Soft Computing (2021) 25:2085–2101
https://doi.org/10.1007/s00500-020-05279-8

METHODOLOGIES AND APPLICATION

An improved deep forest for alleviating the data imbalance problem

Jie Gao[1] · Kunhong Liu[1] · Beizhan Wang[1] · Dong Wang[2] · Qingqi Hong[1]

Published online: 28 August 2020

Abstract
Most deep learning methods have inherent defects and are rarely applied in the classification task of small-sized imbalanced datasets. On the one hand, data imbalance causes the classification results of the model to be biased toward the majority class. On the other hand, limited training data results in over-fitting. Deep forest (DF) is an interesting deep learning model that can perfectly work on small-sized datasets, and its performance is highly competitive with deep neural networks. In the present study, a variant of the DF called the imbalanced deep forest (IMDF) is proposed to effectively improve the classification performance of the minority class. It aims to explore the application of deep learning on small-sized imbalanced datasets. The IMDF is the cascade of multiple layers, where each layer is the ensemble of multiple units. The main idea behind the proposed method is to enable each unit of the IMDF to handle imbalanced data so that the classification results of the entire IMDF are biased toward minority class. Performed experiments demonstrate the effectiveness of the proposed method.

Keywords Deep forest · AdaBoost · SMOTE · Imbalanced data

1 Introduction

Rare events are normally defined as unusual patterns and abnormal behaviors that occur at a low frequency of less than 5%, or even less than 0.1% (Maher Maalouf 2011). Moreover, rare events are characterized by under-represented instances and high misclassification costs. Studies show that it is a great challenge to detect rare events in real applications, such as network intrusion detection and cancer detection. More specifically, the number of intrusions shares a very small fraction of the total network traffic. However, misclassifying a cancerous patient as a non-cancerous one postpones the timely treatment and may result in a heavy cost (Chawla et al. 2003). Therefore, the nature of these applications requires a high detection rate for rare events.

In the field of data mining, identifying rare events is essentially a binary classification problem of imbalanced data (Branco et al. 2016). It is normally assumed that the class with the smallest size forms the minority class (or the positive class), while other classes form the majority class (or the negative class). Imbalanced rate (IR) is an indicator, which is normally used to evaluate the skew level of the dataset. It is defined as the ratio of the number of the majority class to that of the minority class (Loyola-González et al. 2017). Most conventional machine learning algorithms, such as the decision tree (Nie et al. 2011) and Naive Bayes (Jiang et al. 2016) are designed based on the assumption that the training data is balanced and evenly distributed in each class. Studies show that in the case of imbalanced data, these algorithms often provide sub-optimal classification results, which are biased toward the majority class (Seiffert et al. 2010). In other words, conventional machine learning algorithms are not accurate enough for identifying minority instances. It is worth noting that any multi-classification problem can be transformed into the combination of multiple binary classifications problems through class decomposition techniques.

Communicated by V. Loia.

✉ Kunhong Liu
lkhqz@xmu.edu.cn

✉ Beizhan Wang
wangbz@xmu.edu.cn

[1] School of Informatics, Xiamen University, Xiamen 361005, People's Republic of China

[2] State Grid Fujian Electric Power Company, Fuzhou 350003, People's Republic of China

Reviewing the literature indicates that over the past two decades, many improved non-deep machine learning techniques have been applied to address imbalanced data, and more than 600 investigations were proposed in this regard (Branco et al. 2016; Haixiang et al. 2017). However, the deep learning method has only been applied in a few investigations. Johnson and Khoshgoftaar (2019) reviewed existing deep learning techniques for addressing the imbalanced data. They found only 15 articles on this issue, and mainly focused on the large-scale computer vision tasks.

Anand et al. (1993) initially explored the effects of imbalanced data on the back-propagation algorithm in the neural network. They showed that in the case of imbalanced data, the length of gradient components of the minority class is much smaller than that of the majority class. In other words, the majority class essentially dominates the network gradient and is responsible for updating the model weight. Therefore, classification errors of the majority class quickly reduce during early iterations, while classification errors of minority class often increase so that the network gets stuck in a slow convergence mode. They concluded that applying the neural network on imbalanced data leads to biased classification results toward the majority class. Moreover, one of the main challenges of deep neural networks (DNNs) as well as other neural networks is that the limited training data results in over-fitting (Utkin and Ryabinin 2018). Studies show that in the case of small-sized imbalanced datasets, the number of minority instances is even rarer. These situations increase the challenges of DNNs to identify minority instances.

Recently, Zhou and Feng (2017) proposed an interesting method, called the deep forest (DF) method, which can be regarded as an alternative to DNNs. The main part of the DF method is the multi-layer cascade structure, which is similar to that of DNNs. Each layer in the DF method contains a set of random forests (RFs). In other words, each layer in the DF method consists of different decision tree ensembles. On the other hand, each RF can be regarded as a unit of DF, just like a neuron in DNNs. In comparison with DNNs that require great efforts for tuning hyper-parameters and large-scale training data, DF has superior characteristics, including simple training, few hyper-parameters, and reasonable performance on the small-sized datasets. Zhou and Feng (2019) pointed out that the performance of the DF method is highly competitive with DNNs. It is worth noting that the DF method can be implemented in diverse applications by endowing the corresponding function to its units (Utkin 2019; Utkin and Ryabinin 2018; Utkin et al. 2019).

In the present study, it is intended to propose the imbalanced deep forest (IMDF) method, which can be considered as a variant of the DF method to effectively handle the data imbalance problem. The main idea behind the proposed method is to dedicate each unit of the IMDF to handle imbalanced data so that the classification results of the entire IMDF are biased toward the minority class. It is expected to explore the application of the deep learning method on small-sized imbalanced datasets. Three main contributions of the present study are as follows:

1. A new unit is designed for the IMDF, which can be considered as an improved AdaBoost algorithm (Freund and Schapire 1996). Each unit focuses on the misclassified minority instances in each iteration and then strengthens the learning of these instances in the next iteration by implementing the synthetic minority over-sampling technique (SMOTE). This iterative learning mechanism makes the unit algorithm more effective for identifying minority instances.
2. There are two initialization strategies for the unit, both of them can enhance the discriminative ability for the minority class in diverse manners. Since each layer of the IMDF consists of multiple units, different initialization strategies are employed to produce units with high diversity to promote the generalization ability of the ensemble.
3. The IMDF is implemented by the layer-by-layer data processing, in-model feature transformation, and the greedy training mechanism.

The rest of the present study is organized as follows: Related background knowledge is presented in Sect. 2. Then, the construction of the IMDF is described in Sect. 3. The experimental results and the corresponding discussions are presented in Sect. 4. Finally, concluding remarks are provided in Sect. 5.

2 Background

2.1 Synthetic minority over-sampling technique

SMOTE is a data-level approach and can be regarded as a benchmark for imbalanced learning (Chawla et al. 2002). The SMOTE can be applied to generate new synthetic minority instances to rebalance the imbalanced training set. Suppose that a minority class instance X_i is selected as a basis to create new synthetic instances (see Fig. 1). Based on a distance metric, several nearest neighbors of the same class (points X_{i1} to X_{i5}) are chosen from the original training set. Then, a randomized interpolation is carried out to obtain new synthetic instances R_1 to R_5, respectively. The pseudo-code of the SMOTE is as the form below (Fernandez et al. 2018):

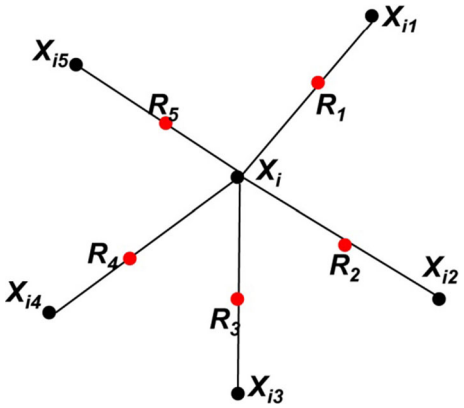


Fig. 1 Creating synthetic instances through SMOTE

1. Calculate the difference between X_i and one of its K-nearest neighbors (by default, $K = 5$).
2. The difference is multiplied by a random number between 0 and 1.
3. Add this difference to X_i to generate a new synthetic instance.

How many minority instances should be generated by the SMOTE to ensure the reasonable performance of the learned classifier? In the imbalanced learning, there is no universally accepted optimal class distribution, and a balanced distribution 50:50 is usually regarded as the near-optimal ratio. However, studies show that a ratio of 35:65 between the minority class and the majority class may result in better classification performance when the minority instances are rare (Seiffert et al. 2010).

2.2 Boosting algorithm

Boosting is an ensemble technique that can iteratively train a set of weak hypotheses. Reviewing the literature indicates that the most common boosting algorithm is the AdaBoost algorithm (Freund and Schapire 1996), which can be considered as an advanced sampling technique so that it can be carried out by reweighting or resampling (Seiffert et al. 2010). When the boosting technique is carried out through the reweighting, instance weights are utilized to calculate the weight loss of the base classifier. On the other hand, when the boosting technique is realized by the resampling, instance weights are used to resample the original training dataset to generate diverse training datasets. In each iteration, weights of misclassified instances increase, while weights of correct classified instances decrease in the next iteration. When all iterative calculations are completed, all learned weak hypotheses participate in the weighted voting to classify unknown instances. This technique is effective to deal with imbalanced data because minority instances are often misclassified so that higher weights are required in the subsequent iterations.

However, the same weights are assigned in the boosting technique for all misclassified instances. In other words, the boosting handles misclassified majority and minority instances in the same way. In the case of imbalanced data, the sampling distribution is biased toward the majority class in subsequent iterations. Although boosting reduces the bias in the final ensemble, it is not effective enough for datasets with skew class distribution (Chawla et al. 2003).

In order to resolve this shortcoming, the SMOTEBoost algorithm (Chawla et al. 2003) has been proposed that combines the SMOTE with the boosting mechanism. Studies show that this algorithm can alleviate the data imbalance problem in the learning process of base classifiers. Furthermore, the RUSBoost algorithm (Seiffert et al. 2010) adopts random under-sampling in the majority class during each iteration. It should be indicated that both SMOTEBoost and RUSBoost algorithms are representatives of the iterative ensemble by combining resampling methods with a boosting mechanism. Therefore, they are selected in the present study to evaluate obtained results from the experiment.

