a novel design in this step of our method, i.e., we undersample the dataset with *different undersampling ratios*. From EasyEnsemble, we know that when ensembling base learners trained on balanced subsets, the ensemble performance will vary depending on the number of base learners. Obviously, if the sampling ratios on different classes change for different data subsets, the performance of the obtained ensemble will also exhibit more diversity. As our goal is to obtain heterogeneous trade-offs among classes, combining base learners with heterogeneous emphases over classes will help.

3.3. Objective modeling for many-class cases

In the previous subsection, we use the Eq. (3) version of objective modeling, where the validation accuracy of each class is modeled as objective. Therefore, we name this method as $\text{MMSE}_{\text{class}}$. This type of objective modeling is flexible, and if the optimization problem is well solved, any optimal trade-off between classes can be obtained. However, when the number of classes is large, the multi-objective problem becomes difficult to optimize because most of the generated solutions are incomparable. In such cases, we propose a margin-based version of objective modeling, and we name the MMSE method equipped with margin-based objective modeling as $\text{MMSE}_{\text{margin}}$.

The concept of margin has been long used in evaluating the model's training performance [13], showing its effectiveness in both generalization ability and robustness. There have been some new research results recently, such as applying it to multi-label problems [36], or using its distribution to characterize classifier performance more precisely [27]. Inspired by the fact that optimizing label-wise and instance-wise margins can optimize various commonly used multi-label performance measures [36], we decided to optimize the multi-class version of label-wise and instance-wise margins to address our Scenario I. And we apply different methods to aggregate per-class margins so that our method can retain certain advantages in Scenario II. Here we introduce the multi-class version of label-wise and instance-wise margins.

The *label-wise margin* on instance $\boldsymbol{x}_i$ is defined to be

$$\gamma_i^{\text{label}}(f, \boldsymbol{x}_i) = \min_{y'} \left\{ f^{(y)}\left(\boldsymbol{x}_i\right) - f^{(y' \neq y)}\left(\boldsymbol{x}_i\right) \right\}, \tag{4}$$

where y is the ground-truth label of instance $\boldsymbol{x}_i$. We group the label-wise margin on the instances from the y-th class

$$\bar{\gamma}_y^{\text{label}}(f, V) = \frac{1}{|V_y|} \sum_{i \in V_y} \gamma_i^{\text{label}}\left(f, \boldsymbol{x}_i\right). \tag{5}$$

The *instance-wise margin* on label y is defined to be

$$\gamma_y^{\text{inst}}(f, V) = \min_{a,b} \left\{ f^{(y)}\left(\boldsymbol{x}_a\right) - f^{(y)}\left(\boldsymbol{x}_b\right) \mid a \in V_y, b \in V \backslash V_y \right\}. \tag{6}$$

Instance-wise margin is already defined on each class. But in practice, using the minimum margin of all pairs of instances is not robust, since noise or difficult instances may easily cause a meaningless value of γ_y^{inst}. Therefore we modify Eq. (6) into a more robust mean version

$$\bar{\gamma}_y^{\text{inst}}(f, V) = \left\{ \frac{1}{|V_y|} \sum_{a \in V_y} f^{(y)}\left(\boldsymbol{x}_a\right) - \frac{1}{|V \backslash V_y|} \sum_{b \in V \backslash V_y} f^{(y)}\left(\boldsymbol{x}_b\right) \right\}. \tag{7}$$

The objective vector for $\text{MMSE}_{\text{margin}}$ is defined as

$$\boldsymbol{g}(\boldsymbol{s}, V) = \left(\gamma^{\text{label}}\left(F_{\boldsymbol{s}}, V\right), \gamma^{\text{inst}}\left(F_{\boldsymbol{s}}, V\right), -|\boldsymbol{s}|\right), \tag{8}$$

where

$$\gamma^{\text{label}}\left(F_{\boldsymbol{s}}, V\right) = \frac{1}{l} \sum_{y} \bar{\gamma}_y^{\text{label}}\left(F_{\boldsymbol{s}}, V\right), \tag{9}$$

$$\gamma^{\text{inst}}\left(F_{\boldsymbol{s}}, V\right) = \min_{y} \bar{\gamma}_y^{\text{inst}}\left(F_{\boldsymbol{s}}, V\right). \tag{10}$$

Algorithm 1 MMSE

Input: Training data Tr, validation data V, objective modeling $\boldsymbol{g}$, evaluation criterion *eval* denoting the decision making process.

Output: An ensemble.

1: Train base learners $\{h_i\}_{i=1}^n$ using different training samples obtained by different sampling strategies.

2: Use NSGA-II to solve the problem $\arg\max_{\boldsymbol{s}} \boldsymbol{g}(\boldsymbol{s}, V)$ obtain a set of Pareto optimal solutions.

3: Present the optimal ensembles to the decision-maker and she selects an ensemble according to *eval*.

We use the average and minimum for $\bar{\gamma}_y^{\text{label}}$ and $\bar{\gamma}_y^{\text{inst}}$ respectively to emphasize different aspects of performance across classes. With the objective modeling in Eq. (8), the number of objectives is limited to 3, no matter how many classes there are. Meanwhile, the label-wise and instance-wise margins are related to common performance measures, and the third objective $-|\boldsymbol{s}|$ benefits the theoretical analysis. An analysis for $\text{MMSE}_{\text{margin}}$ is provided in Section 4.

The pseudocode of MMSE is shown in Algorithm 1. It applies to both $\text{MMSE}_{\text{class}}$ and $\text{MMSE}_{\text{margin}}$, the only difference is the objective modeling $\boldsymbol{g}$. NSGA-II [8] is adopted as the multi-objective optimization algorithm. It is a well-established multi-objective evolutionary algorithm suitable for such combinatorial multi-objective problems. It is suitable for $\text{MMSE}_{\text{margin}}$ with only three objectives and can achieve a theoretical guarantee of optimization time complexity as will be shown in Section 4. For consistency, we also use NSGA-II for $\text{MMSE}_{\text{class}}$. The evaluation criterion *eval* denotes the decision-maker's decision process after obtaining a set of Pareto-optimal solutions. When presenting the obtained solutions to the decision-maker, we can use multi-dimensional data visualization methods, such as parallel coordinates [20, 47], where Step C in Figure 2 is an example.

4. Theoretical analysis

4.1. Theoretical results

In this section, we prove that $\text{MMSE}_{\text{margin}}$ can optimize common multi-class performance measures with approximation guarantee. Detailed proofs for theorems will be given in Section 4.2.

As we have reduced the number of objectives in $\text{MMSE}_{\text{margin}}$, we need to analyze the expressiveness of the objective modeling. We now show that if the multi-class version of label-wise margin and instance-wise margins are optimized, then common multi-class imbalance measures can be optimized.

Proposition 1. *If all the label-wise margins on dataset D are positive, then Average Accuracy, G-mean, macro-F1, micro-F1 are optimized.*

Proposition 2. *If all the instance-wise margins on dataset D are positive, then macro-AUC and MAUC are optimized.*

Then we analyze the approximation guarantee of $\text{MMSE}_{\text{margin}}$, with NSGA-II being its multi-objective optimization algorithm. This analysis ensures that the two objectives of $\text{MMSE}_{\text{margin}}$ can be optimized and have a time complexity guarantee. Let the selector vector $\boldsymbol{s}$ represent a subset S of V by assigning $s_i = 1$ if the i-th base learner of V is in S and $s_i = 0$ otherwise. Obviously, γ^{label} and γ^{inst} are two set functions that are both non-monotone[3] and non-submodular[4]. Therefore, we introduce the ϵ-approximate monotonicity in Definition 1 and β-approximate submodularity in Definition 2 to characterize how close a set function g is to monotonicity and submodularity, respectively.

Definition 1 (ϵ-Approximate Monotonicity [21])**.** Let $\epsilon \geq 0$. A set function $g : 2^V \to \mathbb{R}$ is ϵ-approximately monotone if for any $S \subseteq V$ and $v \notin S$.

$$g(S \cup \{v\}) \geq g(S) - \epsilon.$$

It is easy to see that g is monotone iff $\epsilon = 0$.

Definition 2 (β-Approximate Submodularity [7])**.** Let $0 \leq \beta \leq 1$. A set function $g : 2^V \to \mathbb{R}$ is β-approximately submodular if for any $S, T \subseteq V$ and $v \in V$,

$$\sum_{v \in T \setminus S} (g(S \cup \{v\}) - g(S)) \geq \beta(g(S \cup T) - g(S)).$$

It is easy to see that g is submodular iff $\beta = 1$.

[3] A set function $g : 2^n \to \mathbb{R}$ is monotone if $\forall X \subseteq Y \subseteq V$, $g(X) \leq g(Y)$.

[4] A set function g is submodular if it satisfies the "diminishing returns" property, i.e., $\forall X \subseteq Y \subseteq V, \sum_{v \in Y \setminus X} (g(X \cup \{v\}) - g(X)) \geq g(X \cup Y) - g(X)$.

Assume the solutions in the first nondominated front will not be excluded from the population by NSGA-II. Let ϵ_1 and β_1 be the approximate monotonicity and approximate submodularity parameter of γ^{label}, respectively, ϵ_2 and β_2 be the approximate monotonicity and approximate submodularity parameter of γ^{inst}, respectively. Proposition 3 gives the approximation guarantee of $\text{MMSE}_{\text{margin}}$ on γ^{label} and γ^{inst}.

Proposition 3. *For the selective ensemble problem defined in Eq.* (8) *for* $MMSE_{margin}$, *the expected number of iterations of* NSGA-II *until finding a solution* $\boldsymbol{s}$ *with* $|\boldsymbol{s}| \leq m$ *and* $\gamma^{\text{label}} \geq (1 - e^{-\beta_1}) \cdot (\text{OPT}^{\text{label}} - m\epsilon_1)$, *and a solution* $\boldsymbol{t}$ *with* $|\boldsymbol{t}| \leq m$ *and* $\gamma^{\text{inst}} \geq (1 - e^{-\beta_2}) \cdot (\text{OPT}^{\text{inst}} - m\epsilon_2)$ *is* $O(n(\log n + m))$, *where* $\text{OPT}^{\text{label}}$ *and* OPT^{inst} *denote the optimal value of* γ^{label} *and the optimal value of* γ^{inst}, *respectively.*

Proof sketch. We first prove that with the approximate monotonicity and approximate submodularity assumption, we can always find an element to add to a set with certain improvements. Then by tracking the probability that such an improvement happens on the best solution in the population, we count the expected number of iterations required by NSGA-II to achieve the desired approximate guarantee. □

Remark 1. As Proposition 3 demonstrates, the multi-objective selective ensemble procedure of $\text{MMSE}_{\text{margin}}$ can achieve the approximate optimal value of average label-wise margin γ^{label} and minimum instance-wise margin γ^{inst}. These two margins are statistics of label-wise margin γ_i^{label} and instance-wise margin γ_y^{inst}. And from Proposition 1 and Proposition 2 we know that, if γ_i^{label} and γ_y^{inst} are optimized on all instances and all classes, common multi-class performance measures are optimized.