2.3 Evaluation metrics

More than 40 evaluation metrics have been proposed so far to handle data imbalance issues. Branco et al. (2016) comprehensively reviewed these metrics. In this section, it is intended to briefly explain some commonly used metrics. The confusion matrix in Table 1 is applied to assess the performance of machine learning algorithms for imbalanced datasets. In the case of binary classification, labels "P" and "N" correspond to the class of interest (minority class or positive class), and a combination of all the other classes (majority class or negative class), respectively. Moreover, TP, FN, FP, and TN denote four possible classification results.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{TN} + \text{FP}} \tag{1}$$

$$\text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})} \tag{2}$$

$$\text{Recall} = \text{TP rate} = \frac{\text{TP}}{(\text{TP} + \text{FN})} \tag{3}$$

$$F\text{ - value} = \frac{\left(1 + \beta^2\right) * \text{Recall} * \text{Precision}}{\beta^2 * \text{Recall} + \text{Precision}} \tag{4}$$

$$\text{FP rate} = \frac{\text{FP}}{(\text{TN} + \text{FP})} \tag{5}$$

Equation 1 indicates that the imbalanced learning guided by the global performance metric such as the accuracy

Table 1 Four classification results of the confusion matrix

	Predicted label "P"	Predicted label "N"
Actual label "P" (minority class)	True positive (TP)	False negatives (FN)
Actual label "N" (majority class)	False positives (FP)	True negatives (TN)

induces a bias toward the majority class (Loyola-González et al. 2016). Suppose that shares of the majority and minority classes in an imbalance dataset are 95% and 5%, respectively. In this case, a trivial classifier that labels all testing instances as the majority class can achieve high accuracy. However, the main goal of imbalanced learning is to accurately identify the minority class, so such accuracy cannot effectively reflect the predictive performance of the classifiers for the minority class.

Equation 2 reveals that the precision rate is sensitive to the data imbalance because it takes false positives (FP) into account. However, the precision alone cannot accurately express the classification status, because it provides no insight into false negatives (FN). On the other hand, Eq. 3 indicates that the recall rate is not affected by the data imbalance because it only depends on the minority class. However, the recall rate does not consider FP. In reality, the precision and recall rates often conflict with each other. To resolve this problem, the F-value (Eq. 4) is defined as a trade-off between the precision rate and the recall rate, where β is corresponding to the relative importance between the precision and recall. In the present study, β is set to 1.

Receiver operating characteristic (ROC) curves can also be used to evaluate the performance of classifiers for imbalanced datasets. It is a two-dimensional graph in which the true positives (TP) and FP rates are plotted on the y- and the x-axes, respectively. When comparing multiple algorithms by ROC curves, it is a challenge to distinguish the best algorithm, unless one curve outperforms the others on the entire space. The area under a ROC curve (AUC) provides a single metric for evaluating algorithms on the average.

3 The proposed method

3.1 The unit construction

The proposed IMDF is a variant of DF, where each unit is an improved AdaBoost algorithm. The ensemble of multiple units forms each layer in the IMDF, while the cascade of multiple layers forms the overall IMDF. In this section, the construction of one unit is initially described. The corresponding pseudo-code and the flowchart are presented in Algorithm 1 and Fig. 2, respectively.

Suppose that the size of the original training set (S) is m. Moreover, X and Y are the instance space and the label space, respectively. Then $x_i \in X$ and $y_i \in Y$ are an instance and the corresponding label, respectively. It is worth noting that $y_i = +1$ corresponds to the minority class, while $y_i = -1$ corresponds to the majority class. h is the learned weak hypothesis (base classifier), and T is the number of iterations, which can also be regarded as the number of base classifiers. D and M denote arrays that store the weights of all instances and misclassified minority instances, respectively. D and M are continuously updated in each iteration.

In the initialization step, the sizes of the minority class and the majority class are assumed a and b, respectively, where $a \leq b$ and $a + b = m$. There are two strategies to assign initial instance weights. Case 1 is a conventional approach, where all instances are given the uniform weight ($\frac{1}{m}$). In this strategy, it is assumed that all instances have the same probability to appear in the training set after the first resampling. In the case of imbalanced data, the training set after the first resampling is mainly composed of majority instances. In case 2, the weight of each minority instance is set to $\frac{1}{2a}$, while the weight of each majority instance is set to $\frac{1}{2b}$. Since $a + b = m$ and a is much smaller than b, then $\frac{1}{2a}$ is much larger than $\frac{1}{2b}$. Comparing weights in both strategies indicates that case 2 significantly increases the weight of minority instances during the initialization. In other words, case 2 intends to add more minority instances to the training set after the first resampling. For each unit, only one strategy is selected as the initialization method of instance weights. The units starting with different initialization strategies are regarded as diverse units.

Algorithm 1 iteratively trains T base classifiers, where $t \in [1, T]$. When $t = 1$, all steps except step 3 are performed. Moreover, when $t > 1$, all steps except step 2 are performed. The detailed steps are described as follows:

Step 1 The sampling is performed with replacement according to the instance weights (D_t) to obtain the training set S_t. The size of S_t is the same as that of the original training set S.

Step 2 Training the base classifier h_t with the training set S_t. The decision tree is utilized as the prototype of the base classifier.

Step 3 Take the misclassified minority instances of the previous iteration (M_{t-1}) as the basis, and then generate n synthetic minority instances (S_n) through SMOTE. The

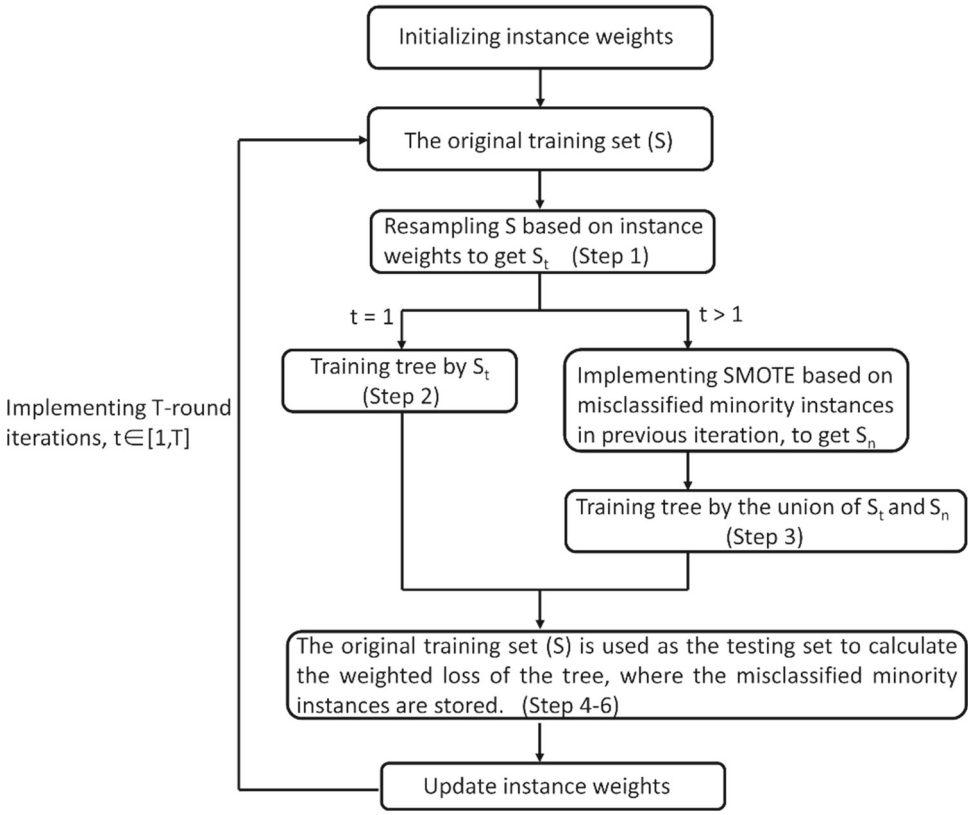


Fig. 2 Flowchart of the unit algorithm

combination of S_t and S_n sets is used to train the base classifier.

Step 4 The original training set S is used as the testing set to verify the performance of h_t.

Step 5 The misclassified minority instances are stored in M_t, and then they are used as the basis of SMOTE to create new synthetic instances in the next iteration.

Step 6 Calculate the sum of weights of misclassified minority instances to obtain weighted loss of h_t.

Step 7 Update the weight of all instances to get $D_{(t+1)}$.

After training T base classifiers, when predicting an unknown instance, the unit algorithm outputs the symbol of the weighted sum of the classification result of all base classifiers. In this case, " + " or "−" determines whether an unknown instance belongs to the minority class or the majority class.

In the design process of the unit algorithm, three points should be emphasized. First, misclassified minority instances in each iteration can be regarded as the boundary instances in the classification process so that they need more attention. SMOTE is used to generate a set of synthetic instances based on these boundary instances in the next iteration, where the amount of SMOTE will be discussed in the experiment section. Then new synthetic instances participate in the training process of the classifier. The main idea behind this participation is to let base classifiers learn a broader classification boundary for the minority class. In other words, base classifiers can be more sensitive to identify minority instances. Moreover, the randomness of resampling and SMOTE guarantees the diversity between each training set so that the learned base classifiers are diverse. AdaBoost is a representative algorithm of ensemble learning, reasonable diversity among base classifiers is an essential condition to ensure the performance of AdaBoost. Furthermore, different initialization strategies are applied to further increase diversity among different units. More specifically, case 1 moderately improves TP and minimizes the loss of TN, while case 2 maximizes the TP without considering the degradation of TN. Differences originating from different initialization strategies will be analyzed in the next section.

A unit algorithm not only can be directly used to predict the labels of unknown instances but also can be embedded in the IMDF framework. A unit in the IMDF can be regarded as a neuron in DNNs. To this end, the output form of Algorithm 1 should be modified. The output can be expressed in the form of class probability vector in the form below:

$$H(x) = \left[p^{(-1)}, p^{(+1)}\right] \tag{6}$$

where p is the weighted sum of base classifiers with the same predicted result and is defined as the following:

$$p^y = \sum_{t=1}^{T} I(h_t(x) = y)\alpha_t \quad (7)$$

where superscript y is a label, $y \in \{-1, +1\}$. $I(*)$ is an indicator function, here $I(\text{true}) = 1$ and $I(\text{false}) = 0$; α_t is the weight update parameter in step 7 of Algorithm 1 and can also be regarded as the weight of the corresponding base classifier.

Algorithm 1 Unit algorithm for processing the imbalanced data.