4.2. Proofs

4.2.1. Proof of Proposition 1

Proof. If label-wise margin is positive on an instance $\boldsymbol{x}_i$, we have $f^{(y)}(\boldsymbol{x}_i) > f^{(y' \neq y)}(\boldsymbol{x}_i)$. Therefore,

$$\forall \boldsymbol{x}_i, h(\boldsymbol{x}_i) = \arg\max_j f^{(j)}(\boldsymbol{x}_i) = y \ .$$

Then we have $\forall y, \frac{1}{|D_y|} \sum_{i \in D_y} \mathbb{1}_{[h(\boldsymbol{x}_i) = y]} = 1$. Hence, Avg. $\text{Acc}(h) = 1$, $\text{G-mean}(h) = 1$.

We also have $\sum\limits_{i \in D_y} \mathbb{1}_{[h(\boldsymbol{x}_i)=y]} = |D_y|$, therefore

$$\text{macro-F1}(h) = \frac{1}{l} \sum_{y=1}^{l} \frac{2|D_y|}{|D_y| + |D_y|} = 1,$$

$$\text{micro-F1}(h) = \frac{2 \sum_{j=1}^{l} |D_y|}{|D| + \sum_{j=1}^{l} |D_y|} = \frac{2|D|}{|D| + |D|} = 1.$$

322 $\square$

323 *4.2.2. Proof of Proposition 2*

Proof. If instance-wise margin on label y is positive, then

$$f^{(y)}(\boldsymbol{x}_a) > f^{(y)}(\boldsymbol{x}_b), \forall a \in D_y, b \in D \backslash D_y \ .$$

324 Hence,

$$\begin{aligned} \mathcal{S}_{\text{macro}}^{y} &= \left\{(a,b) \in D_y \times \{D \backslash D_y\} \mid f^{(y)}(\boldsymbol{x}_a) \geq f^{(y)}(\boldsymbol{x}_b)\right\} \\ &= |D_y||D \backslash D_y| \ . \end{aligned}$$

325 If it holds for all y, then

$$\text{macro-AUC}(f) = \frac{1}{l} \sum_{y=1}^{l} \frac{\mathcal{S}_{\text{macro}}^{y}}{|D_y|\,|D \backslash D_y|} = 1 \ .$$

326 We also have

$$\begin{aligned} \hat{A}(i \mid j) &= \frac{1}{|D_i||D_j|} \left\{(a,b) \in D_i \times D_j \mid f^{(j)}(\boldsymbol{x}_a) \geq f^{(j)}(\boldsymbol{x}_b)\right\} \\ &= 1 \ , \end{aligned}$$

327 and

$$\hat{A}(i,j) = [\hat{A}(i \mid j) + \hat{A}(j \mid i)]/2 = 1 \ .$$

328 Therefore, MAUC $(f) = \frac{2}{l(l-1)} \sum_{i<j} \hat{A}(i,j) = 1.$ $\square$

4.2.3. Proof of Proposition 3

Proof. The proof relies on Lemma 1 and Lemma 2, which are inspired by [31]. The detailed proofs of these lemmas are presented later.

Lemma 1. *Assume that a set function g is ϵ-approximately monotone as in Definition 1 and β-approximately submodular as in Definition 2. For any $\boldsymbol{s} \in \{0,1\}^n$ with $|\boldsymbol{s}| < m$, there exists one element $v \notin \boldsymbol{s}$ such that*

$$g(\boldsymbol{s} \cup \{v\}) - g(\boldsymbol{s}) \geq \beta/m \cdot (\mathrm{OPT} - g(\boldsymbol{s})) - \beta \cdot \epsilon \ ,$$

where m is the size constraint.

Assume that the number of selected base learners does not exceed m, Lemma 1 proves that for any $\boldsymbol{s} \in \{0,1\}^n$ with $|\boldsymbol{s}| < m$, there always exists another element, the inclusion of which can bring an improvement on g roughly proportional to the current distance to the optimum.

Lemma 2. *To maximize an ϵ-approximately monotone and β-approximately submodular set function g, the expected number of iterations of the* NSGA-II *until finding a solution $\boldsymbol{s}$ with $|\boldsymbol{s}| \leq m$ and $g(\boldsymbol{s}) \geq (1 - e^{-\beta}) \cdot (\mathrm{OPT} - m\epsilon)$ is $O(n(\log n + m))$, where* OPT *denotes the optimal value.*

Lemma 2 proves the approximation guarantee of NSGA-II on any ϵ-approximately monotone and β-approximately submodular set function g. As in previous analyses (e.g.,[2, 12]), we may assume that there is a set S_d of m "dummy" elements whose marginal contribution to any set is 0, i.e., for any $S \subseteq V, g(S) = g(S \backslash S_d)$.

By substituting the parameters ϵ_1 and β_1 of γ^{label} as well as ϵ_2 and β_2 of γ^{inst} into Lemma 2, the theorem can be directly obtained. □

Proof of Lemma 1. Let $\boldsymbol{s}^*$ be an optimal solution containing at most m items, i.e., $\boldsymbol{s}^* = \arg\max_{\boldsymbol{s} \in \{0,1\}^n, |\boldsymbol{s}| \leq m} g(\boldsymbol{s})$, and OPT denote the optimal value, i.e., $g(\boldsymbol{s}^*) = \mathrm{OPT}$. We denote the elements in $\boldsymbol{s} \backslash \boldsymbol{s}^*$ by $u_1^*, u_2^*, \cdots, u_t^*$, where $t = |\boldsymbol{s} \backslash \boldsymbol{s}^*|$. Note that $t < m$ as $|\boldsymbol{s}| < m$. Because g is ϵ-approximately monotone, we have

$$\begin{aligned} g(\boldsymbol{s}^* \cup \boldsymbol{s}) &= g(\boldsymbol{s}^* \cup \{u_1^*, u_2^*, \cdots, u_t^*\}) \\ &\geq g(\boldsymbol{s}^* \cup \{u_1^*, u_2^*, \cdots, u_{t-1}^*\}) - \epsilon \\ &\geq \cdots \geq g(\boldsymbol{s}^*) - t\epsilon \\ &\geq g(\boldsymbol{s}^*) - m\epsilon, \end{aligned} \tag{11}$$

where the first three inequalities hold by Definition 1. We denote the elements in $\boldsymbol{s}^*\backslash\boldsymbol{s}$ by $v_1^*, v_2^*, \cdots, v_l^*$, where $l = |\boldsymbol{s}^*\backslash\boldsymbol{s}| \leq m$. Then, we have

$$\begin{aligned}
&g(\boldsymbol{s}^*) - g(\boldsymbol{s}) - m\epsilon \leq g(\boldsymbol{s} \cup \boldsymbol{s}^*) - g(\boldsymbol{s}) \\
&= g(\boldsymbol{s} \cup \{v_1^*, v_2^*, \cdots, v_l^*\}) - g(\boldsymbol{s}) \\
&\leq \frac{1}{\beta} \sum_{j=1}^{l} (g(\boldsymbol{s} \cup \{v_j^*\}) - g(\boldsymbol{s})),
\end{aligned} \tag{12}$$

where the first inequality holds by Eq. (11), the first equality holds by the definition of $\boldsymbol{s}^*\backslash\boldsymbol{s}$, and the last inequality holds by Definition 2. Let $v^* = \arg\max_{v \in V/\boldsymbol{s}} g(\boldsymbol{s} \cup \{v\})$. Eq. (12) implies that

$$g(\boldsymbol{s}^*) - g(\boldsymbol{s}) - m\epsilon \leq l/\beta \cdot (g(\boldsymbol{s} \cup \{v^*\}) - g(\boldsymbol{s})).$$

Due to the existence of m dummy elements and $|\boldsymbol{s}| < m$, there must exist one dummy element $v \notin \boldsymbol{s}$ satisfying $g(\boldsymbol{s} \cup \{v\}) - g(\boldsymbol{s}) = 0$; this implies that $g(\boldsymbol{s} \cup \{v^*\}) - g(\boldsymbol{s}) \geq 0$. As $l \leq m$, we have $g(\boldsymbol{s}^*) - g(\boldsymbol{s}) - m\epsilon \leq m/\beta \cdot (g(\boldsymbol{s} \cup \{v^*\}) - g(\boldsymbol{s}))$, leading to $g(\boldsymbol{s} \cup \{v^*\}) - g(\boldsymbol{s}) \geq \beta/m \cdot (\mathrm{OPT} - g(\boldsymbol{s})) - \beta \cdot \epsilon$. □

Proof of Lemma 2. We divide the optimization process into two phases: (1) starts from an initial population P with constant size N and finishes after including the special solution $\mathbf{0}$ (i.e., empty set) in population; (2) starts after phase (1) and finishes after finding a solution with the desired approximation guarantee.

For phase (1), we consider the minimum number of 1-bits of the solutions in the population P, denoted by J_{min}. That is, $J_{min} = \min\{|\boldsymbol{s}| \mid \boldsymbol{s} \in P\}$. Assume that currently $J_{min} = i > 0$, and let $\boldsymbol{s}$ be a corresponding solution, i.e., $|\boldsymbol{s}| = i$. It is easy to see that J_{min} cannot increase because $\boldsymbol{s}$ cannot be weakly dominated by a solution with more 1-bits. In each iteration of NSGA-II, to decrease J_{min}, it is sufficient to select $\boldsymbol{s}$ and flip only one 1-bit of $\boldsymbol{s}$ by the bit-wise mutation operator. This is because the newly generated solution $\boldsymbol{s}'$ now has the smallest number of 1-bits (i.e., $|\boldsymbol{s}'| = i - 1$) and no solution in P can dominate it; thus it will be included into P. In our setting, the bit-wise mutation is performed with a probability of $1/2$, randomly selecting a parent solution and independently flipping each bit with a probability of $1/n$. Thus, the probability of selecting $\boldsymbol{s}$ from the population and flipping only one 1-bit of $\boldsymbol{s}$ by bit-wise mutation is $\frac{1}{2} \cdot \frac{1}{N} \cdot \frac{i}{n}(1 - 1/n)^{n-1} \geq \frac{i}{2enN}$, since the probability of operating bit-wise mutation is $\frac{1}{2}$, the probability of selecting $\boldsymbol{s}$ is $\frac{1}{N}$ due to uniform selection and $\boldsymbol{s}$ has i 1-bits.

In each iteration of NSGA-II, there are N offspring solutions to be generated. Thus, the probability of decreasing J_{min} by at least 1 in each iteration of NSGA-II is at least $N \cdot \frac{i}{2enN} = \frac{i}{2en}$. Note that $J_{min} \leq n$. We can then get that the expected number of iterations of phase (1) (i.e., J_{min} reaches $\mathbf{0}$) is at most $\sum_{i=1}^{n} \frac{2en}{i} = O(n \log n)$. Note that the solution $\mathbf{0}$ will always be kept in P once generated, since it has the smallest subset size 0 and no other solution can weakly dominate it.

For phase (2), we consider a quantity J_{max}, which is defined as

$$\begin{aligned} J_{max} = \max\{j \in \{0, 1, \cdots, m\} \mid \exists \boldsymbol{s} \in P : \\ |\boldsymbol{s}| \leq j \wedge g(\boldsymbol{s}) \geq \left(1 - \left(1 - \frac{\beta}{m}\right)^j\right) \cdot (\mathrm{OPT} - m\epsilon)\}. \end{aligned}$$

That is, J_{max} denotes the maximum value of $j \in \{0, 1, \cdots, m\}$ such that in the population P, there exists a solution $\boldsymbol{s}$ with $|\boldsymbol{s}| \leq j$ and $g(\boldsymbol{s}) \geq (1 - (1 - \beta/m)^j) \cdot (\mathrm{OPT} - m\epsilon)$. The solution that satisfies this condition may not be unique in the population, but there must be one in the first front. We consider the solution $\boldsymbol{s}$ in the first front of NSGA-II. We analyze the expected number of iterations until $J_{max} = m$, which implies that there exists one solution $\boldsymbol{s}$ in P satisfying that $|\boldsymbol{s}| \leq m$ and $g(\boldsymbol{s}) \geq (1 - (1 - \beta/m)^m) \cdot (\mathrm{OPT} - m\epsilon) \geq (1 - e^{-\beta}) \cdot (\mathrm{OPT} - m\epsilon)$. That is, the desired approximation guarantee is reached.

The current value of J_{max} is at least 0, since the population P contains the solution $\mathbf{0}$, which will always be kept in P once generated. Assume that currently $J_{max} = i < m$. Let $\boldsymbol{s}$ be a corresponding solution with the value i, i.e., $|\boldsymbol{s}| \leq i$ and $g(\boldsymbol{s}) \geq (1 - (1 - \beta/m)^i) \cdot (\mathrm{OPT} - m\epsilon)$. It is easy to see that J_{max} cannot decrease because cleaning $\boldsymbol{s}$ from P implies that $\boldsymbol{s}$ is weakly dominated by a newly generated solution $\hat{\boldsymbol{s}}$, which must satisfy that $|\hat{\boldsymbol{s}}| \leq |\boldsymbol{s}|$ and $g(\hat{\boldsymbol{s}}) \geq g(\boldsymbol{s})$. By Lemma 1, we know that flipping one specific 0-bit of $\boldsymbol{s}$ (i.e., adding a specific element) can generate a new solution $\boldsymbol{s}'$, which satisfies $g(\boldsymbol{s}') - g(\boldsymbol{s}) \geq \frac{\beta}{m}(\mathrm{OPT} - g(\boldsymbol{s})) - \beta\epsilon$. Then, we have

$$\begin{aligned} g(\boldsymbol{s}') &\geq \left(1 - \frac{\beta}{m}\right) g(\boldsymbol{s}) + \frac{\beta}{m}\mathrm{OPT} - \beta\epsilon \\ &\geq \left(1 - \left(1 - \frac{\beta}{m}\right)^{i+1}\right) \cdot (\mathrm{OPT} - m\epsilon), \end{aligned}$$

where the last inequality is derived by $g(\boldsymbol{s}) \geq (1-(1-\beta/m)^i) \cdot (\mathrm{OPT} - m\epsilon)$. After generating $\boldsymbol{s}'$, it can be guaranteed that there must be a solution weakly dominant $\boldsymbol{s}'$ in the first front, and $J_{max} \geq i+1$. Thus, J_{max} can increase by at least 1 in one iteration with probability at least $N \cdot \frac{1}{N} \cdot \frac{1}{2} \cdot \frac{1}{n}(1-\frac{1}{n})^{n-1}$, where $N \cdot \frac{1}{N}$ is the expectation of selecting $\boldsymbol{s}$ as a parent solution when the NSGA-II generates N offspring solutions in each iteration, $\frac{1}{2}$ is the probability of operating bit-wise mutation to the parent solution $\boldsymbol{s}$ and $\frac{1}{n}(1-\frac{1}{n})^{n-1}$ is the probability of flipping a specific bit of $\boldsymbol{s}$ while keeping other bits unchanged. This implies that it needs at most $2en$ expected number of iterations to increase J_{max}. Thus, after at most $2emn = O(mn)$ iterations in expectation, J_{max} must have reached m.

Then, by summing up the expected number of iterations of two phases, we get that the expected number of iterations of NSGA-II for finding a solution $\boldsymbol{s}$ with $|\boldsymbol{s}| \leq m$ and $g(\boldsymbol{s}) \geq (1-e^{-\beta}) \cdot (\mathrm{OPT} - m\epsilon)$ is $O(n(\log n + m))$. □

5. Experiments

In this section, we show with experiments that our methods can efficiently generate many diverse and highly competitive classification models.

5.1. Experimental setup

5.1.1. Compared methods

Considering that our methods employ multiple rebalancing strategies (specifically, all are forms of undersampling) and decision tree ensembles, we select compared methods that share these key components. The compared methods must be capable of handling multi-class problems. Unlike our methods, which offer a wide range of choices for decision-makers, existing methods can only use predetermined rebalancing strategies and offer only one solution.

We compare our proposed methods $\mathrm{MMSE}_{\mathrm{class}}$ and $\mathrm{MMSE}_{\mathrm{margin}}$ to the following six state-of-the-art ensemble-based multi-class imbalanced learning methods.

- SMOTE [4]: It is a synthesized oversampling algorithm. We oversample all the other classes to have the same training samples as the majority class. Then we use multi-class AdaBoost [15] classifier on the rebalanced dataset.

- EasyEnsemble [25]: It uses undersampling without replacement to generate multiple balanced training subsets and trains a multi-class AdaBoost on each of them, then combines them.

- BalancedRF [6]: It uses undersampling with replacement to generate multiple balanced subsets first, then trains a decision tree with random feature selection on each of the subsets, then combines them.

- SMOTEBoost [5]: It adds a step of synthesized oversampling to make a balanced training set in each round of boosting. We extend it to multi-class cases in a way similar to multi-class AdaBoost.

- MDEP [37]: It is a multi-objective selective ensemble method that simultaneously optimizes validation error, ensemble size, and margin distribution. We use rebalanced base learners as input.

- DEP [19]: It is a two-stage selective ensemble method that first optimizes the combination error and then solely optimizes the validation error. We use rebalanced base learners as input.

5.1.2. Datasets

We conduct experiments on ten multi-class datasets, including seven LIBSVM datasets [3], one UCI dataset, and two real-world application datasets. The number of classes varies from 3 (*dna*) to 26 (*letter*). The number of features varies from 6 (*car*) to 2565 (*miRNA*). Table 2 records the number of training instances of each class. In the last column we show the imbalance rate of each dataset, which is calculated by dividing the number of samples in the largest class by the number of samples in the smallest class.

Among the benchmark datasets, *car*, *dna* is naturally imbalanced, and *vehicle*, *satimage*, *pendigits*, *usps*, *letter*, *segment* are artificially made imbalanced.

The real-world dataset *acoustic* is naturally imbalanced. The task aims at predicting the function of an acoustic system. The dataset has 21 continuous features, indicating the angle of the placements of 21 acoustic units that determine the function of the system. The four classes are namely *amplify*, *minify*, *cage*, *harvest*. The first two classes mean that the sound will decrease or increase inside the acoustic system. *cage* means there is a sharp decrease in the sound field that the system becomes a cage to shield from the sound [34]. *harvest* means the energy is greatly magnified in a small area that it can

Table 2: Information of the datasets

Dataset	Number of training instances in each class	Imbalance rate
car	[307 55 968 52]	18.6
vehicle	[170 140 110 80]	2.1
dna	[507 487 1074]	2.2
satimage	[993 486 956 414 425 809]	2.4
pendigits	[700 600 500 400 300 200 100 70 50 30]	23.3
usps	[800 600 400 400 300 200 100 80 60 50]	16
letter	[520 500 480 460 440 420 400 380 360 340 320 300 280 260 240 220 200 180 160 140 120 100 80 60 40 20]	26
segment	[264 210 160 110 80 50 30]	8.8
acoustic	[2477 723 2674 526]	5.1
miRNA	[2207 256 92 92 92 92 92 92 70 64]	34.5

be captured in the form of electricity [1, 28], meanwhile can be dangerous when the energy focusing is undesired. The extreme cases *cage* and *harvest* naturally happen less often.

The real-world dataset *miRNA* is naturally imbalanced. Circulating microRNAs (miRNAs) are promising biomarkers that could be applied to early detection of cancer. We experimented with data processed from serum miRNA profiles [45], which has 2565 features, each one of which denotes the expression level of certain miRNA[5]. The ten classes are *Healthy*, *Ovarian Cancer*, *Breast Cancer*, *Colorectal Cancer*, *Gastric Cancer*, *Lung Cancer*, *Pancreatic Cancer*, *Sarcoma*, *Esophageal Cancer* and *Hepatocellular Carcinoma*.