Given: Training dataset $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, $x_i \in X$, with labels $y_i \in Y = \{+1, -1\}$, where $y_i = +1$ corresponds to minority class, $y_i = -1$ corresponds to majority class.

h: Learned weak hypothesis

T: Number of iterations

D: Weight distribution of all examples

M: A set of misclassified minority class samples

Initialization

a and b are the size of minority class and majority class, where a⩽b and a+b=m.

Case 1: $D_1 = \{d_1, \ldots, d_m\}$, $i \in [1, m]$, $d_i = \frac{1}{m}$ for all i.

Case 2: $D_1 = \{d_1, \ldots, d_m\}$, $i \in [1, m]$, and $\begin{cases} if\ y_i = +1,\ d_i = \frac{1}{2a} \\ if\ y_i = -1, d_i = \frac{1}{2b} \end{cases}$

Algorithm process

FOR t =1,2, …, T DO

Step 1 Create temporary training dataset S_t with distribution D_t using random sampling with replacement, $|S_t| = |S|$. //time complexity: O(m)

Step 2 If t=1, train a weak learner using S_t, then go to step 4. //time complexity: O(tree)

Step 3 If t>1, creating n synthetic minority instances (S_n) from M_{t-1} by SMOTE, then train a weak learner using the union of S_t and S_n. //time complexity: O(n)+O(tree)

Step 4 Calculate weak hypothesis h_t: $X \times Y \rightarrow \{+1, -1\}$ //time complexity: O(tree)

Step 5 Set $M_t = \{ (x_i, y_i) \mid i \in [1,m], y_i = +1 \text{ and } h_t(x_i) \neq y_i \}$ //time complexity: O(1)

Step 6 Calculate the weighted loss of hypothesis h_t:

$$e_t = \sum_{x_i \in S:\, h_t(x_i) \neq y_i} D_t(i)$$ //time complexity: O(m)

Step 7 Calculate the weight update parameters:

$$\alpha_t = \frac{1}{2} log \frac{1-e_t}{e_t}$$ //time complexity: O(1)

Update D_t:

$$D_{t+1}(i) = \left(\frac{D_t(i)}{Z_t}\right) \exp\left(-\alpha_t y_i h_t(x_i)\right)$$ //time complexity: O(m)

where Z_t is a normalization factor

$$Z_t = \sum_{i=1}^{m} D_t(i) \exp\left(-\alpha_t y_i h_t(x_i)\right)$$

Output the final hypothesis:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

3.2 The IMDF structure

Before introducing the IMDF framework, some notations and indices should be defined. Suppose that there are R layers in the IMDF. Each layer contains Q units and each unit consists of T decision trees. The components of the IMDF can be represented as follows:

- h_t represents the t-th decision tree in a unit, where $t \in [1, T]$.
- U_q denotes the q-th unit in an IMDF layer, where $q \in [1, Q]$.
- L_r is the r-th layer, which is the ensemble of multiple units, where $r \in [1, R]$.
- F_r represents the cascade of multiple layers, starting from the first layer up to the r-th layer.

Representation learning in DNNs relies on the layer-by-layer processing of the raw feature, IMDF uses a cascade structure to achieve this goal. Figure 3 illustrates a trained IMDF model. It indicates that each layer consists of two Unit-1s and two Unit-2s, where Unit-1 (Unit-2) determines that the unit algorithm uses case 1 (case 2) as the initialization strategy. Since each layer of the IMDF is the ensemble of multiple units, units with different types are deployed to encourage high diversity and promote the generalization ability of the ensemble, where the number of decision trees in each unit (T) and the number of units in each layer (Q) are hyper-parameters of the IMDF. It should be indicated that these parameters can be adjusted according to the available computing resources.

In this part, it is assumed that an unknown instance x (raw feature vector) and two classes should be predicted. Equation 7 indicates that an individual unit generates a two-dimensional class probability vector, where the vector element is defined as the weighted sum of decision trees with the same predicted result. In total, eight-dimensional (2×4) class vector is concatenated for the representation of the raw feature vector. More specifically, the DF concatenates the output of the previous layer with the raw feature vector as the input at the next layer. On the one hand, the concatenation of the class vectors and the original feature vector realizes the representation of the original feature vector. Meanwhile, the concatenated vector contains classification results of the previous layer, one implication is that the classification result of the previous layer is used to guide the classification process of the next layer. The last layer of IMDF is the output layer, the sum of the elements of all class vectors is averaged, where the element with the max value corresponds to the predicted label.

The training process of the IMDF is the same as the original DF (Zhou and Feng 2019). However, it should be emphasized that the layer number of the IMDF (R) is generated adaptively by the greedy algorithm. The correlation between the layer and cascade can be mathematically expressed as the following:

$$F_r(x) = \begin{cases} L_1(x) & r = 1 \\ L_r([x, F_{r-1}(x)]) & r > 1 \end{cases} \tag{8}$$

where $[x, F_{r-1}(x)]$ concatenates the raw feature vector and the output of the previous cascade. After expanding a new layer, the performance of the current cascade is estimated. When the performance of F_r is lower than that of F_{r-1}, the model stops training and keeps F_{r-1} as the final model. Thus, the number of cascade layers is automatically determined so that the IMDF adaptively decides its model complexity by terminating the training at the right time. This cascade structure is inspired by a layer-by-layer data processing and in-model feature transformation in DNNs. Compared with the layer number of DNNs, which is pre-set before the model training, the greed training mechanism enables the IMDF to adaptively decide the required layer number.

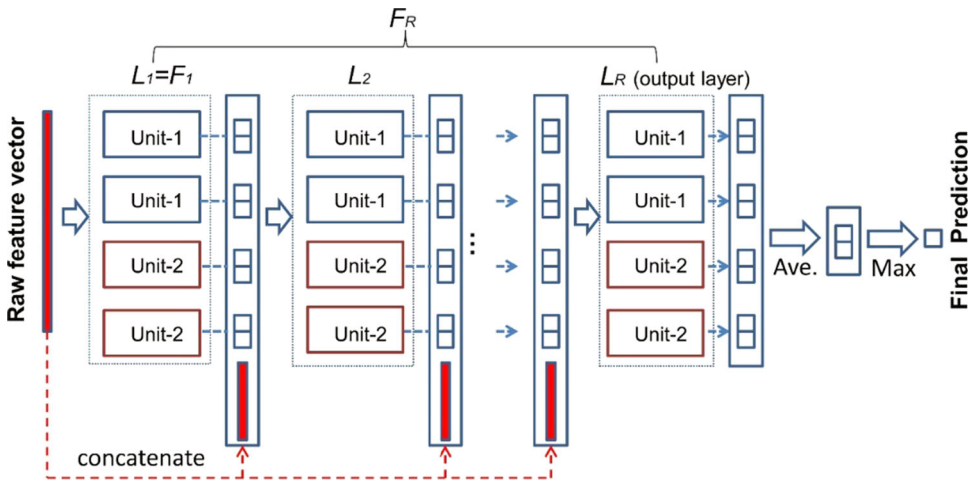


Fig. 3 The overall structure of the imbalanced deep forest

3.3 Time complexity analysis of the IMDF

Algorithm 1 shows that the time complexity should be determined for each step. In step 1, the size of the dataset after resampling is m. Therefore, the resampling process can be regarded as a loop with m times so that the time complexity of step 1 is $O(m)$. In step 2, it is supposed that the time complexity for training a decision tree is O(tree). In step 3, n minority instances are initially synthesized through the SMOTE and then the decision tree is trained. Accordingly, the time complexity of step 3 is $O(n) + O$(tree). In step 4, the learned decision tree is used to predict the labels of the original training set. Consequently, the complexity of step 4 can be regarded as O(tree). In step 5, minority instances that are misclassified by the learned decision tree are selected so that the time complexity of this step is $O(1)$. In step 6, the weighted loss of the decision tree is calculated, where the corresponding time complexity is $O(m)$. In step 7, the updating parameter is initially calculated and then weights of all instances are determined accordingly. It can be proved that the time complexity of this step is $O(1) + O(m) = O(m)$.

Finally, an adding operation is implemented on the time complexity of steps 1–7, which shows that the time complexity of the process is $O(m + n +$ tree), where m, n, and 'tree' refer to the resampling with replacement, SMOTE, and the decision tree, respectively. Based on the discussions in Sect. 3.2, each unit algorithm executes T iterations. Moreover, the IMDF consists of R layers, where each layer has Q units. Therefore, the time complexity of the overall IMDF is $R * Q * T * O(m + n +$ tree).

4 Experiments and discussions

4.1 Datasets

In this section, experiments are conducted on 15 imbalanced datasets, which are selected by increasing IR varying from 8.6 to 130. These datasets are adopted from some well-known public machine learning repositories, including UCI (Bache and Lichman 2013), LIBSVM (Chang and Lin 2011), and KDD Cup (Siddique et al. 2019). However, they have been modified as binary classification datasets for imbalanced learning (Lemaitre et al. 2017). Table 2 shows the basic information of each dataset, where the attribute "Repository and Target" represents the data source and the class of interest (minority class). For example, the dataset "ecoli," which is adopted from the UCI repository, has class "imU" as the minority class and combines other remaining classes as the majority class.

4.2 The performance analysis of Unit-1 and Unit-2

In the experiment, the parameter F-value is used as the main metric, while parameters recall and AUC are used as the auxiliary metric to evaluate the performance of Unit-1 and Unit-2 on 15 imbalanced datasets. Each unit algorithm performs 10 times fivefold cross-validation on each dataset.

In Algorithm 1, it is assumed that a and $\mathbf{b}$ are the size of minority and majority classes, respectively, where $a \leq b$ and $a + b = m$. Moreover, a' is the size of the minority class after resampling. The hyper-parameter K is the over-sampling ratio, which is defined as follows:

$$K = \frac{a'}{b} \tag{9}$$

The parameter K determines the amount of SMOTE (n) in the third step of Algorithm 1:

$$n = b * K - a \tag{10}$$

In the present study, the parameter K is set to 5%, 20%, 35%, 50%, 65%, 80%, and 100%. Moreover, it is intended to investigate the influence of the number of decision trees T on the performance of units. To this end, the parameter T is set to 50, 100, and 150. In other words, it is intended to study the performance of Unit-1 and Unit-2 as functions of parameters K and T.