5.1.3. Configurations

Experiments were run on a Windows 10 machine with a 3.40 GHz Intel i7-13700KF CPU and 32 GB memory. Each dataset is randomly partitioned into training and test sets, and this partitioning process is repeated 10 times independently and the average result is reported. In the training process of all the methods, the training set is further partitioned into model training set and validation set with ratio 3:1 and with stratified sampling, where the validation set is used for selective ensemble and model selection.

[5]The miRNA data can be downloaded from `https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE106817`

For the proposed methods $\text{MMSE}_{\text{class}}$ and $\text{MMSE}_{\text{margin}}$, 100 data subsets are generated each randomly using 'not minority' or 'middle' sampling strategies, with 'not minority' we undersample all the other classes to have the same training instances as the minority class, and with 'middle' we first randomly select a class, then undersample classes bigger than that class to have the same number of training instances as that class. Therefore, a class has different undersampling rates in different subsets. For each data subset, the base learner is randomly chosen from an Adaboost with 10 trees or a random forest with 5 trees. The population size of NSGA-II is set to 100, and the maximum number of generations is 100. When generating new solutions, we randomly perform crossover or mutation with probability 0.5 respectively. When doing crossover, we randomly select two parent solutions uniformly, and randomly select the position of encodings to combine them into a new solution. When performing mutation, we randomly select a parent solution and operate a bit-wise mutation that independently flips each bit of solution with probability $1/n$. Considering the estimation of performance on the validation set is not accurate, inspired by PONSS [30] that deals with noisy problems, we use a domination strategy with a threshold.

The hyperparameters of the compared methods are selected based on the performance on the validation set. Specifically, we rank the performance of each hyperparameter value, and then select the hyperparameter with the best average rank on the six performance measures. The number of neighbors in SMOTE is selected from $\{3, 5\}$. The number of base learners in EasyEnsemble is selected from $\{10, 20, 50\}$ and the number of trees in each Adaboost is set to 10. The number of decision trees in BalancedRF is selected from $\{10, 20, 50\}$. The maximum number of base learners in SMOTEBoost is selected from $\{10, 20, 50\}$. As one of the objectives of MMSE is to reduce the size of ensemble, the number of base learners output by MMSE is less than 50. The above settings ensure that the obtained models contain roughly the same number of individual learners. For MDEP, the individual learners are the same as MMSE, the population size is 100 and the maximum number of generations is 100. For DEP, the data subsets are generated the same as MMSE, the base learners are decision trees. The maximum number of generations in each stage is 50, and the population size is 100 for both stages. This setting ensures that the total number of fitness evaluations during MDEP and DEP is the same as that of MMSE.

5.2. Results and discussion

We show that our proposed methods are superior in both Scenario I and Scenario II decision-making processes.

5.2.1. Scenario I

After $\text{MMSE}_{\text{class}}$ or $\text{MMSE}_{\text{margin}}$ obtains a collection of diverse optimal solutions, we examine them with varied performance measures as described in Section 3.1. In detail, we choose the best ensemble on the validation set under each measure and report the corresponding result on the test set. And for the compared methods that each generate a single model only, we simply report the model performance on all six measures.

Table 3 and Table 4 show the results on six common performance measures, where the best result on each dataset and each measure is bolded. From the experimental results, our methods $\text{MMSE}_{\text{class}}$ and $\text{MMSE}_{\text{margin}}$ outperform other methods in all evaluation metrics on almost all the datasets, and obtain very competitive results on the others.

Specifically, on *letter* dataset, $\text{MMSE}_{\text{margin}}$ has a better average score than $\text{MMSE}_{\text{class}}$ on all the performance measures. This is because *letter* has 26 classes, which is a relatively large number. For $\text{MMSE}_{\text{class}}$, this means the number of objectives is large and the optimization process becomes difficult. At this point, $\text{MMSE}_{\text{margin}}$ is able to perform well because the number of objectives remains unchanged. This demonstrates that the objective modeling in $\text{MMSE}_{\text{margin}}$, which incorporates margin to aggregate the performances of the classes, is proved to be successful. On the other hand, $\text{MMSE}_{\text{class}}$ has its own advantages. For example, on *acoustic* dataset, which has only 4 classes, $\text{MMSE}_{\text{class}}$ outperforms $\text{MMSE}_{\text{margin}}$ on all the measures.

To show a summary of the compared methods on all datasets, Figure 3 plots the average rank of each method on each performance measure. According to the Friedman-Nemenyi test at significance level 0.1, we can observe that 1) Our methods $\text{MMSE}_{\text{class}}$ and $\text{MMSE}_{\text{margin}}$ achieve the best average rank on all the performance measures, and they are roughly equally good. 2) MDEP, BalancedRF and SMOTE are significantly worse than our methods on all the performance measures. 3) Compared with DEP, SMOTEBoost, and EasyEnsemble, our methods have no significant advantage, but have better average rank on all the performance measures. This indicates the high competitiveness of our method on these measures, and in Section 5.2.2, we will further show the richness of the solutions we provide.

Table 3: Experimental results on benchmark datasets of common performance measures. The results are shown in mean±std.(rank) of 10 times of running. The smaller the rank, the better the performance. The best accuracy is highlighted in bold type. An entry is marked with a bullet '•' if the method is significantly worse than MMSE$_{\text{class}}$ or MMSE$_{\text{margin}}$ based on the Wilcoxon rank-sum test with confidence level 0.1.