The first dataset is "ecoli." It has a small size (336), and its IR is less than ten. Figures 4 and 5 show the F-values of Unit-1 and Unit-2 for the "ecoli" dataset. It is observed that when the number of decision trees T is set to 50, the F-values increase for the over-sampling ratio K ranging from 5 to 65%. Moreover, Unit-1 and Unit-2 achieve the highest F-value at $K = 20\%$ and $T = 50$, respectively. Figures 4 and 5 show that as the number of decision trees increases, there is no clear improvement in the F-value. Moreover, the change trends of the F-value curves depend on K. In other words, the optimal K value determines the best F-value.

The next dataset is "sick_euthyroid". It has a longer size (3163) and its IR is close to that of "ecoli." In the present study, it is intended to determine whether different units behave similarly on the datasets with different sizes but similar IR or not. Figures 6 and 7 illustrate that Units-1 and Unit-2 achieve the best F-values when $T = 50$ and $K = 5\%$.

According to Table 2, the IR of "ecoli" and "sick_euthyroid" are close to ten. Therefore, three other datasets with increasing IR are selected to show their F-value distribution. These datasets are the dataset "oil" (IR = 22), "mammography" (IR = 42), and "abalone_19" (IR = 130). Figures 8 and 9 indicate that for the "oil" dataset, Unit-1 and Unit-2 achieve the best F-values when $T = 50$, and K takes 20% and 5%, respectively. Moreover,

Table 2 Imbalanced datasets for binary classification tasks

Id	Datasets	Repository and target	IR	Size	Feature
1	ecoli	UCI, target: imU	8.6:1	336	7
2	sick_euthyroid	UCI, target: sick_euthyroid	9.8:1	3163	42
3	spectrometer	UCI, target: ≥ 44	11:1	531	93
4	us_crime	UCI, target: > 0.65	12:1	1994	100
5	scene	LIBSVM, target: > one label	13:1	2407	294
6	libras_move	UCI, target: 1	14:1	360	90
7	coil_2000	KDD CUP, target: minority	16:1	9822	85
8	arrhythmia	UCI, target: 06	17:1	452	278
9	solar_flare_m0	UCI, target: $M > 0$	19:1	1389	32
10	oil	UCI, target: minority	22:1	937	49
11	wine_quality	UCI, wine, target: ≤ 4	26:1	4898	11
12	ozone_level	UCI, target: ozone	34:1	2536	72
13	mammography	UCI, target: minority	42:1	11,183	6
14	protein_homo	KDD CUP, target: minority	111:1	145,751	74
15	abalone_19	UCI, target: 19	130:1	4177	10

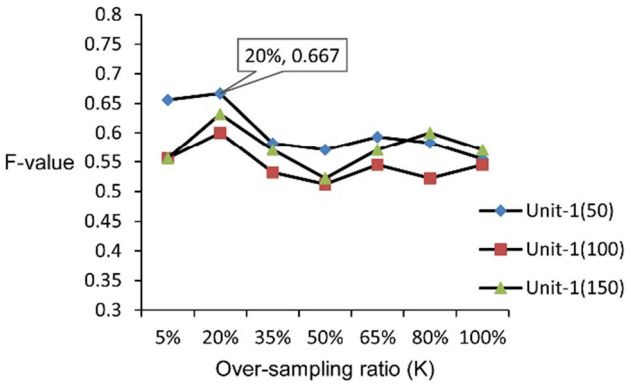


Fig. 4 *F*-value measures for "ecoli" dataset based on Unit-1

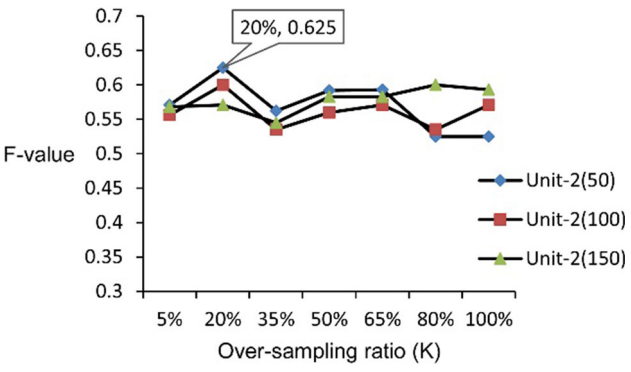


Fig. 5 *F*-value measures for "ecoli" dataset based on Unit-2

Figs. 10 and 11 illustrate that for the "mammography" dataset, Unit-1 and Unit-2 achieve the best *F*-values when $T = 50$ and $K = 5\%$. Finally, Figs. 12 and 13 show that for the highly skewed dataset "abalone_19", Unit-1 and Unit-2 achieve the best *F*-values when $T = 50$ and K equal 35% and 5%, respectively. It should be indicated that in all

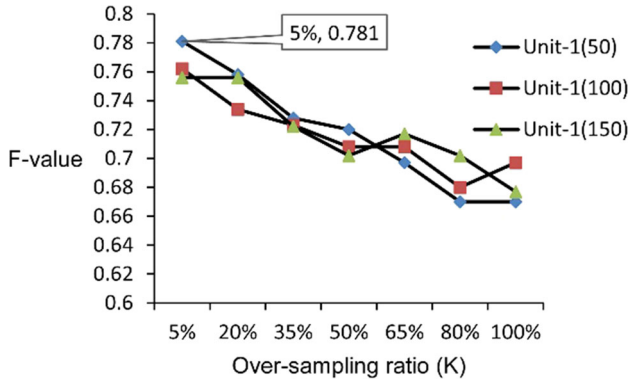


Fig. 6 *F*-value measures for "sick_euthyroid" dataset based on Unit-1

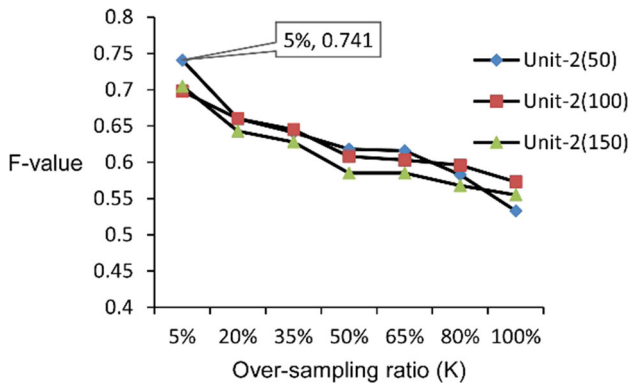


Fig. 7 *F*-value measures for "sick_euthyroid" dataset based on Unit-2

experiments, it is found that Unit-1 and Unit-2 achieve the best *F*-values when $T = 50$. Furthermore, it is found that in most datasets, the unit algorithm has the best performance when K equals 5% or 20%.

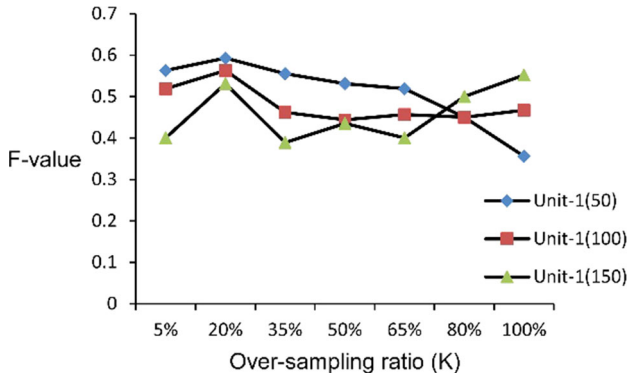


Fig. 8 *F*-value measures for "oil" dataset based on Unit-1

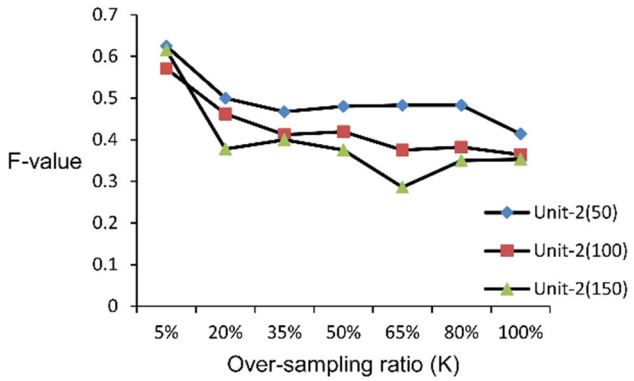


Fig. 9 *F*-value measures for "oil" dataset based on Unit-2

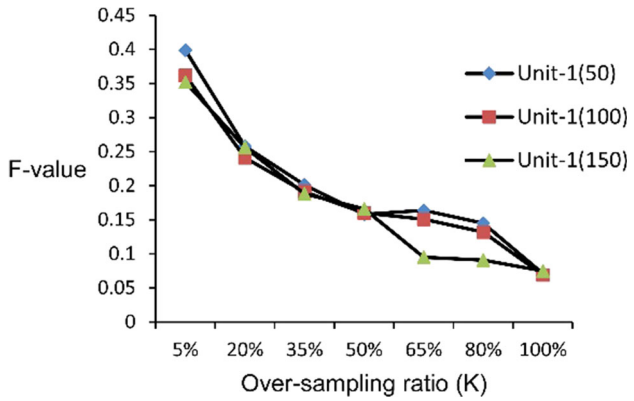


Fig. 10 *F*-value measures for "mammography" dataset based on Unit-1

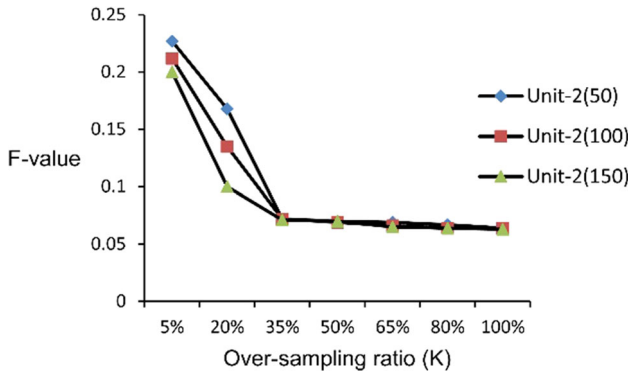


Fig. 11 *F*-value measures for "mammography" dataset based on Unit-2

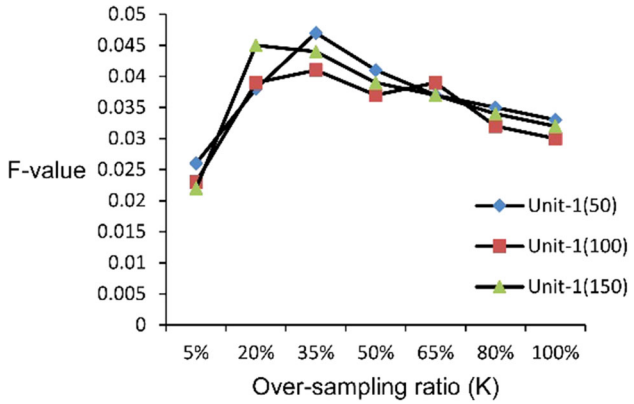


Fig. 12 *F*-value measures for "abalone_19" dataset based on Unit-1

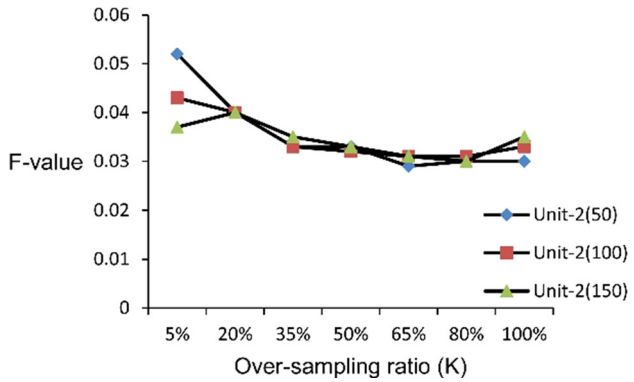


Fig. 13 *F*-value measures for "abalone_19" dataset based on Unit-2

For the convenience of comparison, T and K are set to 50 and 20%, respectively. Table 3 shows the performance of different units for different datasets. The value in parentheses ranks the unit algorithms for each dataset separately. More specifically, the best-performing algorithm gets the rank 1 and the second-best gets the rank 2. In the case of similarities (like for the arrhythmia dataset in Table 3), mean ranks are assigned (e.g. $(1 + 2)/2 = 1.5$). A smaller rank indicates better performance. The average rank for each algorithm on different datasets is shown in Table 3. From the *F*-value metric point of view, the average rank of Unit-1 (1.03) is much better than that of Unit-2 (1.97). However, the performance is the opposite in terms of the recall metric. Furthermore, from the AUC metric point of view, Unit-2 (1.40) slightly outperforms Unit-1 (1.60).

As mentioned in Sect. 3.1, the main difference between Unit-1 and Unit-2 is the initialization strategy of instance

Table 3 Performance of F-value, recall, and AUC metrics for Unit-1 and Unit-2

Id	Dataset	*F*-value		Recall		AUC	
		Unit-1	Unit-2	Unit-1	Unit-2	Unit-1	Unit-2
1	ecoli	$0.667_{(1)}$	$0.615_{(2)}$	$0.778_{(2)}$	$0.889_{(1)}$	$0.843_{(2)}$	$0.853_{(1)}$
2	sick_euthyroid	$0.781_{(1)}$	$0.740_{(2)}$	$0.868_{(2)}$	$0.882_{(1)}$	$0.916_{(1)}$	$0.885_{(2)}$
3	spectrometer	$0.880_{(1)}$	$0.833_{(2)}$	$0.909_{(2)}$	$1.000_{(1)}$	$0.927_{(2)}$	$0.956_{(1)}$
4	us_crime	$0.480_{(1)}$	$0.470_{(2)}$	$0.625_{(2)}$	$0.729_{(1)}$	$0.793_{(1.5)}$	$0.793_{(1.5)}$
5	scene	$0.267_{(1)}$	$0.245_{(2)}$	$0.540_{(2)}$	$0.556_{(1)}$	$0.618_{(1.5)}$	$0.618_{(1.5)}$
6	libras_move	$0.875_{(1)}$	$0.857_{(2)}$	$0.750_{(2)}$	$0.875_{(1)}$	$0.848_{(2)}$	$0.902_{(1)}$
7	coil_2000	$0.182_{(1)}$	$0.168_{(2)}$	$0.626_{(2)}$	$0.807_{(1)}$	$0.606_{(2)}$	$0.613_{(1)}$
8	arrhythmia	$0.889_{(1.5)}$	$0.889_{(1.5)}$	$1.000_{(1.5)}$	$1.000_{(1.5)}$	$0.963_{(2)}$	$0.981_{(1)}$
9	solar_flare_m0	$0.202_{(1)}$	$0.174_{(2)}$	$0.650_{(2)}$	$0.750_{(1)}$	$0.664_{(2)}$	$0.685_{(1)}$
10	oil	$0.625_{(1)}$	$0.593_{(2)}$	$0.538_{(2)}$	$0.615_{(1)}$	$0.781_{(1)}$	$0.778_{(2)}$
11	wine_quality	$0.258_{(1)}$	$0.187_{(2)}$	$0.478_{(2)}$	$0.681_{(1)}$	$0.659_{(1)}$	$0.629_{(2)}$
12	ozone_level	$0.265_{(1)}$	$0.232_{(2)}$	$0.381_{(2)}$	$0.524_{(1)}$	$0.685_{(2)}$	$0.713_{(1)}$
13	mammography	$0.437_{(1)}$	$0.227_{(2)}$	$0.714_{(2)}$	$0.857_{(1)}$	$0.829_{(1)}$	$0.723_{(2)}$
14	protein_homo	$0.767_{(1)}$	$0.543_{(2)}$	$0.879_{(2)}$	$0.943_{(1)}$	$0.932_{(1.5)}$	$0.932_{(1.5)}$
15	abalone_19	$0.052_{(1)}$	$0.047_{(2)}$	$0.727_{(2)}$	$0.909_{(1)}$	$0.727_{(1.5)}$	$0.727_{(1.5)}$
	Average rank:	(1.03)	(1.97)	(1.97)	(1.03)	(1.60)	(1.40)

weights. More specifically, Unit-1 moderately improves TP and minimizes the degradation of TN, while Unit-2 maximizes TP without considering the loss of TN. Therefore, it is concluded that Unit-1 and Unit-2 have better performance from the *F*-value and recall metrics points of view, respectively.

4.3 The performance analysis of IMDFs with different configurations

To construct a good ensemble, the base classifier should be accurate and diverse. Otherwise, the combination of base classifiers has a negligible impact to improve the classification performance (Dai et al. 2017; Zhou and Feng 2019). In the cascade structure, applying different units in each layer encourages reasonable diversity among base classifiers. Moreover, assigning the appropriate K to each unit ensures each base classifier to achieve the best performance. It should be indicated that both of them are necessary for IMDF's performance. The number of units in each layer is a user-defined parameter that depends on the available computational resource. For simplicity, the unit number in each layer is set to four, and some superscripts are defined to denote different configurations of the IMDF. Table 4 presents these configurations.

Table 4 Different configurations for IMDF

Notation	Configuration
$\text{IMDF}^{(1x4)}$	Four Unit-1 s in each layer
$\text{IMDF}^{(2x4)}$	Four Unit-2 s in each layer
$\text{IMDF}^{(1x2+2x2)}$	Two Unit-1 s and two Unit-2 s in each layer

Figures 14, 15, 16, and 17 show the performance of IMDFs with different configurations on the first four datasets of Table 2. It is observed that the green bars with number tags stand for the best performances. Therefore, the configuration of $\text{IMDF}^{(1\times2+2\times2)}$ is better than that of $\text{IMDF}^{(1\times4)}$ and $\text{IMDF}^{(2\times4)}$. Moreover, the optimal performance of $\text{IMDF}^{(1\times2+2\times2)}$ can be achieved when $T = 50$ and K is about 20%, where K values for Unit-1 and Unit-2 are the same.

Table 5 presents the best *F*-values of IMDFs with different configurations on different datasets. It is observed that the average rank of $\text{IMDF}^{(1\times2+2\times2)}$ is better than that of $\text{IMDF}^{(1\times4)}$ and $\text{IMDF}^{(2\times4)}$. It is inferred that the combination of different units is better than the combination of the same units.

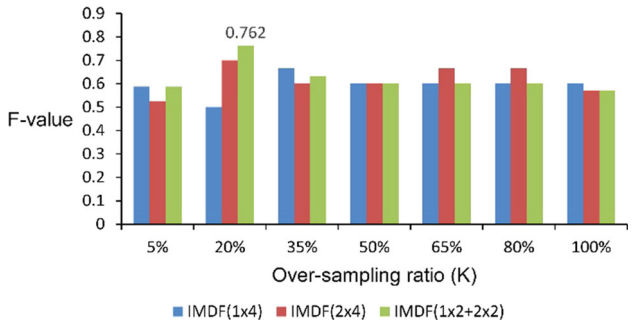


Fig. 14 *F*-values measures for "ecoli" dataset based on IMDFs with different configurations

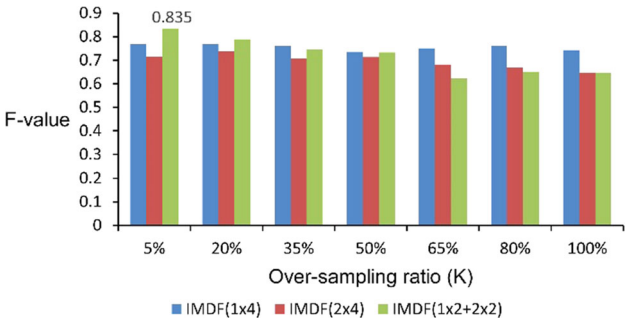


Fig. 15 *F*-values measures for "sick_euthroid" dataset based on IMDFs with different configurations

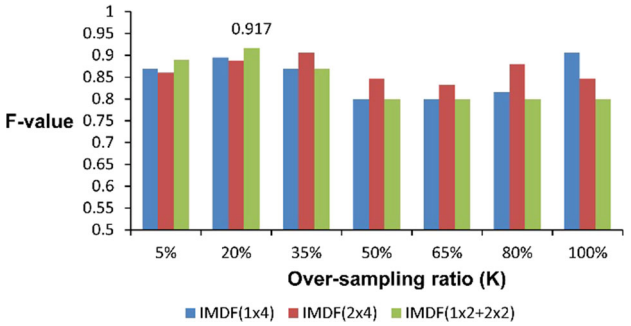


Fig. 16 *F*-values measures for "spectrometer" dataset based on IMDFS with different configurations

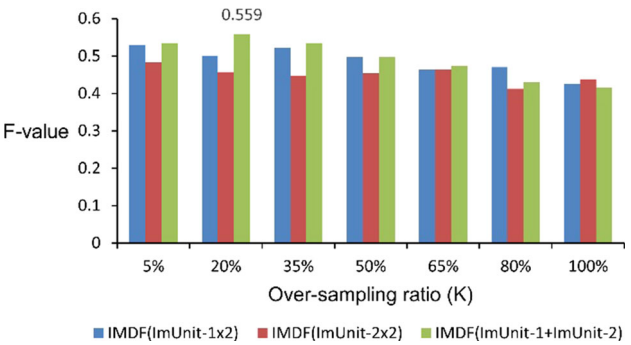


Fig. 17 *F*-values measures for "us_crime" dataset based on IMDFS with different configurations

In Table 5, $\text{IMDF}^{(1\times 2+2\times 2)}$ is also compared with the original DF. It should be indicated that the DF contains two random forests and two completely random tree forests in each layer with 500 decision trees in each forest (Zhou and Feng 2019). It is observed that three zero values appear on the result of DF; this is because the DF incorrectly classifies all minority instances. In the subsequent experiments, $\text{IMDF}^{(1\times 2+2\times 2)}$ with $T = 50$ and $K = 20\%$ is adopted as the standard configuration, and the superscript is not marked.