Dataset	Method	avg. acc	G-mean	F1-macro	F1-micro	macro-AUC	MAUC
car	SMOTE	0.912±0.018(6)•	0.908±0.018(6)•	0.909±0.011(4)•	0.953±0.013(3)	0.967±0.017(8)•	0.962±0.020(8)•
	EasyEnsemble	0.911±0.017(7)•	0.908±0.017(7)•	0.797±0.032(7)•	0.844±0.019(7)•	0.976±0.005(6)•	0.989±0.003(5)•
	BalancedRF	0.918±0.024(5)•	0.915±0.024(5)•	0.833±0.041(6)•	0.873±0.021(6)•	0.979±0.005(5)•	0.988±0.005(6)•
	SMOTEBoost	0.928±0.038(4)•	0.923±0.043(4)•	0.939±0.034(2)	**0.968±0.017(1)**	**0.997±0.002(1)**	0.995±0.003(4)•
	MDEP	0.884±0.027(8)•	0.880±0.029(8)•	0.796±0.068(8)•	0.843±0.055(8)•	0.967±0.014(7)•	0.980±0.008(7)•
	DEP	0.949±0.016(3)•	0.948±0.016(3)•	0.907±0.022(5)•	0.930±0.019(5)•	0.994±0.003(4)•	0.997±0.002(3)•
	MMSE$_{\text{class}}$(ours)	0.957±0.023(2)	0.956±0.023(2)	0.929±0.030(3)	0.953±0.016(4)	0.996±0.003(3)	0.998±0.002(2)
	MMSE$_{\text{margin}}$(ours)	**0.964±0.020(1)**	**0.963±0.021(1)**	**0.945±0.024(1)**	0.962±0.016(2)	0.997±0.002(2)	**0.998±0.002(1)**
vehicle	SMOTE	0.661±0.041(8)•	0.641±0.044(8)•	0.666±0.039(8)•	0.660±0.041(8)•	0.774±0.027(8)•	0.774±0.027(8)•
	EasyEnsemble	0.729±0.031(5)	0.689±0.041(6)	0.721±0.033(5)	0.726±0.031(5)	0.919±0.008(2)	**0.920±0.008(1)**
	BalancedRF	0.727±0.024(6)	0.692±0.034(5)	0.720±0.026(6)	0.725±0.024(6)	0.909±0.008(6)•	0.910±0.008(6)•
	SMOTEBoost	0.737±0.020(2)	**0.704±0.032(1)**	**0.732±0.023(1)**	**0.735±0.021(1)**	0.914±0.011(5)	0.915±0.011(5)
	MDEP	0.699±0.016(7)•	0.662±0.026(7)•	0.696±0.019(7)•	0.697±0.017(7)•	0.895±0.009(7)•	0.895±0.009(7)•
	DEP	0.730±0.021(4)	0.692±0.033(4)	0.724±0.023(4)	0.728±0.022(4)	0.917±0.007(4)	0.918±0.007(4)
	MMSE$_{\text{class}}$(ours)	**0.738±0.032(1)**	0.698±0.041(2)	0.731±0.031(2)	0.734±0.030(2)	**0.919±0.009(1)**	0.919±0.009(2)
	MMSE$_{\text{margin}}$(ours)	0.732±0.027(3)	0.696±0.037(3)	0.728±0.026(3)	0.732±0.025(3)	0.918±0.008(3)	0.919±0.007(3)
dna	SMOTE	0.882±0.015(8)•	0.881±0.016(8)•	0.879±0.015(8)•	0.893±0.014(8)•	0.918±0.022(8)•	0.917±0.022(8)•
	EasyEnsemble	0.938±0.006(4)•	0.937±0.006(4)•	0.923±0.007(6)•	0.928±0.007(6)•	0.991±0.002(4)•	0.992±0.002(4)•
	BalancedRF	0.933±0.011(6)•	0.932±0.011(5)•	0.925±0.011(5)•	0.933±0.009(5)•	0.989±0.002(6)•	0.989±0.002(5)•
	SMOTEBoost	0.933±0.012(5)•	0.932±0.012(6)•	0.931±0.011(4)	0.939±0.011(3)	0.989±0.004(5)•	0.989±0.003(6)•
	MDEP	0.920±0.008(7)•	0.919±0.008(7)•	0.906±0.013(7)•	0.913±0.013(7)•	0.982±0.003(7)•	0.982±0.003(7)•
	DEP	0.942±0.007(2)	0.942±0.007(2)	0.932±0.007(2)	0.938±0.007(4)	0.992±0.002(3)	0.992±0.002(3)
	MMSE$_{\text{class}}$(ours)	**0.944±0.007(1)**	**0.943±0.007(1)**	**0.934±0.009(1)**	**0.941±0.009(1)**	0.993±0.002(2)	0.993±0.002(2)
	MMSE$_{\text{margin}}$(ours)	0.940±0.009(3)	0.940±0.009(3)	0.932±0.012(3)	0.939±0.011(2)	**0.993±0.001(1)**	**0.993±0.001(1)**
satimage	SMOTE	0.825±0.011(8)•	0.814±0.015(8)•	0.824±0.010(8)•	0.847±0.008(8)•	0.897±0.006(8)•	0.895±0.007(8)•
	EasyEnsemble	0.892±0.010(4)	0.888±0.011(4)	0.887±0.009(4)•	0.901±0.008(5)•	0.988±0.002(4)	0.987±0.002(4)
	BalancedRF	0.885±0.010(5)•	0.880±0.012(5)•	0.884±0.009(6)•	0.900±0.008(6)•	0.986±0.002(6)•	0.986±0.002(6)•
	SMOTEBoost	0.878±0.008(6)•	0.861±0.011(7)•	0.886±0.008(5)•	**0.909±0.007(2)**	0.986±0.002(5)•	0.986±0.002(5)•
	MDEP	0.876±0.009(7)•	0.871±0.009(6)•	0.873±0.010(7)•	0.890±0.011(7)•	0.983±0.004(7)•	0.982±0.004(7)•
	DEP	0.895±0.010(2)	0.890±0.011(3)	0.893±0.008(2)	0.908±0.007(3)	0.988±0.002(3)	0.988±0.002(3)
	MMSE$_{\text{class}}$(ours)	0.894±0.011(3)	0.892±0.012(2)	0.893±0.011(3)	0.908±0.009(4)	**0.989±0.002(1)**	**0.988±0.002(1)**
	MMSE$_{\text{margin}}$(ours)	**0.899±0.012(1)**	**0.894±0.014(1)**	**0.895±0.009(1)**	0.909±0.008(1)	0.988±0.002(2)	0.988±0.002(2)
pendigits	SMOTE	0.875±0.012(8)•	0.864±0.016(8)•	0.872±0.013(8)•	0.877±0.012(8)•	0.931±0.007(8)•	0.931±0.007(8)•
	EasyEnsemble	0.949±0.008(4)•	0.948±0.008(4)•	0.949±0.008(4)•	0.949±0.008(4)•	0.998±0.001(4)•	0.998±0.001(4)•
	BalancedRF	0.939±0.011(5)•	0.937±0.011(5)•	0.939±0.011(5)•	0.939±0.011(5)•	0.997±0.001(6)•	0.997±0.001(6)•
	SMOTEBoost	0.931±0.017(6)•	0.922±0.024(7)•	0.928±0.020(7)•	0.932±0.017(6)•	0.997±0.002(5)•	0.997±0.002(5)•
	MDEP	0.930±0.011(7)•	0.928±0.011(6)•	0.930±0.011(6)•	0.931±0.011(7)•	0.996±0.002(7)•	0.996±0.002(7)•
	DEP	**0.963±0.005(1)**	**0.962±0.006(1)**	0.963±0.005(3)	0.963±0.005(3)	0.999±0.000(3)	0.999±0.000(3)
	MMSE$_{\text{class}}$(ours)	0.959±0.006(3)	0.958±0.006(3)	0.963±0.006(2)	**0.964±0.006(1)**	0.999±0.000(2)	**0.999±0.000(1)**
	MMSE$_{\text{margin}}$(ours)	0.962±0.007(2)	0.962±0.007(2)	**0.964±0.006(1)**	0.963±0.006(2)	**0.999±0.000(1)**	0.999±0.000(2)
usps	SMOTE	0.798±0.012(8)•	0.789±0.014(8)•	0.801±0.012(8)•	0.821±0.009(8)•	0.889±0.006(8)•	0.888±0.007(8)•
	EasyEnsemble	0.916±0.006(4)•	0.916±0.006(4)•	0.916±0.005(4)•	0.924±0.004(4)•	0.995±0.001(4)•	0.995±0.001(4)•
	BalancedRF	0.896±0.009(5)•	0.895±0.010(5)•	0.897±0.008(6)•	0.907±0.007(6)•	0.992±0.001(6)•	0.991±0.001(6)•
	SMOTEBoost	0.893±0.008(6)•	0.887±0.008(6)•	0.899±0.007(5)•	0.910±0.006(5)•	0.993±0.001(5)•	0.992±0.001(5)•
	MDEP	0.887±0.014(7)•	0.884±0.015(7)•	0.889±0.015(7)•	0.900±0.014(7)•	0.989±0.005(7)•	0.988±0.005(7)•
	DEP	**0.922±0.006(1)**	**0.920±0.007(1)**	0.924±0.006(2)	0.931±0.005(2)	0.996±0.001(3)	0.995±0.001(3)
	MMSE$_{\text{class}}$(ours)	0.921±0.008(2)	0.920±0.008(2)	**0.926±0.007(1)**	**0.934±0.006(1)**	**0.996±0.001(1)**	0.995±0.001(2)
	MMSE$_{\text{margin}}$(ours)	0.920±0.007(3)	0.919±0.008(3)	0.924±0.007(3)	0.931±0.007(3)	0.996±0.001(2)	**0.995±0.001(1)**

Continued on Table 4

Table 4: Experimental results on benchmark datasets of common performance measures (continued). The results are shown in mean±std.(rank) of 10 times of running. The smaller the rank, the better the performance. The best accuracy is highlighted in bold type. An entry is marked with a bullet '•' if the method is significantly worse than $\text{MMSE}_{\text{class}}$ or $\text{MMSE}_{\text{margin}}$ based on the Wilcoxon rank-sum test with confidence level 0.1.

Dataset	Method	avg. acc	G-mean	F1-macro	F1-micro	macro-AUC	MAUC
letter	SMOTE	0.769±0.006(8)●	0.761±0.007(8)●	0.768±0.006(8)●	0.769±0.006(8)●	0.880±0.003(8)●	0.880±0.003(8)●
	EasyEnsemble	0.836±0.007(5)●	0.834±0.007(5)●	0.838±0.006(5)●	0.837±0.007(5)●	0.993±0.001(4)●	0.993±0.001(4)●
	BalancedRF	0.804±0.006(7)●	0.801±0.007(7)●	0.806±0.006(7)●	0.805±0.006(7)●	0.983±0.001(6)●	0.983±0.001(6)●
	SMOTEBoost	0.875±0.006(4)●	0.866±0.007(4)●	0.873±0.006(4)●	0.875±0.006(4)●	0.990±0.001(5)●	0.990±0.001(5)●
	MDEP	0.810±0.058(6)●	0.802±0.058(6)●	0.809±0.056(6)●	0.810±0.057(6)●	0.978±0.020(7)●	0.978±0.020(7)●
	DEP	0.892±0.005(3)	0.885±0.006(3)	0.891±0.005(3)●	0.893±0.005(3)	**0.996±0.001(1)**	**0.996±0.001(1)**
	$\text{MMSE}_{\text{class}}$(ours)	0.892±0.004(2)	0.887±0.003(2)	0.892±0.002(2)	0.894±0.003(2)	0.996±0.001(3)	0.996±0.001(3)
	$\text{MMSE}_{\text{margin}}$(ours)	**0.894±0.002(1)**	**0.888±0.002(1)**	**0.893±0.003(1)**	**0.895±0.003(1)**	0.996±0.000(2)	0.996±0.000(2)
segment	SMOTE	0.934±0.009(7)●	0.929±0.010(7)●	0.932±0.009(7)●	0.934±0.009(7)●	0.961±0.005(8)●	0.961±0.005(8)●
	EasyEnsemble	0.953±0.009(4)●	0.952±0.010(4)●	0.953±0.009(4)●	0.953±0.009(4)●	0.997±0.001(4)●	0.997±0.001(4)●
	BalancedRF	0.931±0.008(8)●	0.927±0.009(8)●	0.930±0.008(8)●	0.931±0.008(8)●	0.994±0.002(7)●	0.994±0.002(7)●
	SMOTEBoost	0.949±0.007(5)●	0.945±0.009(5)●	0.948±0.008(5)●	0.949±0.007(5)●	0.996±0.001(5)●	0.996±0.001(5)●
	MDEP	0.942±0.012(6)●	0.939±0.013(6)●	0.942±0.012(6)●	0.942±0.012(6)●	0.995±0.002(6)●	0.995±0.002(6)●
	DEP	0.959±0.009(2)	0.957±0.010(2)	0.959±0.009(3)	0.959±0.009(2)	0.997±0.001(3)	0.997±0.001(3)
	$\text{MMSE}_{\text{class}}$(ours)	0.957±0.009(3)	0.955±0.010(3)	0.959±0.009(2)	0.959±0.008(3)	0.997±0.001(2)	0.997±0.001(2)
	$\text{MMSE}_{\text{margin}}$(ours)	**0.960±0.008(1)**	**0.958±0.009(1)**	**0.959±0.008(1)**	**0.961±0.010(1)**	**0.998±0.001(1)**	**0.998±0.001(1)**
acoustic	SMOTE	0.904±0.007(7)●	0.903±0.007(7)●	0.890±0.008(7)●	0.926±0.006(6)●	0.939±0.004(8)●	0.936±0.004(8)●
	EasyEnsemble	0.943±0.004(5)●	0.942±0.004(5)●	0.893±0.005(6)●	0.923±0.003(7)●	0.996±0.000(4)●	0.995±0.001(4)●
	BalancedRF	0.943±0.006(4)●	0.943±0.006(4)●	0.914±0.005(4)●	0.939±0.004(5)●	0.995±0.001(5)●	0.995±0.001(5)●
	SMOTEBoost	0.893±0.010(8)●	0.889±0.011(8)●	0.910±0.009(5)●	0.945±0.005(4)●	0.995±0.001(6)●	0.994±0.001(6)●
	MDEP	0.924±0.009(6)●	0.924±0.010(6)●	0.889±0.011(8)●	0.923±0.010(8)●	0.990±0.003(7)●	0.990±0.003(7)●
	DEP	0.947±0.007(2)	0.947±0.007(2)	0.922±0.007(3)●	0.946±0.005(3)●	0.996±0.000(3)●	0.995±0.000(3)●
	$\text{MMSE}_{\text{class}}$(ours)	**0.948±0.005(1)**	**0.947±0.005(1)**	**0.932±0.005(1)**	**0.954±0.003(1)**	**0.996±0.000(1)**	**0.996±0.000(1)**
	$\text{MMSE}_{\text{margin}}$(ours)	0.947±0.003(3)	0.947±0.003(3)	0.926±0.008(2)	0.949±0.006(2)	0.996±0.000(2)	0.996±0.000(2)
miRNA	SMOTE	0.583±0.033(8)●	0.548±0.043(8)●	0.566±0.031(8)●	0.836±0.009(7)●	0.781±0.017(8)●	0.768±0.018(8)●
	EasyEnsemble	0.791±0.025(4)●	0.773±0.034(3)	0.722±0.024(4)●	0.876±0.007(5)●	0.990±0.002(3)●	0.978±0.005(2)
	BalancedRF	0.702±0.027(5)●	0.672±0.031(5)●	0.649±0.028(6)●	0.852±0.010(6)●	0.974±0.003(6)●	0.946±0.006(6)●
	SMOTEBoost	0.658±0.025(6)●	0.583±0.039(7)●	0.693±0.024(5)●	**0.901±0.007(1)**	0.988±0.002(5)●	0.967±0.005(5)●
	MDEP	0.638±0.051(7)●	0.599±0.056(6)●	0.593±0.051(7)●	0.834±0.032(8)●	0.958±0.019(7)●	0.924±0.020(7)●
	DEP	0.797±0.026(3)	0.771±0.037(4)	0.745±0.030(2)	0.888±0.009(3)●	0.989±0.002(4)●	0.977±0.005(4)●
	$\text{MMSE}_{\text{class}}$(ours)	**0.804±0.014(1)**	**0.787±0.016(1)**	**0.747±0.025(1)**	0.894±0.007(2)	**0.991±0.002(1)**	**0.979±0.004(1)**
	$\text{MMSE}_{\text{margin}}$(ours)	0.799±0.016(2)	0.781±0.022(2)	0.740±0.012(3)	0.888±0.009(4)	0.990±0.002(2)	0.977±0.004(3)