4.4 The performance comparison between IMDF and other algorithms

In this section, the IMDF is initially compared with its components, including decision tree, AdaBoost, and SMOTE. It should be indicated that the decision tree and AdaBoost can be directly applied to calculate *F*-value, recall, and AUC. However, SMOTE is combined with the decision tree to predict unknown instances.

Moreover, IMDF is compared with two classic algorithms, including SMOTEBoost and RUSBoost. It should be indicated that both of them handle the imbalanced data based on the resampling and boosting mechanism. The performance of SMOTEBoost depends on the number of minority instances after over-sampling, while the performance of RUSBoost depends on the number of majority instances after random under-sampling. Therefore, the performance of these two algorithms is also based on adjusting the K parameters.

Furthermore, a conventional DNN is also used for the comparison. It should be indicated that the DNN obtains four hidden layers, where the sizes of hidden layers are 512, 256, 128, and 64, and activation = "relu." The training set by SMOTE is rebalanced before using DNN. Tables 6, 7, and 8 show the comparison of the performance of F-value, recall, and AUC between IMDF and other algorithms, respectively.

In order to analyze Tables 6, 7, and 8 reasonably, both the Friedman test and the Nemenyi test (Demsar 2006) are utilized to determine whether the performance of IMDF is different from that of algorithms. The Friedman test is an improved non-parametric statistical test, which denotes that the performances of all algorithms are equivalent under the null hypothesis. By using Tables 6, 7, and 8, the performances of all algorithms are ranked on each dataset. The rank values are shown in parentheses. Let r_i^j be the rank of the jth of K algorithms on the ith of N datasets. K and N denote the number of algorithms and the number of datasets, respectively. The average rank (R_j) of each algorithm is provided at the bottom of each table, which is defined as follows:

$$R_j = \frac{1}{N}\sum_i r_i^j, \quad i \in [1, N] \tag{11}$$

The smaller the R_j, the better the average performance of the algorithm on all datasets. Moreover, Eq. 12 shows the original Friedman statistic, which is considered as too conservative. The improved version is described as Eq. 13, which is distributed according to the *F*-distribution with $(K-1)$ and $(K-1)(N-1)$ degrees of freedom.

Table 5 F-value performance of IMDFs for different configurations

Id	Dataset	IMDF$^{(1 \times 4)}$	IMDF$^{(2 \times 4)}$	IMDF$^{(1 \times 2 + 2 \times 2)}$	DF
1	ecoli	$0.667_{(3)}$	$0.715_{(2)}$	$0.762_{(1)}$	$0.462_{(4)}$
2	sick_euthyroid	$0.778_{(3)}$	$0.735_{(4)}$	$0.835_{(2)}$	$0.866_{(1)}$
3	spectrometer	$0.917_{(2)}$	$0.917_{(2)}$	$0.917_{(2)}$	$0.588_{(4)}$
4	us_crime	$0.522_{(2)}$	$0.497_{(3)}$	$0.559_{(1)}$	$0.424_{(4)}$
5	scene	$0.259_{(2)}$	$0.236_{(3)}$	$0.295_{(1)}$	$0.061_{(4)}$
6	libras_move	$0.933_{(2)}$	$0.933_{(2)}$	$0.933_{(2)}$	$0.857_{(4)}$
7	coil_2000	$0.189_{(2)}$	$0.161_{(3)}$	$0.208_{(1)}$	$0.023_{(4)}$
8	arrhythmia	$0.909_{(2)}$	$0.889_{(3)}$	$0.941_{(1)}$	$0.000_{(4)}$
9	solar_flare_m0	$0.296_{(2)}$	$0.265_{(3)}$	$0.308_{(1)}$	$0.000_{(4)}$
10	oil	$0.645_{(3)}$	$0.622_{(4)}$	$0.671_{(2)}$	$0.750_{(1)}$
11	wine_quality	$0.328_{(2)}$	$0.281_{(3)}$	$0.366_{(1)}$	$0.225_{(4)}$
12	ozone_level	$0.268_{(2)}$	$0.241_{(3)}$	$0.293_{(1)}$	$0.000_{(4)}$
13	mammography	$0.583_{(3)}$	$0.556_{(4)}$	$0.639_{(1.5)}$	$0.639_{(1.5)}$
14	protein_homo	$0.828_{(3)}$	$0.804_{(4)}$	$0.867_{(1.5)}$	$0.867_{(1.5)}$
15	abalone_19	$0.068_{(2)}$	$0.061_{(3)}$	$0.083_{(1)}$	$0.000_{(4)}$
	Average rank:	(2.33)	(3.07)	(1.33)	(3.27)

Table 6 F-value comparison between IMDF and other algorithms

Id	Dataset	Decision Tree	Ada boost	SMOTE then tree	RUS boost	SMOTE boost	DNN	IMDF
1	ecoli	$0.444_{(6)}$	$0.625_{(4)}$	$0.471_{(5)}$	$0.667_{(2.5)}$	$0.667_{(2.5)}$	$0.322_{(7)}$	$0.762_{(1)}$
2	sick_euthyroid	$0.802_{(5.5)}$	$0.805_{(4)}$	$0.802_{(5.5)}$	$0.827_{(3)}$	$0.835_{(1.5)}$	$0.644_{(7)}$	$0.835_{(1.5)}$
3	spectrometer	$0.727_{(6)}$	$0.857_{(4)}$	$0.762_{(5)}$	$0.895_{(2)}$	$0.885_{(3)}$	$0.556_{(7)}$	$0.917_{(1)}$
4	us_crime	$0.408_{(5.5)}$	$0.395_{(7)}$	$0.408_{(5.5)}$	$0.469_{(3.5)}$	$0.469_{(3.5)}$	$0.473_{(2)}$	$0.559_{(1)}$
5	scene	$0.173_{(7)}$	$0.214_{(5)}$	$0.229_{(4)}$	$0.184_{(6)}$	$0.267_{(3)}$	$0.298_{(1)}$	$0.295_{(2)}$
6	libras_move	$0.500_{(7)}$	$0.545_{(6)}$	$0.824_{(4)}$	$0.857_{(2.5)}$	$0.857_{(2.5)}$	$0.673_{(5)}$	$0.933_{(1)}$
7	coil_2000	$0.164_{(4.5)}$	$0.045_{(7)}$	$0.171_{(3)}$	$0.148_{(6)}$	$0.164_{(4.5)}$	$0.181_{(2)}$	$0.208_{(1)}$
8	arrhythmia	$0.842_{(4.5)}$	$0.778_{(6)}$	$0.842_{(4.5)}$	$0.889_{(2)}$	$0.880_{(3)}$	$0.210_{(7)}$	$0.941_{(1)}$
9	solar_flare_m0	$0.194_{(4)}$	$0.091_{(7)}$	$0.194_{(4)}$	$0.220_{(2)}$	$0.194_{(4)}$	$0.099_{(6)}$	$0.308_{(1)}$
10	oil	$0.483_{(6)}$	$0.522_{(4.5)}$	$0.522_{(4.5)}$	$0.583_{(2)}$	$0.571_{(3)}$	$0.221_{(7)}$	$0.671_{(1)}$
11	wine_quality	$0.328_{(4.5)}$	$0.301_{(6)}$	$0.330_{(2.5)}$	$0.328_{(4.5)}$	$0.330_{(2.5)}$	$0.247_{(7)}$	$0.366_{(1)}$
12	ozone_level	$0.143_{(5)}$	$0.176_{(4)}$	$0.196_{(2.5)}$	$0.133_{(6)}$	$0.196_{(2.5)}$	$0.120_{(7)}$	$0.293_{(1)}$
13	mammography	$0.613_{(2.5)}$	$0.517_{(6)}$	$0.555_{(5)}$	$0.613_{(2.5)}$	$0.591_{(4)}$	$0.507_{(7)}$	$0.639_{(1)}$
14	protein_homo	$0.749_{(5.5)}$	$0.786_{(4)}$	$0.749_{(5.5)}$	$0.814_{(3)}$	$0.825_{(2)}$	$0.579_{(7)}$	$0.867_{(1)}$
15	abalone_19	$0.000_{(6)}$	$0.000_{(6)}$	$0.051_{(2.5)}$	$0.000_{(6)}$	$0.051_{(2.5)}$	$0.026_{(4)}$	$0.083_{(1)}$
	Average rank:	(5.30)	(5.36)	(4.20)	(3.57)	(2.93)	(5.53)	(1.10)

$$X_{\mathrm{F}}^2 = \frac{12N}{K(K+1)}\left[\sum_j R_j^2 - \frac{K(K+1)^2}{4}\right] \tag{12}$$

$$F_{\mathrm{F}} = \frac{(N-1)X_{\mathrm{F}}^2}{N(K-1) - X_{\mathrm{F}}^2} \tag{13}$$

In the experiments, N and K are set to 15 and 7, respectively. The critical value of F_{F} is 2.209 with a 95% confidence level, which can be found in any statistical book. The Friedman statistical results of Tables 6, 7, and 8 are 15.51, 13.56, and 11.26, respectively. All the statistics are much higher than 2.209. Therefore, the null hypothesis can be safely rejected. In other words, there are significant performance differences among these algorithms.