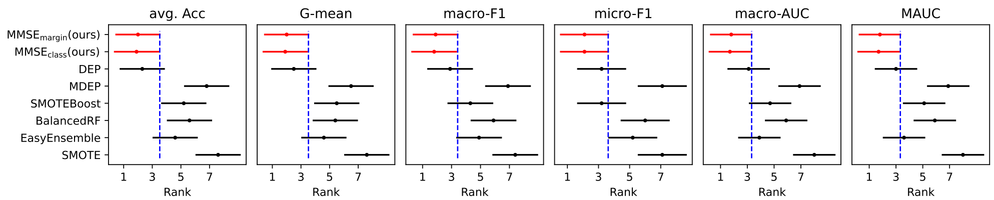


Figure 3: The result of the Friedman-Nemenyi test of the compared methods on different performance measures. The dots indicate the average ranks. The bars indicate the critical difference with the Nemenyi test at significance level 0.1, and compared methods having non-overlapped bars are significantly different.

In summary, our methods select different solutions based on the decision-maker's preferred criterion, and achieve better results than the compared methods. This quantitatively demonstrates that our method provides highly competitive choices.

5.2.2. Scenario II

In Scenario II, the decision-maker may choose any solution on the Pareto front presented to her. So in order to demonstrate the effectiveness of our approach, we need to show that we can provide decision-makers with diverse and good choices.

For ease of presentation, we select three out of six compared methods, namely DEP, EasyEnsemble, and SMOTEBoost. These three methods are better because they are not significantly inferior to our methods. We compare the solution sets generated by our methods with the single solution generated by each of the three selected methods separately. We take the *acoustic* dataset as an example and show the classifiers' validation accuracy for each class in Figure 4. The solutions in red dominate the compared classifier, which means they perform better than the compared classifier in all the classes. The solutions in orange are incomparable with the compared classifier, which means they perform better than the compared classifier in at least one class. In other words, all solutions of our methods shown in Figure 4 have their advantages. And we can observe that these solutions are also very diverse. This shows that our method can provide the decision-maker with rich choices, and these choices are no worse than the best three compared methods.

If we compare the performance of $\text{MMSE}_{\text{class}}$ and $\text{MMSE}_{\text{margin}}$ more carefully, we can observe that the performance of MMSE$_{\text{class}}$ is more widely spread in each class, which clearly reflects the waxing and waning relationship between the performance of each class. In contrast, the solution distribution of MMSE$_{\text{margin}}$ on each class has a relatively consistent trend. This is because MMSE$_{\text{margin}}$ does not directly optimize the accuracy of each class. But even so, it still provides many different trade-offs.

Figure 5 and Figure 6 show the performance of MMSE$_{\text{class}}$ and MMSE$_{\text{margin}}$ respectively on the rest datasets. We can see that both MMSE$_{\text{class}}$ and MMSE$_{\text{margin}}$ obtain diverse and highly competitive solutions on all the datasets.

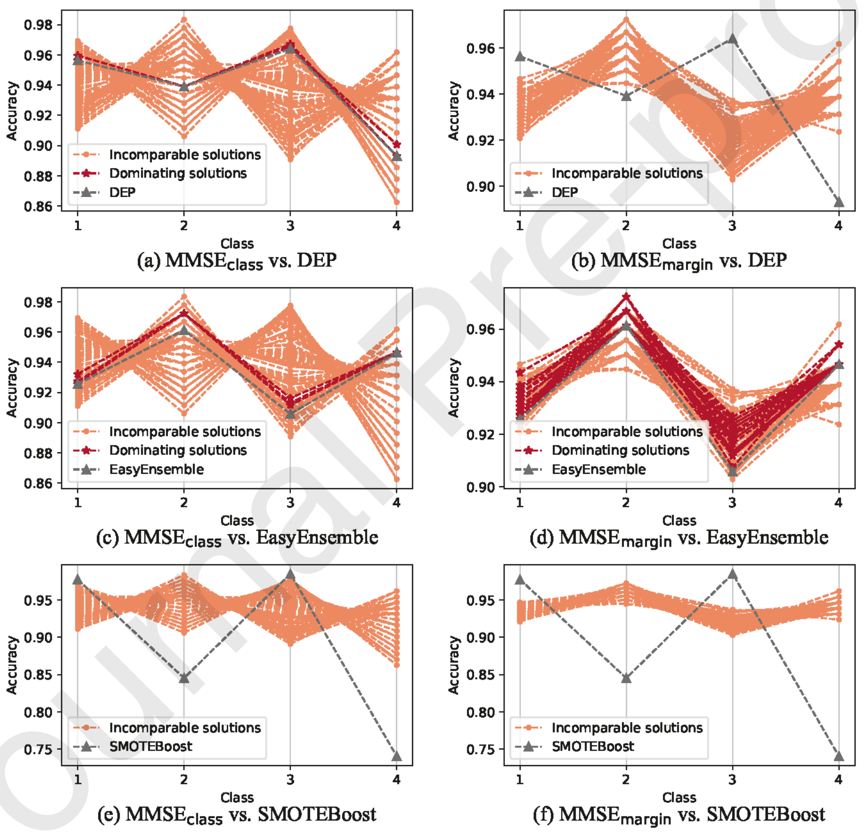


Figure 4: The solutions generated by MMSE$_{\text{class}}$ and MMSE$_{\text{margin}}$ compared with the single classifier generated by DEP, EasyEnsemble, and SMOTEBoost on *acoustic* dataset. The red solutions dominate the compared classifier, and the orange solutions are incomparable with the compared classifier.

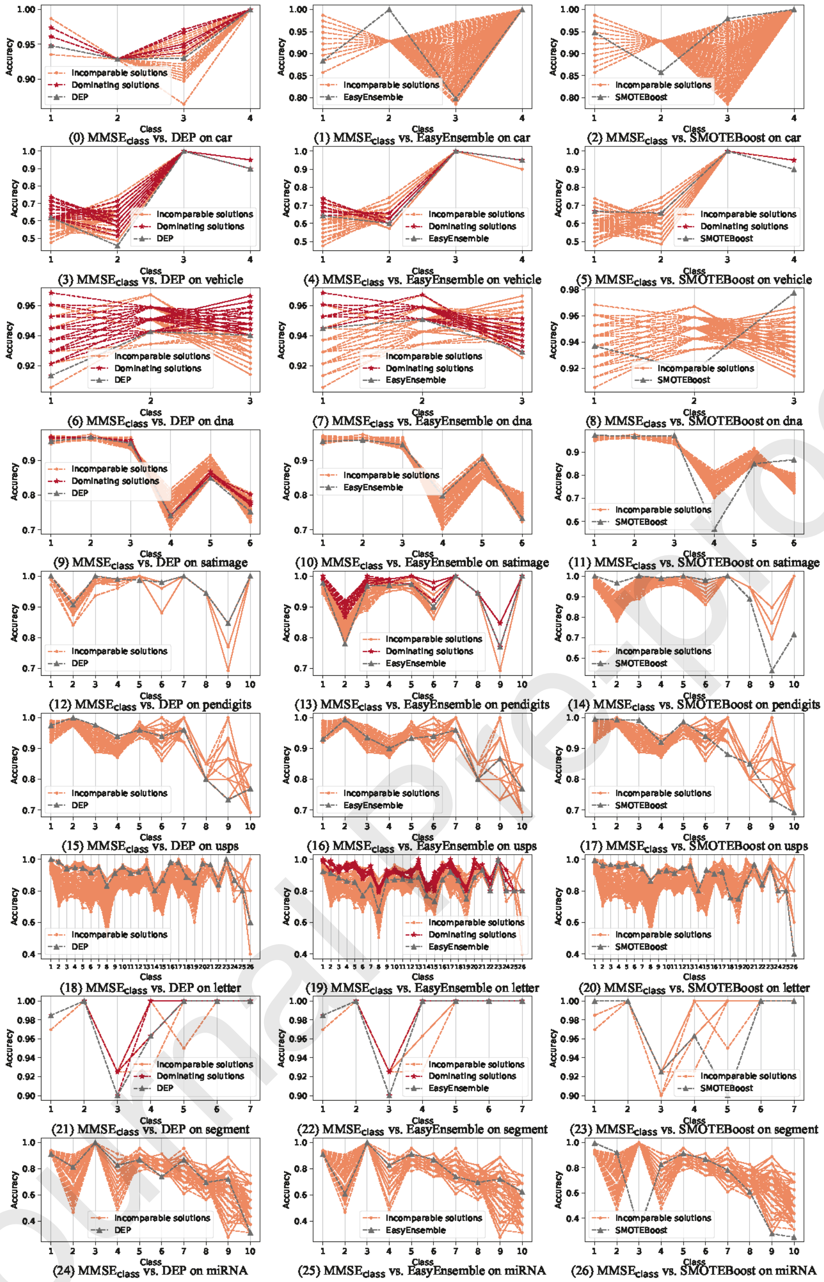


Figure 5: The solutions generated by $\text{MMSE}_{\text{class}}$ compared with the single classifier generated by DEP, EasyEnsemble, and SMOTEBoost on the other nine datasets. The red solutions dominate the compared classifier, and the orange solutions are incomparable with the compared classifier.