Then, the Nemenyi test is performed to verify the performance difference between pairwise algorithms. The performances of two classifiers are significantly different if the corresponding average ranks differ by at least the critical difference (CD):

Table 7 Recall comparison between IMDF and other algorithms

Id	Dataset	Decision Tree	Ada boost	SMOTE then tree	RUS boost	SMOTE boost	DNN	IMDF
1	ecoli	$0.444_{(6)}$	$0.556_{(5)}$	$0.335_{(7)}$	$0.756_{(4)}$	$0.778_{(3)}$	$1.000_{(1)}$	$0.889_{(2)}$
2	sick_euthyroid	$0.829_{(6)}$	$0.842_{(5)}$	$0.908_{(1.5)}$	$0.855_{(4)}$	$0.882_{(3)}$	$0.816_{(7)}$	$0.908_{(1.5)}$
3	spectrometer	$0.727_{(6)}$	$1.000_{(1)}$	$0.909_{(3.5)}$	$0.909_{(3.5)}$	$0.909_{(3.5)}$	$0.673_{(7)}$	$0.909_{(3.5)}$
4	us_crime	$0.438_{(6)}$	$0.319_{(7)}$	$0.458_{(5)}$	$0.500_{(4)}$	$0.625_{(2)}$	$0.521_{(3)}$	$0.792_{(1)}$
5	scene	$0.175_{(6)}$	$0.143_{(7)}$	$0.365_{(3)}$	$0.206_{(5)}$	$0.540_{(1)}$	$0.295_{(4)}$	$0.476_{(2)}$
6	libras_move	$0.375_{(6.5)}$	$0.375_{(6.5)}$	$0.750_{(2)}$	$0.625_{(4.5)}$	$0.750_{(2)}$	$0.625_{(4.5)}$	$0.750_{(2)}$
7	coil_2000	$0.170_{(5)}$	$0.023_{(7)}$	$0.158_{(6)}$	$0.205_{(4)}$	$0.626_{(2)}$	$0.443_{(3)}$	$0.708_{(1)}$
8	arrhythmia	$1.000_{(3)}$	$0.875_{(6)}$	$1.000_{(3)}$	$1.000_{(3)}$	$1.000_{(3)}$	$0.625_{(7)}$	$1.000_{(3)}$
9	solar_flare_m0	$0.150_{(6.5)}$	$0.150_{(6.5)}$	$0.250_{(5)}$	$0.450_{(4)}$	$0.650_{(3)}$	$0.950_{(1)}$	$0.850_{(2)}$
10	oil	$0.538_{(4)}$	$0.462_{(6)}$	$0.538_{(4)}$	$0.769_{(1)}$	$0.538_{(4)}$	$0.415_{(7)}$	$0.615_{(2)}$
11	wine_quality	$0.290_{(6)}$	$0.203_{(7)}$	$0.348_{(5)}$	$0.420_{(4)}$	$0.478_{(3)}$	$0.600_{(2)}$	$0.885_{(1)}$
12	ozone_level	$0.143_{(6.5)}$	$0.143_{(6.5)}$	$0.238_{(4)}$	$0.429_{(2)}$	$0.381_{(3)}$	$0.200_{(5)}$	$0.476_{(1)}$
13	mammography	$0.597_{(5)}$	$0.468_{(7)}$	$0.792_{(3)}$	$0.558_{(6)}$	$0.714_{(4)}$	$0.826_{(2)}$	$0.909_{(1)}$
14	protein_homo	$0.768_{(5)}$	$0.711_{(7)}$	$0.800_{(4)}$	$0.738_{(6)}$	$0.879_{(1.5)}$	$0.873_{(3)}$	$0.879_{(1.5)}$
15	abalone_19	$0.000_{(6.5)}$	$0.000_{(6.5)}$	$0.182_{(4.5)}$	$0.182_{(4.5)}$	$0.227_{(3)}$	$0.273_{(2)}$	$0.455_{(1)}$
	Average rank:	(5.60)	(6.07)	(4.03)	(3.97)	(2.73)	(3.90)	(1.70)

Table 8 AUC comparison between IMDF and other algorithms

Id	Dataset	Decision Tree	Ada boost	SMOTE then tree	RUS boost	SMOTE boost	DNN	IMDF
1	ecoli	$0.659_{(6)}$	$0.767_{(4)}$	$0.656_{(7)}$	$0.772_{(3)}$	$0.843_{(2)}$	$0.765_{(5)}$	$0.928_{(1)}$
2	sick_euthyroid	$0.904_{(6)}$	$0.910_{(5)}$	$0.938_{(1)}$	$0.918_{(3)}$	$0.916_{(4)}$	$0.874_{(7)}$	$0.929_{(2)}$
3	spectrometer	$0.854_{(6)}$	$0.993_{(1)}$	$0.911_{(5)}$	$0.941_{(3)}$	$0.927_{(4)}$	$0.721_{(7)}$	$0.951_{(2)}$
4	us_crime	$0.688_{(6)}$	$0.644_{(7)}$	$0.691_{(5)}$	$0.717_{(4)}$	$0.793_{(2)}$	$0.730_{(3)}$	$0.831_{(1)}$
5	scene	$0.547_{(7)}$	$0.562_{(6)}$	$0.620_{(2)}$	$0.568_{(5)}$	$0.618_{(3)}$	$0.616_{(4)}$	$0.626_{(1)}$
6	libras_move	$0.682_{(7)}$	$0.688_{(6)}$	$0.870_{(2)}$	$0.812_{(4)}$	$0.848_{(3)}$	$0.803_{(5)}$	$0.875_{(1)}$
7	coil_2000	$0.557_{(5)}$	$0.511_{(7)}$	$0.574_{(4)}$	$0.554_{(6)}$	$0.606_{(3)}$	$0.612_{(2)}$	$0.652_{(1)}$
8	arrhythmia	$0.988_{(2)}$	$0.926_{(6)}$	$0.969_{(4)}$	$0.984_{(3)}$	$0.963_{(5)}$	$0.595_{(7)}$	$0.996_{(1)}$
9	solar_flare_m0	$0.565_{(5)}$	$0.524_{(7)}$	$0.595_{(4)}$	$0.665_{(2.5)}$	$0.665_{(2.5)}$	$0.541_{(6)}$	$0.736_{(1)}$
10	oil	$0.753_{(3)}$	$0.723_{(5.5)}$	$0.741_{(4)}$	$0.864_{(1)}$	$0.781_{(2)}$	$0.645_{(7)}$	$0.723_{(5.5)}$
11	wine_quality	$0.632_{(6)}$	$0.589_{(7)}$	$0.649_{(4.5)}$	$0.668_{(2)}$	$0.659_{(3)}$	$0.711_{(1)}$	$0.649_{(4.5)}$
12	ozone_level	$0.559_{(6)}$	$0.565_{(5)}$	$0.597_{(4)}$	$0.700_{(1.5)}$	$0.685_{(3)}$	$0.507_{(7)}$	$0.700_{(1.5)}$
13	mammography	$0.795_{(5)}$	$0.732_{(7)}$	$0.885_{(2)}$	$0.766_{(6)}$	$0.829_{(4)}$	$0.895_{(1)}$	$0.850_{(3)}$
14	protein_homo	$0.883_{(5)}$	$0.855_{(7)}$	$0.895_{(4)}$	$0.866_{(6)}$	$0.933_{(1.5)}$	$0.931_{(3)}$	$0.933_{(1.5)}$
15	abalone_19	$0.495_{(7)}$	$0.500_{(6)}$	$0.572_{(3)}$	$0.547_{(4)}$	$0.727_{(1)}$	$0.542_{(5)}$	$0.633_{(2)}$
	Average rank:	(5.47)	(5.77)	(3.70)	(3.60)	(2.87)	(4.67)	(1.93)

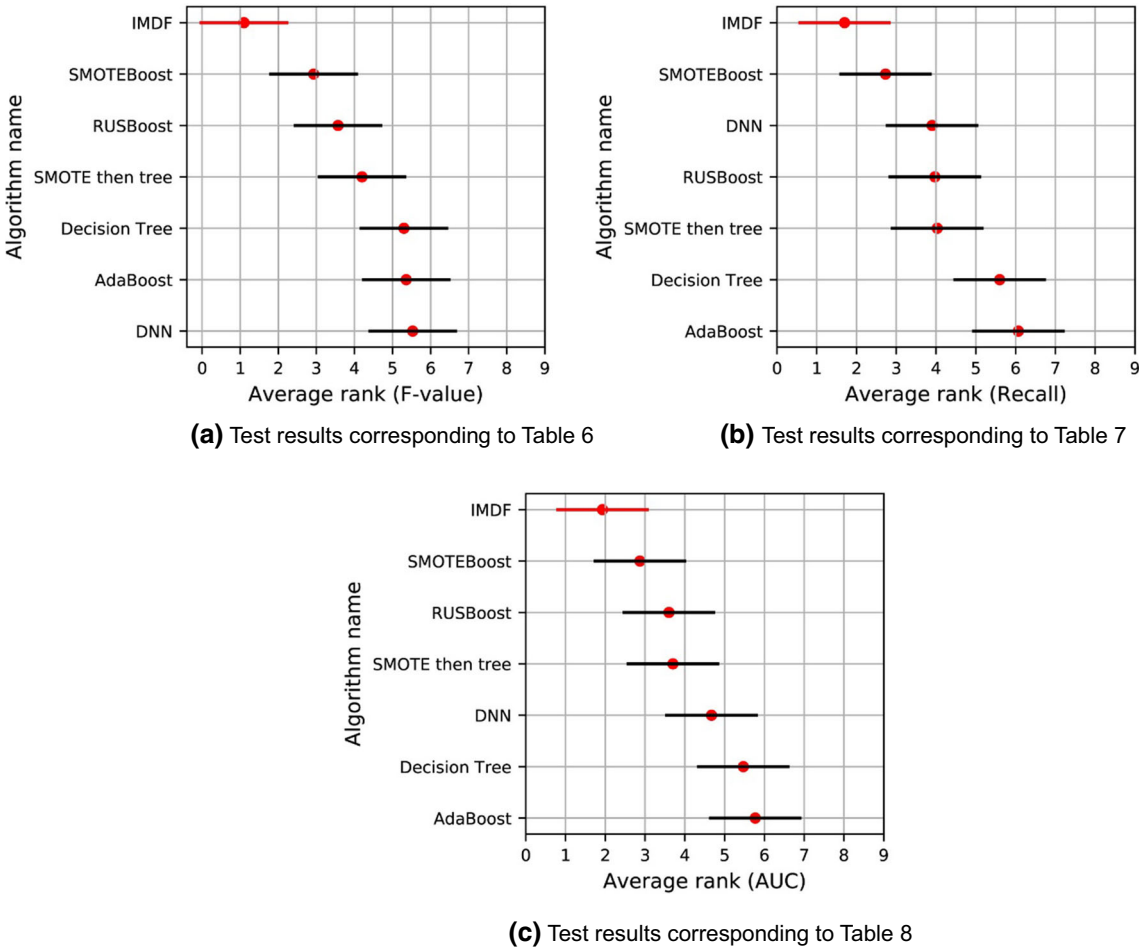
$$\mathrm{CD} = q_\alpha \sqrt{\frac{K(K+1)}{6N}} \tag{14}$$

According to Eq. 14 and Table 9, in the conducted experiments, $\mathrm{CD} = 2.949\sqrt{\frac{7(7+1)}{6*15}} = 2.326$ with 95% confidence level. Figure 15a–c shows the obtained results from Tables 6, 7 and 8, respectively. In Fig. 18, the average rank of each algorithm is marked by a red dot, and a bar across each dot shows the range of CD. It is observed that the two algorithms are significantly different when there is no overlap between the bars.

Figure 18a–c shows that the average ranks of IMDF are the best on three metrics. Since the obtained results always get the lowest scores in all subfigures, it is safe to conclude that IMDF gains the highest rank in all cases. Moreover, in

Table 9 Critical values for the two-tailed Nemenyi test

#Num of algorithms	2	3	4	5	6	7	8	9	10
$q_{0.05}$	1.960	2.343	2.569	2.728	2.850	2.949	3.031	3.102	3.164
$q_{0.10}$	1.645	2.052	2.291	2.459	2.589	2.693	2.780	2.855	2.920



(a) Test results corresponding to Table 6

(b) Test results corresponding to Table 7

(c) Test results corresponding to Table 8

Fig. 18 The comparison results between seven algorithms based on the Nemenyi test

most cases, the bar of the IMDF does not overlap with the bars of its components, including SMOTE, AdaBoost, and decision tree as well as DNN. Therefore, the performance of the IMDF is superior to that of its components as well as DNN on the three metrics. Furthermore, although SMOTEBoost is the second best, it is still half bar behind IMDF.

5 Conclusions

In the present study, the IMDF algorithm is proposed. It is intended to utilize each unit of the IMDF to handle the imbalanced data so that the classification results of the entire IMDF are biased toward the minority class. Therefore, an improved AdaBoost algorithm is used as the unit of the IMDF. In each iteration of the AdaBoost algorithm, more attention is paid to misclassified minority instances, and learning on them is strengthened in the next iteration by implementing SMOTE. The numerical experiments indicate that incorporating the proposed AdaBoost algorithm into the DF lead to outperforming results. It should be indicated that IMDF is not performed on large datasets such as MNIST(Lecun et al. 1998) and CIFAR-10(Su et al. 2019). The main reason is that IMDF aims to explore the application of deep learning on small-sized imbalanced datasets.

The limitations of the proposed IMDF are as the following: Firstly, it is required to adjust the over-sampling

ratio K of SMOTE to reach the best performance of the proposed unit algorithm. In other words, using SMOTE leads to an additional tuning parameter of the IMDF. Moreover, the unit algorithm generates new synthesized minority instances that participate in the training of base classifiers in each iteration. This operation leads to additional computational time that should be considered. Furthermore, in the cascading structure of the IMDF, the output of the previous layer (class probability vectors) is concatenated with original features as the input of the next layer. It is expected that the classification results of the previous layer can guide the classification process of the next layer. However, if the original features are of high dimensionality, the information of the output of the previous layer is likely to be overwhelmed. In order to solve this problem, the number of units at every level of the forest cascade is increased. However, this method leads to additional computational time.

Furthermore, the following important future research directions for IMDF are recommended. Firstly, IMDF solves the classification task of imbalanced datasets. However, the ideas underlying the IMDF can be extended to the machine learning models solving regression problems of imbalanced datasets. For example, the modified AdaBoost regressor (Drucker 1997) is used as the unit of IMDF. Secondly, the current IMDF is only applied to the binary classification for imbalanced datasets. Moreover, the identification for multiple minority classes can be simultaneously improved in the case of the multi-class classification. For example, the modified AdaBoost-SAMME (Zhu et al. 2009) can be used as the basic framework of the unit algorithm. Thirdly, it should be indicated that the number of units at every cascade layer should be high enough to prevent concatenated class probability vectors being overwhelmed. This situation significantly increases the computational complexity of the IMDF. The abovementioned defect can be alleviated by applying an interesting method called the confidence screening mechanism (Pang et al. 2018), which significantly reduces the training and testing times of IMDF at each level. The main idea of the confidence screening mechanism is training instances with high confidence that can be directly passed to the final layer rather than passing layer-by-layer.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Nos. 61772023, 61502402) and the Fundamental Research Funds for the Central Universities (No. 20720180073).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Human and animals rights This article does not contain any studies with human participants or animals performed by any of the authors.

References

Anand R, Mehrotra KG, Mohan CK, Ranka S (1993) An improved algorithm for neural network classification of imbalanced training sets. IEEE Trans Neural Netw 4:962–969. https://doi.org/10.1109/72.286891

Bache K, Lichman M (2013) UCI machine learning repository. School of Information and Computer Sciences, University of California, Irvine. [Online]. http://archive.ics.uci.edu/ml

Branco P, Torgo L, Ribeiro RP (2016) A survey of predictive modeling on imbalanced domains. ACM Comput Surv 49:1–50. https://doi.org/10.1145/2907070

Chang C-C, Lin C-J (2011) LIBSVM: a library for support vector machines. ACM Trans Intell Syst Technol. https://doi.org/10.1145/1961189.1961199

Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357. https://doi.org/10.1613/jair.953

Chawla NV, Lazarevic A, Hall LO, Bowyer KW (2003) SMOTE-Boost: improving prediction of the minority class in boosting. Eur Conf Princ Data Min Knowl Discov. https://doi.org/10.1007/978-3-540-39804-2_12

Dai Q, Ye R, Liu Z (2017) Considering diversity and accuracy simultaneously for ensemble pruning. Appl Soft Comput 58:75–91. https://doi.org/10.1016/j.asoc.2017.04.058

Demsar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Drucker H (1997) Improving regressors using boosting techniques. In: Proceedings of the fourteenth international conference on machine learning, pp 107–115

Fernandez A, GarcIa S, Herrera F (2018) SMOTE for learning from imbalanced data: progress and challenges. J Artif Intell Res 61:863–905. https://doi.org/10.1613/jair.1.11192

Freund Y, Schapire R (1996) Experiments with a new boosting algorithm. In: Proceedings of the thirteenth international conference on machine learning

Haixiang G, Yijing L, Shang J, Mingyun G, Yuanyue H, Bing G (2017) Learning from class-imbalanced data: review of methods and applications. Expert Syst Appl 73:220–239. https://doi.org/10.1016/j.eswa.2016.12.035

Jiang L, Li C, Wang S, Zhang L (2016) Deep feature weighting for naive Bayes and its application to text classification. Eng Appl Artif Intell 52:26–39. https://doi.org/10.1016/j.engappai.2016.02.002

Johnson JM, Khoshgoftaar TM (2019) Survey on deep learning with class imbalance. J Big Data 6:27. https://doi.org/10.1186/s40537-019-0192-5

Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 11:2278–2324. https://doi.org/10.1109/5.726791

Lemaitre G, Nogueira F, Oliveira D, Aridas C (2017) Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning. J Mach Learn Res 18:1–5

Loyola-González O, Martínez-Trinidad JF, Carrasco-Ochoa JA, García-Borroto M (2016) Study of the impact of resampling methods for contrast pattern based classifiers in imbalanced databases. Neurocomputing 175:935–947. https://doi.org/10.1016/j.neucom.2015.04.120

Loyola-González O, Medina-Pérez MA, Martínez-Trinidad JF, Carrasco-Ochoa JA, Monroy R, García-Borroto M (2017) PBC4cip: a new contrast pattern-based classifier for class imbalance

problems. Knowl-Based Syst 115:100–109. https://doi.org/10.1016/j.knosys.2016.10.018

Maher Maalouf TBT (2011) Robust weighted kernel logistic regression in imbalanced and rare events data. Comput Stat Data Anal 55:168–183. https://doi.org/10.1016/j.csda.2010.06.014

Nie G, Rowe W, Zhang L, Tian Y, Shi Y (2011) Credit card churn forecasting by logistic regression and decision tree. Expert Syst Appl 38:15273–15285. https://doi.org/10.1016/j.eswa.2011.06.028

Pang M, Ting K-M, Zhao P, Zhou Z-H (2018) Improving deep forest by confidence screening. In: 18th IEEE international conference on data mining, pp 1194–1199. https://doi.org/10.1109/ICDM.2018.00158

Seiffert C, Khoshgoftaar TM, Van Hulse J, Napolitano A (2010) RUSBoost: a hybrid approach to alleviating class imbalance. IEEE Trans Syst Man Cybern Part A-Syst Hum 40:185–197. https://doi.org/10.1109/TSMCA.2009.2029559

Siddique K, Akhtar Z, Khan FA, Kim Y (2019) KDD Cup 99 data sets: a perspective on the role of data sets in network intrusion detection research. Computer 52:41–51. https://doi.org/10.1109/MC.2018.2888764

Su J, Vargas DV, Sakurai K (2019) One pixel attack for fooling deep neural networks. IEEE Trans Evol Comput 23:828–841. https://doi.org/10.1109/TEVC.2019.2890858

Utkin L (2019) An imprecise deep forest for classification. Expert Syst Appl. https://doi.org/10.1016/j.eswa.2019.112978

Utkin L, Ryabinin MA (2018) A siamese deep forest. Knowl Based Syst 139:13–22. https://doi.org/10.1016/j.knosys.2017.10.006

Utkin L, Kovalev MS, Meldo AA (2019) A deep forest classifier with weights of class probability distribution subsets. Knowl Based Syst 173:15–27. https://doi.org/10.1016/j.knosys.2019.02.022

Zhou Z-H, Feng J (2017) Deep forest: towards an alternative to deep neural networks. In: Proceedings of the twenty-sixth international joint conference on artificial intelligence, pp 3553–3559. https://doi.org/10.24963/ijcai.2017/497

Zhou Z-H, Feng J (2019) Deep forest. Natl Sci Rev 6:74–86. https://doi.org/10.1093/nsr/nwy108

Zhu J, Zou H, Rosset S, Hastie T (2009) Multi-class AdaBoost. Stat Interface 2:349–360. https://doi.org/10.4310/SII.2009.v2.n3.a8