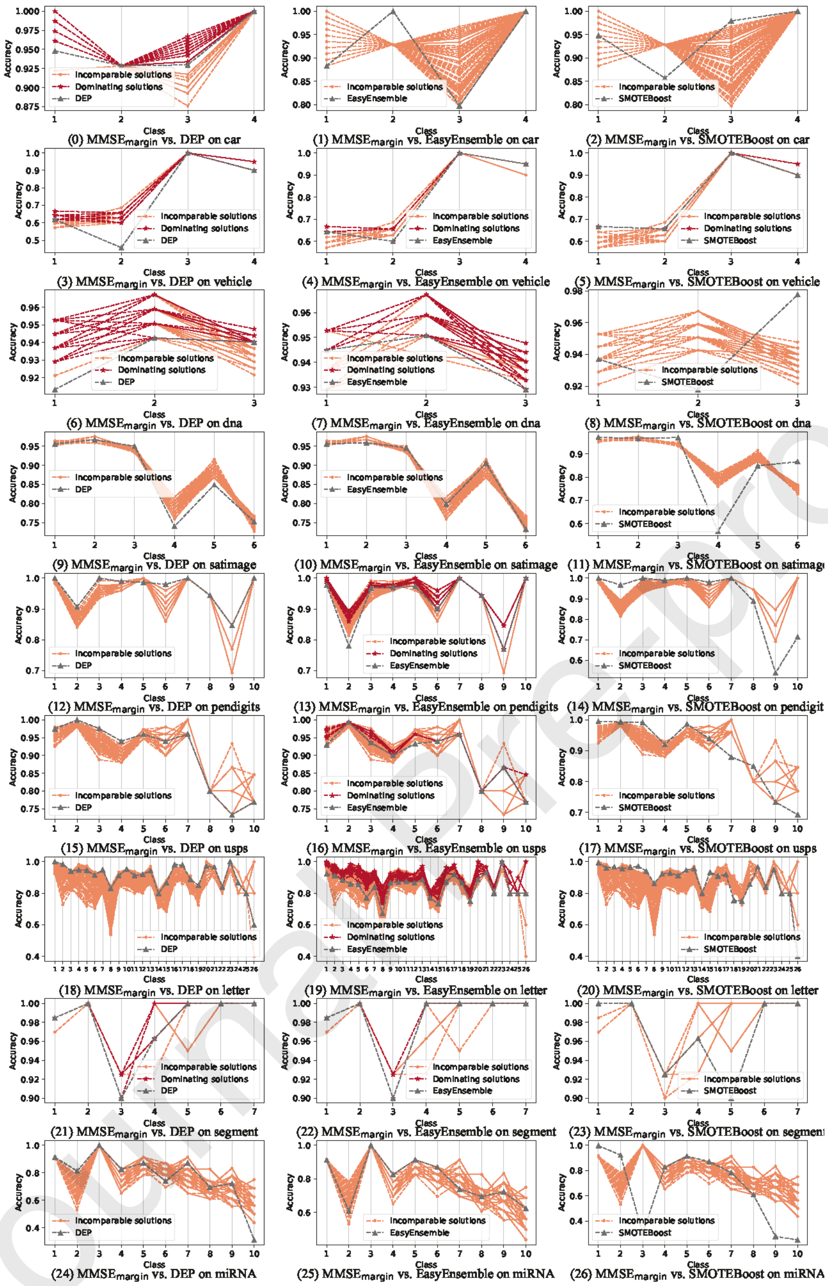


Figure 6: The solutions generated by MMSE$_{\text{margin}}$ compared with the single classifier generated by DEP, EasyEnsemble, and SMOTEBoost on the other nine datasets. The red solutions dominate the compared classifier, and the orange solutions are incomparable with the compared classifier.

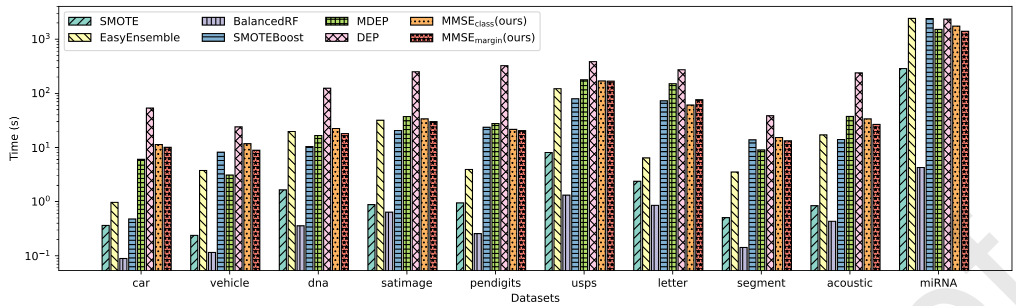


Figure 7: Running time comparison.

5.3. *Running time comparison*

In this subsection, we compare the running time of different methods. The running time of our methods $\mathrm{MMSE_{class}}$ and $\mathrm{MMSE_{margin}}$ include training of base learners, multi-objective evolutionary optimization, and the evaluation of the obtained solution set on all the performance measures. Because our methods need to show the decision-maker the performance of all the obtained solutions in all the classes and different evaluation criteria, it is fair to include this part of the time. The running time of the compared methods includes the hyper-parameter tuning and the evaluation of the obtained single model on all the performance measures. As we can observe in Figure 7, the running time of $\mathrm{MMSE_{class}}$ and $\mathrm{MMSE_{margin}}$ is comparable with EasyEnsemble and SMOTEBoost, the running time of MDEP is roughly the same, while DEP has even longer running time. That is to say, our methods successfully obtain diverse highly competitive solutions efficiently.

5.4. *Effectiveness of optimizing margins*

$\mathrm{MMSE_{margin}}$ is a novel design of objective modeling proposed to reduce the number of objectives. In Section 4, we proved that optimizing label-wise margin can optimize *Average Accuracy*, *G-mean*, *macro-F1*, *micro-F1*, and optimizing the instance-wise margin can optimize *macro-AUC* and *MAUC*. Therefore, in this subsection, we experimentally verify it. We choose the *letter* dataset, which has a large number of classes that can best demonstrate the advantages of $\mathrm{MMSE_{margin}}$. We record the objective values and performance measures of all solutions generated during the multi-objective optimization process. Figure 8 verifies the positive correlation between optimizing the label-wise margin and *Average Accuracy*, *G-mean*, *macro-F1*,

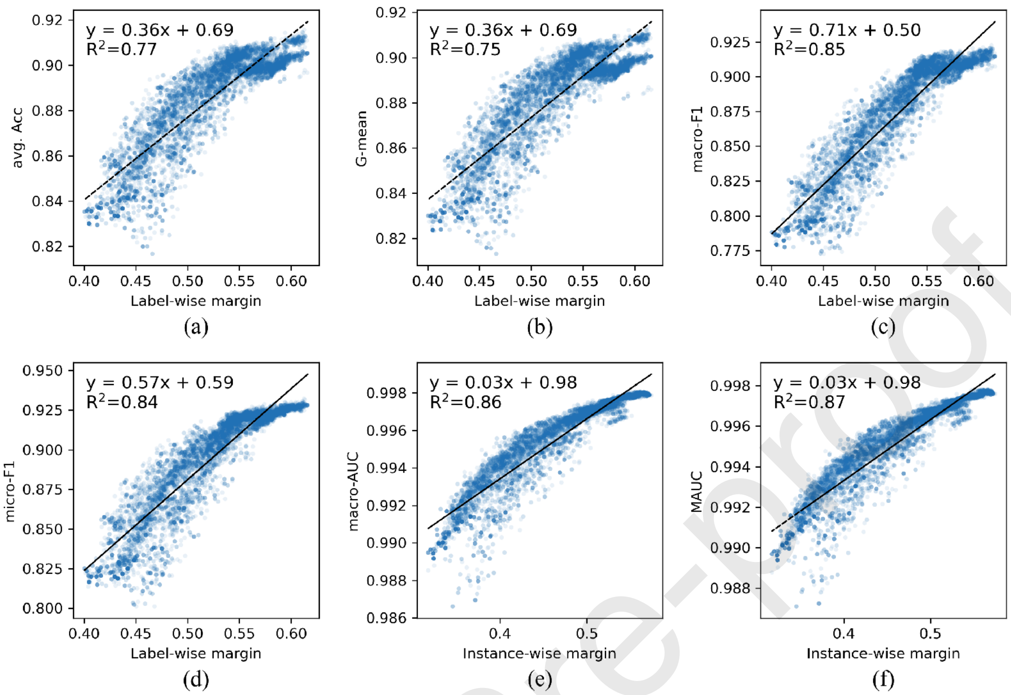


Figure 8: The relationship between the optimization objective and the performance measure that can be optimized in theory. The points are all the solutions generated during the multi-objective evolutionary optimization of applying $\text{MMSE}_{\text{margin}}$ on the *letter* dataset.

micro-F1, and the positive correlation between optimizing the instance-wise margin and *macro-AUC* and *MAUC* through two-dimensional scatter plots and the linear fit lines. The slopes of the fitted lines vary greatly because the solutions have different ranges of values on different performance measures, but all slopes are positive, indicating a positive correlation. The key point to note is that the R^2 values in each subplot are good, as an R^2 value close to 1 indicates a good fit.

6. Conclusion

In this paper, we revisit the multi-class imbalance problem from the perspective of multi-objective optimization. Instead of using a predefined rebalancing strategy and generating a single model, we propose the MMSE framework to generate a set of ensembles with the best possible trade-offs between classes. In real-world applications where it is difficult to choose between different trade-off strategies *a priori*, the decision-maker will be in a better position to make the final choice if the optimal trade-offs are given. Specifically, we propose $\mathrm{MMSE}_{\mathrm{class}}$ and $\mathrm{MMSE}_{\mathrm{margin}}$. The latter enjoys a theoretical guarantee. And experimental results verify that both $\mathrm{MMSE}_{\mathrm{class}}$ and $\mathrm{MMSE}_{\mathrm{margin}}$ can obtain diverse and highly competitive solutions within an acceptable running time.

Currently, we are dealing with class imbalance problems where there is a relative lack of samples in the small classes. An interesting future work is to explore how to use the small class information more effectively when the small class samples are extremely scarce. Another interesting direction for future work is to design specific optimization algorithms for this combinatorial multi-objective optimization problem.

Acknowledgments

This work was supported by the National Science Foundation of China (62276124,62306104). The authors would like to thank Professor Bin Liang and student Wei Wang for providing an acoustic model that generates the acoustic data set.

References

[1] Ahmed, R., Mir, F., Banerjee, S., 2017. A review on energy harvesting approaches for renewable energies from ambient vibrations and acoustic waves using piezoelectricity. Smart Materials and Structures 26, 085031.

[2] Buchbinder, N., Feldman, M., Naor, J., Schwartz, R., 2014. Submodular maximization with cardinality constraints, in: Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1433–1452.

[3] Chang, C.C., Lin, C.J., 2011. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 1–27.

[4] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. Smote: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research 16, 321–357.

[5] Chawla, N.V., Lazarevic, A., Hall, L.O., Bowyer, K.W., 2003. Smoteboost: Improving prediction of the minority class in boosting, in: Proceedings of the 7th European Conference on Principles of Data Mining and Knowledge Discovery, pp. 107–119.

[6] Chen, C., Liaw, A., Breiman, L., et al., 2004. Using random forest to learn imbalanced data. University of California, Berkeley 110, 24.

[7] Das, A., Kempe, D., 2011. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection, in: Proceedings of the 28th International Conference on Machine Learning, pp. 1057–1064.

[8] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6, 182–197.

[9] Du, H., Zhang, Y., Zhang, L., Chen, Y.C., 2023. Selective ensemble learning algorithm for imbalanced dataset. Computer Science and Information Systems , 23–23.

[10] Estabrooks, A., Jo, T., Japkowicz, N., 2004. A multiple resampling method for learning from imbalanced data sets. Computational Intelligence 20, 18–36.

[11] Fernandes, E.R., de Carvalho, A.C., Yao, X., 2019. Ensemble of classifiers based on multiobjective genetic sampling for imbalanced data. IEEE Transactions on Knowledge and Data Engineering 32, 1104–1115.

[12] Friedrich, T., Göbel, A., Neumann, F., Quinzan, F., Rothenberger, R., 2019. Greedy maximization of functions with bounded curvature under partition matroid constraints, in: Proceedings of the 33rd AAAI Conference on Artificial Intelligence, pp. 2272–2279.

[13] Guo, Y., Zhang, C., 2021. Recent advances in large margin learning. IEEE Transactions on Pattern Analysis and Machine Intelligence 44, 7167–7174.

[14] Hand, D.J., Till, R.J., 2001. A simple generalisation of the area under the roc curve for multiple class classification problems. Machine Learning 45, 171–186.

[15] Hastie, T., Rosset, S., Zhu, J., Zou, H., 2009. Multi-class adaboost. Statistics and its Interface 2, 349–360.

[16] He, H., Bai, Y., Garcia, E.A., Li, S., 2008. Adasyn: Adaptive synthetic sampling approach for imbalanced learning, in: Proceedings of International Joint Conference on Neural Networks, pp. 1322–1328.

[17] He, H., Garcia, E.A., 2009. Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering 21, 1263–1284.

[18] He, H., Ma, Y., 2013. Imbalanced Learning: Foundations, Algorithms, and Applications. John Wiley & Sons.

[19] He, Y.X., Wu, Y.C., Qian, C., Zhou, Z.H., 2024. Margin distribution and structural diversity guided ensemble pruning. Machine Learning doi:10.1007/s10994-023-06429-3.

[20] Inselberg, A., Dimsdale, B., 1990. Parallel coordinates: a tool for visualizing multi-dimensional geometry, in: Proceedings of the 1st IEEE conference on visualizatio, pp. 361–378.

[21] Krause, A., Singh, A.P., Guestrin, C., 2008. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. Journal of Machine Learning Research 9, 235–284.

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

[23] Liang, J., Zhang, Y., Chen, K., Qu, B., Yu, K., Yue, C., Suganthan, P.N., 2024. An evolutionary multiobjective method based on dominance and decomposition for feature selection in classification. Science China Information Sciences 67, 120101.

[24] Liu, X.Y., Li, Q.Q., Zhou, Z.H., 2013. Learning imbalanced multi-class data with optimal dichotomy weights, in: Proceedings of the 13th IEEE International Conference on Data Mining, pp. 478–487.

[25] Liu, X.Y., Wu, J., Zhou, Z.H., 2009. Exploratory undersampling for class-imbalance learning. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 2, 539–550.

[26] Liu, X.Y., Zhou, Z.H., 2006. The influence of class imbalance on cost-sensitive learning: An empirical study, in: Proceedings of the 6th IEEE International Conference on Data Mining, pp. 970–974.

[27] Lyu, S.H., Yang, L., Zhou, Z.H., 2019. A refined margin distribution analysis for forest representation learning, in: Advances in Neural Information Processing Systems 32, pp. 5531–5541.

[28] Pillai, M.A., Deenadayalan, E., 2014. A review of acoustic energy harvesting. International Journal of Precision Engineering and Manufacturing 15, 949–965.

[29] Prajapati, A., Parashar, A., Rathee, A., 2023. Multi-dimensional information-driven many-objective software remodularization approach. Frontiers of Computer Science 17, 173209.

[30] Qian, C., Shi, J.C., Yu, Y., Tang, K., Zhou, Z.H., 2017. Subset selection under noise, in: Advances in Neural Information Processing Systems 30, pp. 3563–3573.

[31] Qian, C., Yu, Y., Zhou, Z.H., 2015. Subset selection by pareto optimization, in: Advances in Neural Information Processing Systems 28, pp. 1765–1773.

[32] Roshan, S.E., Asadi, S., 2020. Improvement of bagging performance for classification of imbalanced datasets using evolutionary multi-objective optimization. Engineering Applications of Artificial Intelligence 87, 103319.

[33] Seiffert, C., Khoshgoftaar, T.M., Van Hulse, J., Napolitano, A., 2009. RUSBoost: A hybrid approach to alleviating class imbalance. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 40, 185–197.

[34] Shen, C., Xie, Y., Li, J., Cummer, S.A., Jing, Y., 2018. Acoustic metacages for sound shielding with steady air flow. Journal of Applied Physics 123, 124501.

[35] Wang, S., Yao, X., 2012. Multiclass imbalance problems: Analysis and potential solutions. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 42, 1119–1130.

[36] Wu, X.Z., Zhou, Z.H., 2017. A unified view of multi-label performance measures, in: Proceedings of the 34th International Conference on Machine Learning, pp. 3780–3788.

[37] Wu, Y.C., He, Y.X., Qian, C., Zhou, Z.H., 2022. Multi-objective evolutionary ensemble pruning guided by margin distribution, in: Proceedings of the 17th International Conference on Parallel Problem Solving from Nature, pp. 427–441.

[38] Xu, Y., Yu, Z., Chen, C.P., 2024. Classifier ensemble based on multiview optimization for high-dimensional imbalanced data classification. IEEE Transactions on Neural Networks and Learning Systems 31, 870–883.

[39] Xue, Y., Cai, X., Neri, F., 2022. A multi-objective evolutionary algorithm with interval based initialization and self-adaptive crossover operator for large-scale feature selection in classification. Applied Soft Computing 127, 109420.

[40] Xue, Y., Tang, Y., Xu, X., Liang, J., Neri, F., 2021. Multi-objective feature selection with missing data in classification. IEEE Transactions on Emerging Topics in Computational Intelligence 6, 355–364.

[41] Yang, K., Yu, Z., Chen, C.P., Cao, W., Wong, H.S., You, J., Han, G., 2021. Progressive hybrid classifier ensemble for imbalanced data. IEEE Transactions on Systems, Man, and Cybernetics: Systems 52, 2464–2478.

[42] Yang, K., Yu, Z., Chen, C.P., Cao, W., You, J., Wong, H.S., 2022. Incremental weighted ensemble broad learning system for imbalanced data. IEEE Transactions on Knowledge and Data Engineering 34, 5809–5824.

[43] Yang, K., Yu, Z., Wen, X., Cao, W., Chen, C.P., Wong, H.S., You, J., 2020. Hybrid classifier ensemble for imbalanced data. IEEE transactions on neural networks and learning systems 31, 1387–1400.

[44] Yang, P., Zhang, L., Liu, H., Li, G., 2024. Reducing idleness in financial cloud services via multi-objective evolutionary reinforcement learning based load balancer. Science China Information Sciences 67, 120102.

[45] Yokoi, A., Matsuzaki, J., Yamamoto, Y., Yoneoka, Y., Takahashi, K., Shimizu, H., Uehara, T., Ishikawa, M., Ikeda, S.i., Sonoda, T., et al., 2018. Integrated extracellular microRNA profiling for ovarian cancer screening. Nature Communications 9, 1–10.

[46] Zhang, C., Xue, Y., Neri, F., Cai, X., Slowik, A., 2024. Multi-objective self-adaptive particle swarm optimization for large-scale feature selection in classification. International journal of neural systems , 2450014–2450014.

[47] Zhen, L., Li, M., Peng, D., Yao, X., 2020. Objective reduction for visualising many-objective solution sets. Information Sciences 512, 278–294.

[48] Zhou, Z.H., 2012. Ensemble Methods: Foundations and Algorithms. Chapman & Hall/CRC, Boca Raton, FL.

[49] Zhou, Z.H., 2022. Open-environment machine learning. National Science Review 9, nwac123. doi:10.1093/nsr/nwac123.

[50] Zhou, Z.H., Yu, Y., Qian, C., 2019. Evolutionary Learning: Advances in Theories and Algorithms. Springer, Singapore.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: