Knowledge-Based Systems 158 (2018) 81–93

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

An empirical comparison on state-of-the-art multi-class imbalance learning algorithms and a new diversified ensemble learning scheme

Jingjun Bi[a], Chongsheng Zhang[a,*]

[a] The Big Data Research Center, Henan University, Kaifeng 475001, China

ARTICLE INFO

Keywords:
Multi-class imbalance learning
Classification algorithms
Decomposition methods for multi-class data
Multi-class imbalanced data classification

ABSTRACT

Class-imbalance learning is one of the most challenging problems in machine learning. As a new and important direction in this field, multi-class imbalanced data classification has attracted a great many research focus in recent years. In this paper, we first make a very comprehensive review on state-of-the-art classification algorithms for multi-class imbalanced data. Moreover, we propose a new multi-class imbalance classification algorithm, which is hereafter referred to as the Diversified Error Correcting Output Codes (DECOC) method. The main idea of DECOC is to combine the improved ECOC (Error Correcting Output Codes) method for tackling class imbalance, and the diversified ensemble learning framework, which finds the best classification algorithm (out of many heterogeneous classification algorithms) for each individual sub-dataset resampled from the original data. We conduct experiments on 19 public datasets to empirically compare the performance of DECOC with 17 state-of-the-art multi-class imbalance learning algorithms, using 4 different accuracy measures: overall accuracy, Geometric mean, F-measure, and Area Under Curve. Experimental results demonstrate that DECOC achieves significantly better accuracy performance than the other 17 algorithms on these accuracy metrics. To advance research in this field, we make all the source codes of DECOC and the above-mentioned 17 state-of-the-art algorithms for imbalanced data classification be available at GitHub: https://github.com/chongshengzhang/Multi_Imbalance.

1. Introduction

Imbalance learning has received significant research efforts in the past decade [1–3], which has many practical applications, such as credit card fraud detection, medical diagnosis, etc. Well-established algorithms for binary imbalance learning are SMOTE [4], ADASYN [5], EasyEnsemble and BalanceCascade [6].

In recent years, many researchers have been focusing on multi-class imbalanced data classification [7], which is more challenging than binary imbalance learning. Existing algorithms for multi-class imbalance learning either extends binary imbalance classifiers to multi-class data using the decomposition methods (e.g., One-vs-All), or adapt the intrinsic process in building the decision trees (e.g., the node splitting criterion for HDDT [8]) or adopt ensemble-based approaches to address the multi-class imbalance issue [3].

In this work, we first review recent advances in multi-class imbalance learning, where 10 major state-of-the-art methods [8–16] and another 7 of their variants will be addressed. In Table 1, we summarize our investigations of these methods, describe their characteristics and point out their strengths and weaknesses.

Moreover, we design a new multi-class imbalance classification algorithm, namely DECOC (Diversified Error Correcting Output Codes). It is a multiple-classifier system (i.e., ensemble learning method), which generates an ECOC (Error Correcting Output Codes) matrix for sparse schemes. ECOC builds a codeword for each class to obtain the largest distance between various classes, then transforms the multi-class into codewords and builds a dichotomy classifier for each column of transformed codeword as the class values. Considering the fact that the dichotomy classifiers contribute differently to the final prediction, DECOC assigns weights to the dichotomies and uses weighted distance for decoding, where the optimal weights will be obtained by minimizing the weighted loss in favor of the minority class(es).

We implement DECOC and the above-mentioned 17 state-of-the-art multi-class imbalance learning algorithms (among them, one multi-class imbalance learning algorithm is extended/adapted from a latest binary imbalance learning algorithm). We carry out experiments on 19 public multi-class imbalanced datasets to comprehensively compare their performance, using 4 different evaluation metrics for accuracy, including the overall accuracy (ACC), Geometric mean (G-mean) [17], F-measure [18] and the Area Under Curve (AUC) metrics [19]. We also

* Corresponding author.
E-mail address: chongsheng.zhang@yahoo.com (C. Zhang).

https://doi.org/10.1016/j.knosys.2018.05.037
Received 15 October 2017; Received in revised form 25 May 2018; Accepted 27 May 2018
Available online 04 June 2018

Table 1
Summary of state-of-the-art multi-class imbalance learning algorithms.

Algorithm	Characteristics	Refer.	Strengths	Weakness
Multi-IM	Combines the binary PRMs-IM algorithm with decomposition methods.	[9]	Easy to implement.	The performance is not effective.
imECOC	Improves the ECOC decomposition method to adapt to imbalanced data.	[10]	Can use conventional classifiers to classify multi-class imbalanced data.	Consumes large amount of computation time.
DOVO	Uses many different classification algorithms to train each binary imbalanced classifier.	[11]	Finds the best classifier for each binary data to combine the advantages of different classification algorithms.	Very computation demanding.
MC-HDDTs	Improves upon HDDT to adapt to multi-class data.	[8]	Inherently changes the node splitting criterion when building the decision tree, to adapt to multi-class data.	The performance is not effective.
AdaBoost.M1	Adapts AdaBoost to multi-class data.	[12]	Easy to implement.	The performance is not good enough on the AUC metric.
AdaC2.M1	Adapts AdaBoost to multi-class data, adopts Genetic Algorithm (GA) to derive the cost of wrong predictions on each class, when weighting the samples.	[13]	Considers the cost of wrong predictions on each class, when weighting the samples.	Genetic Algorithms are very time-consuming.
SAMME	Adapts AdaBoost to multi-class data, with new sample weighting functions.	[14]	Easy to implement.	Has moderate accuracy performance.
AdaBoost.NC	Adapts AdaBoost to multi-class data, adding a punishment parameter when weighting the samples.	[15]	Emphasizes the diversity of the binary datasets transformed from the multi-class data.	The performance is not good.
PIBoost	Each class is encoded in bits, to transform the multi-class classification problem into multiple binary classification problems.	[16]	Transforms multi-class classification problem to several binary imbalance learning problems.	Consumes large amount of computation time.

compare the running time efficiency of these methods. Extensive experiments demonstrate that DECOC is generally better than all of the other 17 algorithms on all the 4 accuracy measures.

The remainder of this paper is organized as follows. In Section 2, we briefly introduce 5 decomposition strategies which can extend binary classifiers to multi-class data, and review recent advances in multi-class imbalanced data classification. In Section 3, we introduce our new algorithm-DECOC. We present the experimental settings in Section 4 and compare the performance of the algorithms in Section 5. Finally, we conclude the paper in Section 6.

2. State-of-the-art on multi-class imbalance learning

2.1. From binary learners to multi-class classifiers: the decomposition methods

One-vs-All (OVA), also known as 'One-against-all', is a relatively simple decomposition strategy. For each class (category), it labels this class as a 'positive class' and all the other classes as 'negative classes', then trains a corresponding classification model. This way, multi-class classification is transformed into multiple binary classification problems. If the original data has c classes (categories), a total of c binary classifiers will be learnt.

The One-vs-One (OVO) decomposition strategy first selects a subset from the original data that only contains the instances for each pair of classes, then trains a binary classifier for each pair of classes, hence a total of c (c-1) / 2 binary classifiers will be obtained. In the prediction phase, all the c (c-1) / 2 binary classifiers will be used to predict a new instance, and their corresponding predictions will be combined using certain rules to make the final prediction.

The ECOC (Error Correcting Output Codes) decomposition method, proposed by Dietterich and Bakiri [20], uses the idea of error correction output coding to classify the multi-class data. ECOC can adopt different encoding and decoding methods. It first builds a codeword for each class to obtain the largest distance (such as Hamming distance) between various classes, thus transforms the classes of the multi-class data into c codewords (when using OVA as the encoding method). In the testing phase, it uses these c classifiers to respectively predict the test sample, then obtains a combined output.

The All-and-One (A&O) method, proposed in [21], combines the advantages of OVA and OVO to avoid their shortcomings. When predicting a new sample, A&O first uses OVA to get the top-2 prediction results (c_i, c_j), then adopts the OVO classifier previously trained for the pair of classes containing c_i and c_j to make the final prediction.

One-Against-Higher-Order (OAHO) [22] is a decomposition method specifically designed for imbalanced data. OAHO first sorts the class by the number of samples in descending order. Let the sorted classes being $\{C_1,\ C_2,\ \ldots, C_k\}$, with C_1 having the largest number of samples. Starting from C_1 until C_{k-1}, OAHO sequentially labels the current class as 'positive class' and all the rest classes with lower ranks as 'negative classes', then trains a binary classifier. Therefore, there will be k-1 binary classifiers in total. When predicting a new sample, the first classifier is used to predict the sample, if the prediction result is C_1, then outputs C_1 as the final result; otherwise, it switches to the second classifier to make the prediction, and so on, until the final prediction result is obtained.

2.2. Multi-IM

PRMs-IM [23] is a classification algorithm for binary imbalanced data. Let m be the ratio between the number of majority samples and that of the minority samples. PRMs-IM randomly divides the majority samples into m parts, next combines each part with all the minority instances, then trains a corresponding binary classifier. In the prediction phase, it uses weighted voting to ensemble the outputs of the m classifiers and makes the final prediction. Ghanem et al. [9] propose Multi-IM that combines A&O and PRMs-IM, where PRMs-IM is adopted to train the classifier for A&O. Besides A&O, in this work we will also combine the OVA, OVO and OAHO decomposition methods with PRMs-IM to further investigate the performance of PRMs-IM.

2.3. imECOC

The imECOC method proposed in [10] is an improved ECOC method for tackling the class imbalance problem with ordinary binary classifiers. imECOC includes the following techniques: (1) in each binary classifier, it simultaneously considers the between-class and the within-class imbalance; (2) in the prediction phase, it assigns different weights to different binary classifiers, then decodes it with weighted distance and finds the closest codeword and the corresponding class.

2.4. DOVO

All of the decomposition methods mentioned above decompose the original dataset and use the same classification algorithm to train a classifier. As to which algorithm should be chosen, the best algorithm for different datasets is usually different [24–26]. Even on the same dataset, the best algorithms for different distributed samples are different [27–28].

Inspired by the above observations, many researchers start to use a number of different classifiers to improve the classification accuracy, which is usually referred to as multi-expert classifier system (or multiple classifier system) [29–31]. The multiple classifier system is based on the diversity of training classifiers, thus it consists of a collection of different classifiers [32]. The diversity of classifiers can be obtained through a variety of strategies, for example, trying different classification algorithms or generating a few smaller datasets from the same dataset to train different classifiers.

For the OVO method, the multiple classifier system also applies. After decomposing the original dataset, each subset corresponds to different samples, so the best algorithm for different subsets may also be different. The Diversified One-against-One (DOVO) method proposed in [11] aims to find the best classification algorithm for each sub-problem when applying the OVO decomposition method to tackle the multi-class classification problems. DOVO combines a variety of classification algorithms with the OVO strategy to select samples for each pair of classes from the same multi-class dataset. Therefore, given the training dataset with c classes, the OVO method decomposes the original dataset into c (c-1) / 2 sub-datasets. For each sub-dataset, all the different classification algorithms will be used to train the corresponding classifiers. For the current sub-dataset, we respectively derive the error rate of each classifier, then selects the classifier with the lowest error rate as the strongest classifier, which will be used as the final classifier for the current sub-dataset. Finally, a total of c (c-1) / 2 strong classifiers will be obtained, which come from different classification algorithms. The prediction phase of DOVO is exactly the same as that of OVO, the predictions from all the classifiers are combined to make the final prediction. DOVO can achieve superior performance due to the fact that it always selects the best classification algorithm for each sub-dataset of the multi-class data, thus it comprehensively utilizes the strengths of different classification algorithms.

2.5. MC-HDDT

The Hellinger distance decision trees (HDDT) method proposed by Cieslak and Chawla in [33] is a classification algorithm based on decision trees for binary imbalanced data. When building a decision tree, the splitting criterion is very important, it determines how the data should be divided to achieve the best results. The splitting criterion used in the HDDT is the Hellinger distance defined as follows:

$$d_H(X_+,\ X_-) = \sqrt{\sum_{j=1}^{p}\left(\sqrt{\frac{|X_{+j}|}{|X_+|}} - \sqrt{\frac{|X_{-j}|}{|X_-|}}\right)^2} \tag{1}$$

where $|X_+|(|X_-|)$ represents the total number of positive (negative) samples, $|X_{+j}|$ $(|X_{-j}|)$ denotes the number of positive (negative) examples with the j-th value (of p distinct values) of the feature. For each node to be splitted, HDDT calculates the Hellinger distance for each attribute on the node, then splits the node using the feature with the maximum Hellinger distance.

Although decision trees inherently support multi-class classification, the Hellinger distance in [33] can only be used for binary imbalanced data, so [12] proposed the Multi-Class HDDT method, by successively taking one or a pair of classes as the positive class and the rest as negative class, when calculating the Hellinger distance for each feature. It next selects the maximum Hellinger value for this feature. Finally, it obtains the maximum Hellinger value for each feature, it will eventually select the feature with maximum Hellinger distance to split the node.

2.6. Variants of AdaBoost for multi-class imbalance learning

AdaBoost (Adaptive Boosting) [12,34] is a binary classification algorithm proposed by Freund and Schapire that integrates multiple weak classifiers to build a stronger classifier. AdaBoost only supports binary data in the beginning, but it was later extended to multi-class scenarios.

AdaBoost.M1 and SAMME (Stagewise Additive Modeling using a Multi-class Exponential loss function) have extended AdaBoost in both the update of samples' weights and the classifier combination strategy. The main difference between them is the method for updating the weights of the samples.

AdaC2.M1 proposed in [13] derives the best cost setting through the genetic algorithm (GA) method, then takes this cost setting into consideration in the subsequent boosting. Genetic algorithm is proposed by Holland in [35], it is based on natural selection and genetics of random search technology. GA can achieve excellent performance in finding the best parameters.

Since GA is very time consuming, in [15], the authors propose AdaBoost.NC which deprecates the GA algorithm, but emphasizes ensemble diversity during training, and exploits its good generalization performance to facilitate class imbalance learning [36].

PIBoost [16] combines binary weak-learners to separate groups of classes, and uses a margin-based exponential loss function to classify multi-class imbalanced data.

2.7. Other related works

The authors in [3] provide a broad review of imbalanced learning algorithms. They also develop a taxonomy for the application domains (e.g., financial management, emergency management) of imbalance learning. However, there is no empirical study on the performance of these algorithms in their work. In [37], the authors compare the performance of several data-level sampling solutions (i.e., apply resampling on the original dataset as a preprocessing step to reduce the negative effect caused by class imbalance) for imbalance learning in the context of churn prediction (to predict whether a customer is a churner and non-churner). They find that the impact of sampling methods depends on the used evaluation metric and is interrelated with the classifiers. However, the sampling solutions tested in their paper, such as SMOTE [4], RUS [38] and ADASYN [5], are restricted to binary imbalance learning algorithms.

The authors in [39] propose a new ensemble method, which first converts an imbalanced dataset into multiple balanced ones (using resampling methods such as SMOTE and RUS), then builds a number of

Algorithm 1
The DECOC algorithm.

```
Input: training set
$D = \{(x_1, y_1), \ldots,(x_N, y_N)\}$, $y = \{1, \ldots,k\}$; the $\lambda$
parameter in Eq. 7; a binary-class learning algorithm $L$.
Output: $\widehat{Y}$
%Training
1.    generate code matrix: $M \in \{-1, 0, +1\}^{k\times l}$ ;
2.    for $t = 1$ to $l$ do
3.        $D^t = \emptyset$
4.        for $i = 1$ to $n$ do
5.            if $M(y_i, t) \neq 0$ then
6.                $D^t = D^t \cup (x_i, M(y_i, t))$
7.            end if
8.        end for
9.        learn a classifier $f_t$:
            a. train candidate classifiers $f_{A_1,D^t}, \ldots f_{A_M,D^t}$, each of which is trained using a
          different algorithm
            b. calculate the validation error for each candidate classifier
            c. find $f_{A_{best},D^t}$ that corresponds to the minimum validation error
            d. $f_t \leftarrow f_{A_{best},D^t}$
10.       calculate the bit distance vector for the training set,
          $b(x_i, r)$, $\forall x_i \in D$, $\forall r = 1, \ldots,k$, according to Eq. 3–5
11.       obtain the optimal dichotomy weights $w$ according to Eq. 7
12.   end for
%Testing
13.   for a test instance $x$, calculate the bit instance vector$b(x, r)$, $\forall r = 1, \ldots,k$,
      according to Eq. 3–5
14.   Output: $\widehat{Y} = \text{argmin}_r w^T b(x, r)$ (Eq. 6)
```

classifiers on these multiple data with a specific classification algorithm. Finally, the classification results of these classifiers for new data are combined with a specific ensemble rule. Yet, their study is also restricted to binary-class imbalanced data.

The authors in [40] propose *HardEnsemble* which uses 50 oversampling-based classifiers and 100 undersampling based classifiers for imbalance learning, and compare its performance with a few classification methods for handling the class-imbalance problem. However, many recent advances in multi-class imbalance learning, e.g., DOVO, HDDT + ECOC, imECOC, SAMME, are not included in their experiments.

The work in [41] proposes to employ feature selection and resampling as preprocessors, then train multiple classifiers for multi-class imbalance learning.

In [42], the authors conduct an empirical study on the possibility of

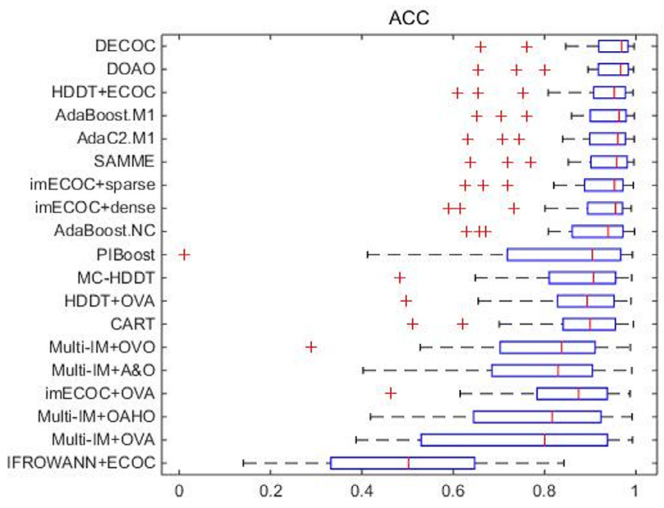

Fig. 1. The box plot of 16 methods on ACC.

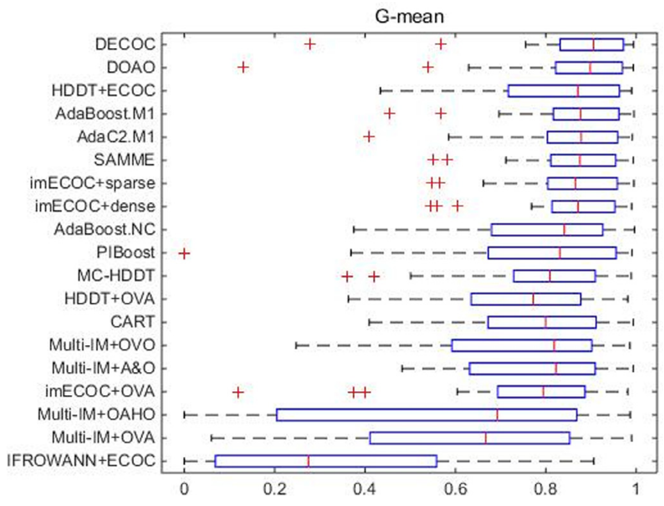

Fig. 2. The box plot of 16 methods on G-mean.

Table 2
Datasets in the experiments.

#	Dataset name	Source	#inst.	#feat.	#class	Ratio of each class
1	Satimage_data_set_indx_fixed	UCI	6435	36	6	0.2382;0.1092;0.2110;0.0973;0.1099;0.2343
2	Cardiotocography_ten_class_data_set_indx_fixed	UCI	2126	22	10	0.2723;0.1806;0.1561;0.1185;0.0926;0.050;0.0380;0.0338;0.0324;0.0249
3	page-blocks-5an-nn	UCI	5472	10	5	0.8978;0.0601;0.0051;0.0159;0.0210
4	IIF_intensity_all_features_data_set_indx_fixed	KEEL	600	57	3	0.3600;0.3250;0.3150
5	penbased-10an-nn	UCI	10,992	16	10	0.1040;0.1040;0.1041;0.0960;0.1041;0.0960;0.0961;0.1039;0.0960;0.0960
6	Wall_following_data_set_indx_fixed	UCI	5456	24	4	0.4041;0.3843;0.1513;0.0601
7	Wine_data_set_indx_fixed	UCI	178	13	3	0.3988;0.3314;0.2696
8	autos_data_set_indx_fixed	KEEL	156	25	5	0.3076 0.2948 0.1858 0.1282 0.08333
9	dermatology_data_set_indx_fixed	UCI	366	33	6	0.306;0.1967;0.1666;0.142;0.1338;0.0546
10	thyroid_data_set_indx_fixed	KEEL	720	21	3	0.925;0.0513;0.0236;
11	a1raw	KEEL	1747	19	5	0.0172;0.0933;0.3995;0.1093;0.3755
12	Optdigits_data_set_indx_fixed	UCI	5620	60	10	0.1017;0.1016;0.101;0.1007;0.1;0.0992;0.0992;0.0991;0.0985;0.0985
13	gas_batch1	UCI	445	128	6	0.2022;0.2202;0.1865;0.0674;0.1573;0.1663
14	cardiotocography	Openml	2126	35	3	0.7785;0.1388;0.0828
15	dataset30pageblocks	Openml	5473	10	5	0.8977;0.0601;0.0051;0.0161;0.021
16	dataset32pendigits	Openml	10,992	16	10	0.1040;0.1040;0.1041;0.0960;0.1041;0.0960;0.0961;0.1039;0.0960;0.0960
17	dataset41glass	Openml	214	9	6	0.3271;0.0794;0.0421;0.3551;0.1355;0.0607
18	GesturePhaseSegmentationProcessed	Openml	9873	32	5	0.2776;0.2124;0.2988;0.1011;0.1101
19	volcanoesa1	Openml	3252	3	5	0.9077;0.0209;0.0178;0.0264;0.0271

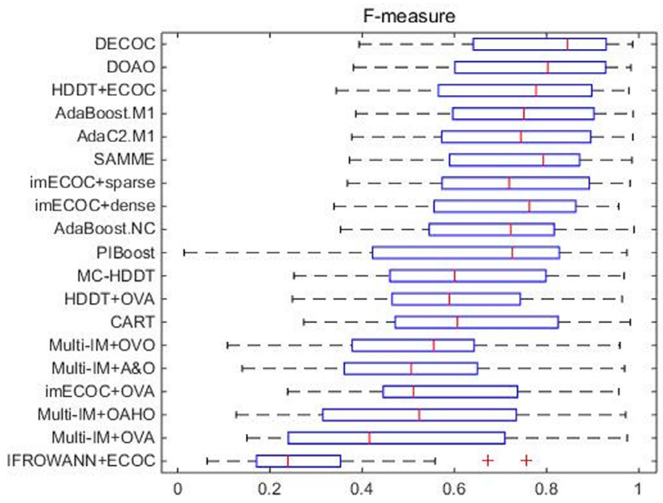

Fig. 3. The box plot of 16 methods on F-measure.

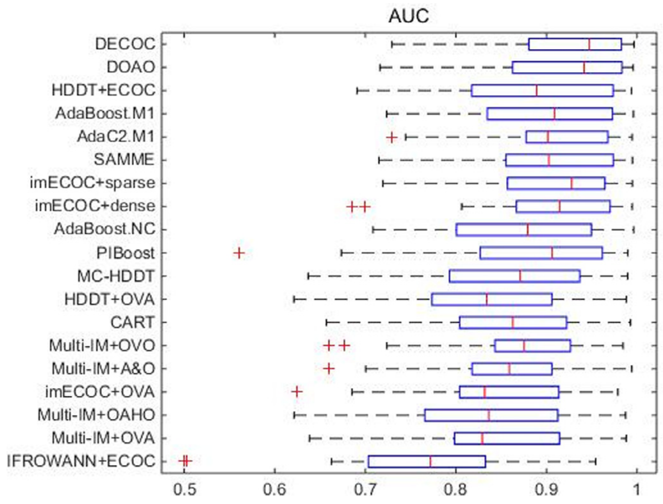

Fig. 4. The box plot of 16 methods on AUC.

combining the One-vs-One (OVO) decomposition strategy with binary ensemble learning approaches (such as SMOTE and RUS with a specific classifier AdaBoost) for multiclass imbalance classification problems. This work only investigated the OVO decomposition strategy (with binary imbalance classification algorithms) for multi-class imbalance learning, while recent algorithms that can inherently handle multi-class imbalance are not considered.

In [43], the authors present a study on oversampling techniques for multi-class imbalanced datasets. They have experimented with two multi-class imbalance learning algorithms, which are Static-SMOTE [44] and AdaBoost.NC. Yet, other multi-class imbalance learning algorithms, such as DOVO, HDDT + ECOC, imECOC, SAMME, are not included in their work.

Dynamic financial distress prediction (DFDP) is important for improving corporate financial risk management. However, earlier studies ignore the time weight of the samples when constructing ensemble FDP models. In [45], the authors propose two new DFDP approaches based on time weighting and Adaboost-based Support Vector Machines (SVM) ensemble. These two approaches consider the time weighting of samples when constructing Adaboost-based SVM ensemble and they are more suitable for DFDP in case of financial distress concept drift.

In [46], the authors build the decision rules by optimizing decision directed acyclic graph (ODDAG) with classical fuzzy decision trees, and ensemble the posterior probabilities of the binary classifiers from One-vs-One (OVO) or One-vs-All (OVA) decomposition strategies for tackling the multi-class imbalance classification problem.

The authors in [47] present a new DT ensemble model for imbalanced enterprise credit evaluation, based on the synthetic minority over-sampling technique (SMOTE) and the Bagging ensemble learning algorithm with differentiated sampling rates (DSR), which is named as DTE-SBD (Decision Tree Ensemble based on SMOTE, Bagging and DSR) in their work.

The work in [48] proposes a new hybrid method for preprocessing imbalanced data-sets through the construction of new samples, using the SMOTE together with the application of an editing technique based on the Rough Set Theory (which is named as SMOTE-RSB) and the lower approximation of a subset.

In [49], the authors introduce a classification algorithm called IFROWANN for binary imbalanced data. IFROWANN is based on fuzzy rough set theory and ordered weighted average aggregation. It designs different strategies to build a weight vector that take data imbalance into account. The authors validate their proposal with extensive experiments, showing that IFROWANN outperforms the other 13 state-of-the-art methods (including SMOTE-RSB) for imbalance learning. Since IFROWANN was originally designed for binary imbalanced data, in this work, we extend it to multi-class imbalanced data using the ECOC (sparse) decomposition strategy will be presented below in Section 3, to validate its performance on multi-class imbalanced data. The corresponding method is referred to as IFROWANN + ECOC hereafter (in the experiment section).

Besides, there are a few other interesting works in this field [50–53], since it develops very fast.

3. The DECOC approach for multi-class imbalance learning

ECOC (sparse, dense) is generally superior to OVO when combined with the same classification method, so we propose to replace the decomposition method in DOVO by ECOC (sparse) to achieve better accuracy. We assign different weights to dichotomies and use weighted distance for decoding, where the optimal weights are obtained by minimizing the weighed loss in favor of the minority classes. By combing the ideas of DOVO and the improved ECOC method for tackling the class imbalance problem (i.e., imECOC), we propose the DECOC method for imbalance data.

In [10], the following formula is used by imECOC to determine the importance of an example of class C_i in the t-th dichotomy:

$$c_i^t = \frac{\max(N_+^t\ ,\ N_-^t)}{|A_z^t|n_i} \tag{2}$$

where N_+^t / N_-^t is the number of positive / negative examples, and n_i is the number of all examples belongs C_i, while $|A_z^t|$ is the number of positive or negative classes in the t-th dichotomy. If C_i is the positive class, then $|A_z^t|$ represents the number of positive classes in the t-th dichotomy; otherwise, $|A_z^t|$ denotes the number of negative classes in the t-th dichotomy.

$$b_t(x,\ r) = \ d_{bit}(f_t(x),\ M(r,\ t)) \tag{3}$$

Formula (3) defines $b_t(x, r)$ as the t-th bit distance of example x to class C_r, it measures the distance between the dichotomy classifier f_t and class C_r's dichotomy code $M(r, t)$.

The bit distance function is defined as:

$$d_{bit}(u,\ v) = \ \frac{1 - uv}{2} \tag{4}$$

Let $b(x,r)$ be the bit distance vector of instance x and C_r:

$$b(x,\ \ r) = (\ b_1(x,\ r), \dots,\ \ b_l(x,\ r))^T \tag{5}$$

Let w be the dichotomy weight vector, then the weighted distance between a test instance x and C_r in the code space is $W^T b(x, r)$ and the class with the minimal distance is assigned to x:

Table 3
Comparisons between DECOC and the 8 selected methods on 4 evaluation metrics.

#dataset	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	SUM
(a) ACC																				
DOAO	1	0	1	1	0	1	1	1	0	0	1	0	1	1	1	1	1	1	1	14
HDDT + ECOC	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	18
AdaBoost.M1	1	0	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	16
AdaC2.M1	1	0	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	16
SAMME	1	0	1	1	1	1	1	0	1	1	1	1	1	0	1	1	0	1	1	15
imECOC + sparse	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	18
imECOC + dense	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	18
AdaBoost.NC	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	18
(b) G-mean																				
DOAO	1	0	1	1	0	1	1	1	0	0	1	0	1	1	1	1	1	1	1	14
HDDT + ECOC	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	16
AdaBoost.M1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	0	1	0	15
AdaC2.M1	1	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	0	0	1	14
SAMME	1	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	16
imECOC + sparse	1	1	0	1	1	0	1	1	1	0	1	1	1	1	1	1	0	1	1	15
imECOC + dense	1	0	0	1	1	1	1	1	1	0	1	1	1	1	0	1	0	1	1	14
AdaBoost.NC	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	17
(c) F-measure																				
DOAO	1	0	1	1	0	1	1	1	0	0	1	0	1	1	1	1	1	1	1	14
HDDT + ECOC	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	18
AdaBoost.M1	1	0	1	1	1	0	1	0	1	1	1	1	1	1	1	1	0	1	1	15
AdaC2.M1	1	0	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	16
SAMME	1	0	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	16
imECOC + sparse	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	18
imECOC + dense	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	18
AdaBoost.NC	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	18
(d) AUC																				
DOAO	1	0	1	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	16
HDDT + ECOC	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	17
AdaBoost.M1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	16
AdaC2.M1	1	0	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	16
SAMME	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	17
imECOC + sparse	1	1	0	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	16
imECOC + dense	1	0	0	1	1	1	1	1	1	0	1	1	1	1	0	1	0	1	0	13
AdaBoost.NC	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	18

$$\widehat{Y} = \text{argmin}_r \, w^T b(x, r) \tag{6}$$

The optimal dichotomy weights w is derived as follows:

$$\begin{aligned} &\min_{w,\varepsilon_i} \; \|w\|^2 + \lambda \textstyle\sum_{i=1}^{n} \gamma_i \xi_i \\ &\text{s.t.} \;\; w^T b(x_i, r) - w^T b(x_i, y_i) \geq \Delta(y_i, r) - \xi_i, \\ &\forall \; r = 1, \ldots, k \quad \xi_i \geq 0 \end{aligned} \tag{7}$$

where λ is the penalty parameter, γ_i is the weight of x_i, which is proportional to the inverse of its class size, ξ_i is slack variable.

The pseudo code of DECOC is shown in Algorithm 1.

In Algorithm 1, steps 1–14 (except step 9) use the idea in imECOC. Step 1 generates the coding matrix, steps 2–12 sequentially train l classifiers and calculate the weight of each classifier, where the optimal weights are obtained by minimizing the weighed loss in favor of the minority classes. Step 9 contains four sub-steps (a–d), which uses an ensemble learning approach. It exhaustively tries each classification algorithm, then selects the best classification method (the one with the lowest validation error) for each sub-dataset. Steps 13 and 14 occur in the testing stages, $\widehat{Y}$ is the final prediction.

4. Experimental settings

4.1. Evaluation metrics for imbalance data

For imbalanced data, it is not enough to only evaluate the overall accuracy (ACC). G-mean [17], AUC [19] and F-measure [18] are also commonly used measures to evaluate the performance of imbalance learning algorithms. G-mean is the geometric mean of the prediction accuracy rate on each class. F-measure is the weighted harmonic mean of precision and recall. The AUC value is the area below the receiver operating characteristic (ROC) curve, which is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.

4.2. Datasets

The experiments in this paper use a total of 19 datasets from UCI, KEEL and Openml, and each experiment is carried out using 5-fold cross validation. The number of instances varies from 156 to 10,992, while the number of features ranges from 3 to 128, and the number of classes is between 3 and 10. Table 2 gives details for each dataset. For example, the 'Wine_data_set_indx_fixed' dataset from UCI is the result of chemical analysis of wines from three different types grown in the same area of Italy. The analysis identifies the amount of the 13 components. The 'thyroid_data_set_indx_fixed' dataset describes the main features of the thyroid and their properties. It contains 21 features, including age, gender, pregnancy and so on. The task of this dataset is to check whether a given patient is normal or suffering from hyperthyroidism / hypothyroidism.

5. Results and analysis

Besides DECOC, we have implemented 9 different multi-class imbalance learning algorithms and 7 variants stated above; there are in total 16 state-of-the-art methods. It should be noted that imECOC + dense / imECOC + sparse / imECOC + OVA correspond to different codewords to be used in imECOC. We also report the classification results from CART, as the baseline (reference). DOVO and DECOC test 8 different classifiers, including SVM, KNN, Logistic

Table 4
Dunn's multiple comparisons test for the ACC of 19 algorithms on 19 datasets (confidence level: 95%).

	DOAO	HDDT + ECOC	AdaBoost.M1	AdaC2.-M1	SAMME	imECOC-+ sparse	imECOC-+ dense	AdaBoost.NC	PIBoost	MC-HDDT	HDDT + OVA	CART	Multi-IM + OVO	Multi-IM + A&O	im-ECOC + OVA	Multi-IM + OA-HO	Multi-IM + OVA	IFROW-ANN + -ECOC
DECOC	**25.00**	82.0	43.0	60.5	41.0	103.5	133.0	141.5	137.5	193.0	187.5	172.5	224.0	244.5	250.0	252.0	240.5	309.5
DOAO		57.0	**18.0**	**35.5**	**16.0**	78.5	108.0	116.5	112.5	168.0	162.5	147.5	199.0	219.5	225.0	227.0	215.5	284.5
HDDT + ECOC			**−39.0**	**−21.5**	**−41.0**	**21.5**	51.0	59.5	55.5	111.0	105.5	90.5	142.0	162.5	168.0	170.0	158.5	227.5
AdaBoost.M1				**17.5**	**−2.0**	60.5	90.0	98.5	94.5	150.0	144.5	129.5	181.0	201.5	207.0	209.0	197.5	266.5
AdaC2.M1					**−19.5**	43.0	72.5	81.0	77.0	132.5	127.0	112.0	163.5	184.0	189.5	191.5	180.0	249.0
SAMME						62.5	92.0	100.5	96.5	152.0	146.5	131.5	183.0	203.5	209.0	211.0	199.5	268.5
imECOC + sparse							**29.5**	38.0	**34.0**	89.5	84.0	69.0	120.5	141.0	146.5	148.5	137.0	206.0
imECOC + dense								**8.5**	**4.5**	60.0	54.5	39.5	91.0	111.5	117.0	119.0	107.5	176.5
AdaBoost.NC									**−4.0**	51.5	46.0	**31.0**	82.5	103.0	108.5	110.5	99.0	168.0
PIBoost										55.5	50.0	**35.0**	86.5	107.0	112.5	114.5	103.0	172.0
MC-HDDT											**−5.5**	**−20.5**	**31.0**	51.5	57.0	59.0	47.5	116.5
HDDT + OVA												**−15.0**	**36.5**	57.0	62.5	64.5	53.0	122.0
CART													51.5	72.0	77.5	79.5	68.0	137.0
Multi-IM + OVO														**20.5**	**26.0**	**28.0**	**16.5**	85.5
Multi-IM + A&O															**5.5**	**7.5**	**−4.0**	65.0
im-ECOC + OVA																**2.0**	**−9.5**	59.5
Multi-IM + OAHO																	**−11.5**	57.5
Multi-IM + OVA																		69.0

Bold values correspond to non-statistically significant comparisons (absolute rank sum differences that are < 37.6242).

Table 5
Dunn's multiple comparison test for the G-mean of 19 algorithms on 19 datasets (confidence level: 95%).

	DOAO	HDDT + -ECOC	AdaBoost.M1	AdaC2.-M1	SAMME	imECOC-+ sparse	imECOC-+ dense	AdaBoost.NC	PIBoost	MC-HDDT	HDDT + -OVA	CART	Multi-IM + OVO	Multi-IM + A&O	im-ECOC + -OVA	Multi-IM + OA-HO	Multi-IM + OVA	IFROW-ANN + -ECOC
DECOC	**33.5**	84.0	**36.0**	**34.0**	50.5	53.0	77.0	144.0	108.0	146.0	188.0	160.5	136.0	133.0	214.0	201.5	209.0	272.0
DOAO		50.5	**2.5**	**0.5**	**17.0**	**19.5**	43.5	110.5	74.5	112.5	154.5	127.0	102.5	99.5	180.5	168.0	175.5	238.5
HDDT + ECOC			**−48.0**	**−50.0**	**−33.5**	**−31.0**	**−7.0**	60.0	**24.0**	62.0	104.0	76.5	52.0	49.0	130.0	117.5	125.0	188.0
AdaBoost.M1				**−2.0**	**14.5**	**17.0**	41.0	108.0	72.0	110.0	152.0	124.5	100.0	97.0	178.0	165.5	173.0	236.0
AdaC2.M1					**16.5**	**19.0**	43.0	110.0	74.0	112.0	154.0	126.5	102.0	99.0	180.0	167.5	175.0	238.0
SAMME						**2.5**	**26.5**	93.5	57.5	95.5	137.5	110.0	85.5	82.5	163.5	151.0	158.5	221.5
imECOC + sparse							**24.0**	91.0	55.0	93.0	135.0	107.5	83.0	80.0	161.0	148.5	156.0	219.0
imECOC + dense								67.0	**31.0**	69.0	111.0	83.5	59.0	56.0	137.0	124.5	132.0	195.0
AdaBoost.NC									**−36.0**	**2.0**	44.0	**16.5**	**−8.0**	**−11.0**	70.0	57.5	65.0	128.0
PIBoost										38.0	80.0	52.5	**28.0**	**25.0**	106.0	93.5	101.0	164.0
MC-HDDT											42.0	**14.5**	**−10.0**	**−13.0**	68.0	55.5	63.0	126.0
HDDT + OVA												**−27.5**	**−52.0**	**−55.0**	**26.0**	**13.5**	**21.0**	84.0
CART													**−24.5**	**−27.5**	53.5	41.0	48.5	111.5
Multi-IM + OVO														**−3.0**	78.0	65.5	73.0	136.0
Multi-IM + A&O															81.0	68.5	76.0	139.0
im-ECOC + OVA																**−12.5**	**−5.0**	58.0
Multi-IM + OAHO																	**7.5**	70.5
Multi-IM + OVA																		63.0

Bold values correspond to non-statistically significant comparisons (absolute rank sum differences that are < 37.6242).

Table 6
Dunn's multiple comparison test for the F-measure of 19 algorithms on 19 datasets (confidence level: 95%).

	DOAO	HDDT + -ECOC	AdaBoos-t.M1	AdaC2.-M1	SAMME	imECOC-+ sparse	imECOC-+ dense	AdaBoost.NC	PIBoost	MC-HDDT	HDDT + -OVA	CART	Multi-IM + OVO	Multi-IM + A&O	im-ECOC + -OVA	Multi-IM + OA-HO	Multi-IM + OVA	IFROW-ANN + -ECOC
DECOC	**22.5**	93.5	46.5	61.0	49.0	97.5	125.5	147.5	131.5	187.5	201.0	173.5	217.0	244.0	246.5	225.5	243.5	308.5
DOAO		71.0	**24.0**	38.5	**26.5**	75.0	103.0	125.0	109.0	165.0	178.5	151.0	194.5	221.5	224.0	203.0	221.0	286.0
HDDT + ECOC			**−47.0**	**−32.5**	**−44.5**	**4.0**	**32.0**	54.0	38.0	94.0	107.5	80.0	123.5	150.5	153.0	132.0	150.0	215.0
AdaBoost.M1				**14.5**	**2.5**	51.0	79.0	101.0	85.0	141.0	154.5	127.0	170.5	197.5	200.0	179.0	197.0	262.0
AdaC2.M1					**−12.0**	**36.5**	64.5	86.5	70.5	126.5	140.0	112.5	156.0	183.0	185.5	164.5	182.5	247.5
SAMME						48.5	76.5	98.5	82.5	138.5	152.0	124.5	168.0	195.0	197.5	176.5	194.5	259.5
imECOC + sparse							**28.0**	50.0	**34.0**	90.0	103.5	76.0	119.5	146.5	149.0	128.0	146.0	211.0
imECOC + dense								**22.0**	**6.0**	62.0	75.5	48.0	91.5	118.5	121.0	100.0	118.0	183.0
AdaBoost.NC									**−16.0**	40.0	53.5	**26.0**	69.5	96.5	99.0	78.0	96.0	161.0
PIBoost										56.0	69.5	42.0	85.5	112.5	115.0	94.0	112.0	177.0
MC-HDDT											**13.5**	**−14.0**	**29.5**	56.5	59.0	38.0	56.0	121.0
HDDT + OVA												**−27.5**	**16.0**	43.0	45.5	**24.5**	42.5	107.5
CART													43.5	70.5	73.0	52.0	70.0	135.0
Multi-IM + OVO														**27.0**	**29.5**	**8.5**	**26.5**	91.5
Multi-IM + A&O															**2.5**	**−18.5**	**−0.5**	64.5
im-ECOC + OVA																**−21.0**	**−3.0**	62.0
Multi-IM + OAHO																	**18.0**	83.0
Multi-IM + OVA																		65.0

Bold values correspond to non-statistically significant comparisons (absolute rank sum differences that are <37.6242).

Regression, C4.5, AdaBoost, Random Forests, CART and Multilayer perceptron (a description of these classifiers can be found in [54]). Whenever a base classifier is needed, all the relevant algorithms in the experiments adopt CART, due to three reasons: (i) CART inherently supports multi-class classification; (ii) it supports weighted samples in the decision trees; (iii) unlike SVM, CART does not need parameter-tuning. All the algorithms in this work are implemented in *Matlab*.

As mentioned earlier in Subsection 2.7, we also include a new and effective binary imbalance learning algorithm IFROWANN (*AV-W6*). We obtained the implementation of IFROWANN from the original authors in [49]. We then translated their codes from *Java* to *Matlab* in order to systematically compare the running time efficiency of IFROWANN with the other algorithms. We have also validated the correctness of our implementation of IFROWANN, by comparing its results with the corresponding outputs of the original implementation provided by the authors [49]. In this work, we combine IFROWANN with the decomposition method ECOC (namely IFROWANN + ECOC) to validate its performance on multi-class imbalanced data.

In total, we report the results of 19 algorithms, including DECOC, CART which is used as the base learner, and the 17 state-of-the-art algorithms described above.

In the appendix, Tables A1–A4 are the detailed accuracy results of each algorithm calculated in terms of ACC, G-mean, F-measure and AUC, respectively. Table A5 reports the running time of different methods (in seconds).

5.1. Comparing the performance of the multi-class imbalance classification algorithms

We first investigate the performance of these 19 methods, using 4 different accuracy measures and the time efficiency measure. In Figs. 1–4, we respectively depict the ACC, G-mean, F-measure and AUC performance of each method on all 19 datasets using box plots. The symbol ' + ' in the box plots refers to an abnormal performance on a dataset; one such symbol represents a single dataset.

From Figs. 1–4, we have the following observations:

(1) On all the 4 measures, DECOC performs the best.
(2) Combining multiple classification algorithms is usually more effective than using a single classification method, as demonstrated by DECOC and DOVO.
(3) When the same classification method combines with different decomposition methods, the more the number of the classifiers, the better performance to be obtained. ECOC (sparse, dense) needs to train $2^n - 1$ classifiers, OVO trains $n(n-1)/2$ classifiers, OVA generates n classifiers, while A&O needs to train $n(n-1)/2 + n$ classifiers, and OAHO trains $n - 1$ classifiers. Since OVA and OAHO train significantly less classifiers than the other decomposition algorithms, they (OVA and OAHO based methods) yield worse accuracy results than the same methods but with the ECOC decomposition strategy.

It should be noted that IFROWANN has proven to be among the top performers in binary imbalance learning. The performance of the multi-class version of this algorithm, i.e., IFROWANN + ECOC, will also be reported.

5.2. Pairwise comparisons between DECOC and state-of-the-art methods

Here, we conduct pair-wise comparisons between DECOC and state-of-the-art to further validate the performance of DECOC. Out of the 19 methods, we select the top-9 performers in accuracy. Table 3 is the comparison between DECOC and the other 8 methods on the 4 evaluation metrics, where the symbol ' 1 ' represents DECOC is better than or equal to the performance of the method to be compared, while ' 0 ' denotes that DECOC is worse than the method in comparison. For

Table 7
Dunn's multiple comparison test for the AUC of 19 algorithms on 19 datasets (confidence level: 95%).

	DOAO	HDDT + -ECOC	AdaBoost.M1	AdaC2.-M1	SAMME	imECOC-+ sparse	imECOC-+ dense	AdaBoost.NC	PIBoost	MC-HDDT	HDDT + -OVA	CART	Multi-IM + OVO	Multi-IM + A&O	im-ECOC + -OVA	Multi-IM + OA-HO	Multi-IM + OVA	IFROW-ANN + -ECOC
DECOC	41.0	106.5	47.0	54.0	68.5	74.5	100.5	153.5	95.5	150.0	222.5	175.0	148.0	145.0	212.5	177.5	188.5	262.5
DOAO		65.5	**6.0**	**13.0**	**27.5**	**33.5**	59.5	112.5	54.5	109.0	181.5	134.0	107.0	104.0	171.5	136.5	147.5	221.5
HDDT + ECOC			**−59.5**	**−52.5**	**−38.0**	**−32.0**	**−6.0**	47.0	**−11.0**	43.5	116.0	68.5	41.5	38.5	106.0	71.0	82.0	156.0
AdaBoost.M1				**7.0**	**21.5**	**27.5**	53.5	106.5	48.5	103.0	175.5	128.0	101.0	98.0	165.5	130.5	141.5	215.5
AdaC2.M1					**14.5**	**20.5**	46.5	99.5	41.5	96.0	168.5	121.0	94.0	91.0	158.5	123.5	134.5	208.5
SAMME						**6.0**	**32.0**	85.0	**27.0**	81.5	154.0	106.5	79.5	76.5	144.0	109.0	120.0	194.0
imECOC + sparse							**26.0**	79.0	**21.0**	75.5	148.0	100.5	73.5	70.5	138.0	103.0	114.0	188.0
imECOC + dense								53.0	**−5.0**	49.5	122.0	74.5	47.5	44.5	112.0	77.0	88.0	162.0
AdaBoost.NC									**−58.0**	**−3.5**	69.0	**21.5**	**−5.5**	**−8.5**	59.0	**24.0**	**35.0**	109.0
PIBoost										54.5	127.0	79.5	52.5	49.5	117.0	82.0	93.0	167.0
MC-HDDT											72.5	**25.0**	**−2.0**	**−5.0**	62.5	**27.5**	38.5	112.5
HDDT + OVA												**−47.5**	**−74.5**	**−77.5**	**−10.0**	**−45.0**	**−34.0**	40.0
CART													**−27.0**	**−30.0**	**37.5**	**2.5**	**13.5**	87.5
Multi-IM + OVO														**−3.0**	64.5	**29.5**	40.5	114.5
Multi-IM + A&O															67.5	**32.5**	43.5	117.5
im-ECOC + OVA																**−35.0**	**−24.0**	50.0
Multi-IM + OAHO																	**11.0**	85.0
Multi-IM + OVA																		74.0

Bold values correspond to non-statistically significant comparisons (absolute rank sum differences that are < 37.6242).

example, in Table 3a, the first symbol ' 1 ' in the first row represents the ACC result of DECOC on the first dataset is better than that of DOVO, whereas the ' 0 ' denote the opposite situation.

Before we compare the performance of these methods, we notice that, on the 19 datasets, there is no constant winner that can always beat all the other algorithms. Nevertheless, the overall performance of DECOC is superior to the other methods, which we will address below.

In Table 3a, we compare the accuracy performance of DECOC and the other 8 selected methods on the ACC metric. We observe that, DECOC is better than DOAO on 14 datasets, but it is worse than DOAO on the rest 5 datasets, whereas DECOC is better than HDDT + ECOC on 18 out of the 19 datasets. It is also better than AdaBoost.M1 on 16 datasets, and it also outperforms AdaC2.M1 on 16 datasets. Similarly, we can compare the performance of DECOC with the rest methods. Overall, we observe that DECOC generally outperforms the other 8 methods on ACC.

In Table 3b, we compare the G-mean accuracy performance of DECOC with the 8 selected methods on all the 19 datasets. Again, DECOC outperforms DOAO on 14 datasets, but it loses on 5 datasets; DECOC is better than HDDT + ECOC on 16 datasets, it also outperforms AdaBoost.M1 on 15 datasets. Similarly, we can compare the performance of DECOC with the other methods. Therefore, DECOC outperforms the other methods on the G-mean metric.

In Table 3c, we compare the F-measure performance of DECOC and the 8 methods. Similar observations still hold on F-measure as in the case of the G-mean metric. For instance, DECOC is better than DOAO on 14 datasets, and it is better than HDDT + ECOC on 18 datasets. It still outperforms imECOC + sparse on 18 datasets. Thus, the overall performance of DECOC is superior to the other methods in comparison.

In Table 3d, we compare the AUC performance of DECOC and the 8 methods on all 19 datasets. We still have similar observations as the above-mentioned metrics.

In summary, DECOC outperforms the other 8 top performers on all the four accuracy metrics, i.e., ACC, G-mean, F-measure and AUC.

Dunn's test is a multiple comparison test that controls the family-wise error rate [55]. Dunn's test can be used as a post-hoc analysis to Friedman test, by comparing pair-wise the difference in the sum of ranks between two classifiers with the expected average difference (based on the number of groups and their size) [56]. The results of our Dunn's test are listed in Tables 4–7.

Table 4 presents Dunn's multiple comparison test for the ACC of 19 algorithms on the 19 datasets (confidence level: 95%). We observe that, the difference between DECOC and DOAO is not statistically significant, while the difference between DECOC and other 17 algorithms is significant. Therefore, DECOC is statistically significantly better than the other algorithms (except DOAO) in terms of ACC.

Table 5 reports Dunn's multiple comparison test for the G-mean of 19 algorithms on the same 19 datasets (confidence level: 95%). We observe that, the difference between DECOC and DOAO/AdaBoost.M1/AdaC2.M1 is not statistically significant, while the difference between DECOC and each of the rest 15 algorithms is significant. Table 6 is Dunn's multiple comparison test for the F-measure of 19 algorithms on the 19 datasets. We observe that, except DOAO, the differences between DECOC and the other 17 algorithms are significant.

Table 7 is Dunn's multiple comparisons test for the AUC of 19 algorithms on 19 datasets (confidence level: 95%). We observe that, the difference between DECOC and all other 18 algorithms is significant.

In summary, in terms of AUC, DECOC is statically the best algorithm for multi-class imbalance learning, while in terms of both ACC and F-measure, DECOC statically outperforms all the other algorithms, except DOVO. Finally, in terms of G-mean, DECOC generally outperforms the other algorithms, but the differences between DECOC and DOAO/AdaBoost.M1/AdaC2.M1 are not statistically significant.

It should be noted that, IFROWANN + ECOC achieves good AUC performances in a few datasets (e.g., *cardiotocography* and *dataset30pageblocks*). However, it does not show outstanding performance on the other three accuracy metrics. This can be explained by the fact that IFROWANN overly favors the minority class inherently. Another reason is that, in our tests we only adopt the generally good parameters ($AV + W6$) for IFROWANN, without tuning its parameters. This is due to the use of ECOC, there are already many combinations for IFROWANN + ECOC which is already very computation-demanding (as can be seen from the experimental results). Nevertheless, as instructed by the authors, if we try different weighting strategies (*W4, W5, W6*) and indiscernibility relation (*TL, AV, MIN*), the performance of IFROWANN + ECOC will be better. But due to the fact that IFROWANN + ECOC is already very time-consuming (as can be seen in Table A5), while the computation time will increase around 8 times if we try out different parameters, so we only adopt the generally good parameters $AV + W6$.

In Table A5 (in the Appendix), we report the running time efficiency of all the different methods. It is clear that DECOC is very time-consuming, since it needs to train a great many heterogeneous classification models.

6. Conclusions

In this paper, we review the recent advances in multi-class imbalance learning. Moreover, we design a new multi-class imbalance classification algorithm, namely DECOC. We carry out experiments on 19 datasets to empirically study the performance of DECOC in comparison with 17 state-of-the-art methods for multi-class imbalance learning as well as the base learner CART. We show that DECOC achieves the best overall performance, in terms of ACC, AUC, G-mean and F-measure. It is clear that the weak point of DECOC is that it is very time-consuming. But due to its excellent performance, DECOC should be considered in applications where classification accuracy is the main focus, whereas time efficiency requirements can be tolerated. In future work, we will try to use parallel methods or Spark-based solutions to significantly improve its efficiency. We will also include IFROWANN as the base learner for DECOC to further improve its accuracy performance. This study should also have important reference value for researchers and engineers in the imbalance learning domain.

Acknowledgments

This work is partially funded by the National Science Foundation of China (NSFC) under grant no. 41401466. It is also supported by the National Key Technology R&D Program of China under grant no. 2015BAK01B06, as well as Henan University under grant no. xxjc20140005. We greatly acknowledge the contributions of the anonymous reviewers for their valuable suggestions and help. We deeply appreciate the great help from Prof. Herrera (University of Granada, Spain), Prof. Ramentol (Universidad de Camagüey “Ignacio Agramonte Loynaz”, Cuba) and Dr. Sarah Vluymans for providing the source codes of their IFROWANN algorithm and helping us set up the corresponding running environment. We also thank our colleague Dr. George Almpanidis for helping us with the writing of this paper, the *Openml* datasets and Dunn's Test.

Appendix

Table A1
The results of different algorithms on ACC.

#dataset	DECOC	DOAO	HDDT + - ECOC	AdaBoost.M1	AdaC2.-M1	SAMME	imECOC- + sparse	imECOC- + dense	AdaBoost.NC	PIBoost	MC-HDDT	HDDT- + OVA	CART	Multi-IM + OVO	Multi-IM + A&O	im-ECOC + - OVA	Multi-IM + OAHO	Multi-IM + OVA	IFROW-ANN + - ECOC
1	**0.9148**	0.9136	0.9091	0.9138	0.9085	0.9102	0.9004	0.9007	0.8977	0.9046	0.8468	0.8368	0.8528	0.8558	0.8401	0.8255	0.8168	0.8124	0.4587
2	0.8946	0.8951	**0.9073**	0.8956	0.8956	0.8989	0.8834	0.8984	0.8481	0.8603	0.8227	0.8377	0.8500	0.8443	0.8312	0.8015	0.7855	0.8006	0.2079
3	**0.9687**	0.9667	0.9611	0.9633	0.9602	0.9655	0.9600	0.9609	0.9534	0.5715	0.9521	0.9539	0.9485	0.9114	0.8971	0.9377	0.9097	0.9289	0.5601
4	**0.9317**	**0.9317**	0.6550	0.7050	0.7067	0.7200	0.6667	0.6150	0.6717	0.7150	0.6483	0.6550	0.6217	0.6500	0.6283	0.6150	0.6717	0.6517	0.3150
5	0.9529	**0.9531**	0.9464	0.9517	0.9481	0.9504	0.9249	0.9440	0.9285	0.9336	0.8060	0.8254	0.8377	0.6892	0.6665	0.8027	0.4204	0.5286	0.3788
6	0.9962	0.9951	0.9938	0.9965	0.9963	0.9958	0.9945	0.9870	**0.9971**	0.9925	0.9907	0.9894	0.9947	0.9877	0.9908	0.9872	0.9918	0.9927	0.2265
7	**0.9608**	0.9492	0.9324	0.9213	0.9606	0.9098	0.9384	0.8930	0.9041	**0.9608**	0.9270	0.9324	0.9159	0.9275	0.9275	0.8930	0.9157	0.9494	0.8259
8	0.8466	0.8014	0.8079	**0.8589**	0.8395	0.8522	0.8204	0.8012	0.8083	**0.8589**	0.7758	0.7754	0.7883	0.7883	0.7625	0.6796	0.7569	0.7377	0.5452
9	0.9808	**0.9836**	0.9781	0.9726	0.9562	0.9617	0.9727	0.9617	0.9454	0.9753	0.9562	0.9043	0.9535	0.9590	0.9562	0.9016	0.9261	0.9207	0.8030
10	0.9819	0.9833	0.9750	0.9708	0.9778	0.9806	**0.9847**	0.9764	0.9750	0.9681	0.9722	0.9722	0.9806	0.9583	0.9583	0.9764	0.9694	0.9764	0.6514
11	**0.9748**	0.9731	0.9508	0.9657	0.9662	0.9582	0.9531	0.9553	0.9393	0.9422	0.9073	0.8872	0.8827	0.8374	0.8500	0.8747	0.8849	0.5341	0.3166
12	0.9833	**0.9851**	0.9774	0.9806	0.9801	0.9811	0.9658	0.9708	0.9544	0.9633	0.8829	0.8931	0.8996	0.8963	0.7393	0.8712	0.4190	0.4827	0.4966
13	**0.9640**	0.9303	0.9528	0.9483	0.9169	0.9416	0.9146	0.9213	0.9101	0.8989	0.9011	0.8674	0.8517	0.8247	0.7753	0.7775	0.6742	0.4112	0.5955
14	0.9868	0.9859	0.9850	0.9864	**0.9878**	**0.9878**	0.9798	0.9741	0.9821	0.4123	0.9845	0.9859	0.9835	0.8095	0.8297	0.9741	0.9746	0.9751	0.8424
15	**0.9761**	0.9733	0.9688	0.9704	0.9647	0.9688	0.9684	0.9684	0.9629	0.0111	0.9635	0.9633	0.9644	0.9094	0.9075	0.9499	0.9393	0.9406	0.7471
16	**0.9938**	0.9936	0.9896	0.9917	0.9910	0.9915	0.9843	0.9899	0.9824	0.9815	0.9540	0.9393	0.9557	0.7441	0.7815	0.9361	0.4684	0.6284	0.5023
17	0.7616	0.7382	0.7520	0.7616	0.7432	**0.7710**	0.7198	0.7336	0.6586	0.7291	0.6583	0.6772	0.7007	0.6633	0.6217	0.6212	0.6355	0.3872	0.4862
18	**0.6606**	0.6552	0.6104	0.6516	0.6323	0.6379	0.6264	0.5902	0.6285	0.6115	0.4837	0.4957	0.5104	0.5276	0.5122	0.4645	0.4663	0.4695	0.1405
19	**0.9828**	0.9815	0.9674	0.9822	0.9748	0.9815	0.9674	0.9705	0.9745	0.9803	0.9336	0.9477	0.9330	0.2881	0.4028	0.9317	0.8736	0.8739	0.6344
AVG	**0.9323**	0.9257	0.9063	0.9151	0.9109	0.9139	0.9013	0.8954	0.8906	0.8037	0.8614	0.8600	0.8645	0.7933	0.7831	0.8327	0.7632	0.7369	0.5123

Table A2
The results of different algorithms on G-mean.

#dataset	DECOC	DOAO	HDDT+-ECOC	AdaBoost.M1	AdaC2.-M1	SAMME	imECOC+sparse	imECOC+dense	AdaBoost.NC	PIBoost	MC-HDDT	HDDT+-OVA	CART	Multi-IM +OVO	Multi-IM +A&O	imECOC+OVA	Multi-IM +OAHO	Multi-IM +OVA	IFROW-ANN +ECOC
1	**0.8803**	0.8740	0.8710	0.8769	0.8790	0.8753	0.8655	0.8713	0.8600	0.8639	0.8092	0.7720	0.8097	0.8317	0.8363	0.7699	0.7843	0.7786	0.2778
2	0.8306	0.8390	0.8406	0.8276	**0.8617**	0.8354	0.8262	0.8605	0.7413	0.8032	0.7280	0.6908	0.7687	0.8135	0.8223	0.6906	0.6922	0.7079	0.0000
3	0.7555	0.7442	0.6992	0.7060	0.7623	0.7292	0.8083	0.8092	0.6232	0.5026	0.7316	0.5965	0.6532	**0.8361**	**0.8361**	0.7350	0.0000	0.6956	0.6300
4	**0.9300**	**0.9300**	0.6160	0.6966	0.6997	0.7116	0.6619	0.6038	0.6594	0.7046	0.6328	0.6160	0.6144	0.6399	0.6158	0.6038	0.6639	0.6244	0.0000
5	0.9522	**0.9530**	0.9460	0.9513	0.9477	0.9501	0.9232	0.9436	0.9281	0.9329	0.8025	0.8210	0.8357	0.3598	0.6219	0.8007	0.0000	0.4015	0.0400
6	0.9944	0.9934	0.9899	0.9951	0.9914	0.9933	0.9948	0.9828	**0.9963**	0.9905	0.9889	0.9814	0.9933	0.9860	0.9932	0.9812	0.9867	0.9897	0.1366
7	0.9604	0.9487	0.9246	0.9245	0.9609	0.9138	0.9367	0.8925	0.9069	**0.9647**	0.9254	0.9246	0.9156	0.9213	0.9188	0.8925	0.9195	0.9423	0.8261
8	0.8213	0.6292	0.7653	0.8322	0.7973	**0.8390**	0.7908	0.7684	0.6480	0.8387	0.7610	0.7311	0.7294	0.7330	0.7319	0.3752	0.7221	0.7104	0.4261
9	0.9780	**0.9791**	0.9685	0.9654	0.9524	0.9560	0.9652	0.9559	0.9200	0.9732	0.9189	0.8370	0.9410	0.9608	0.9546	0.8516	0.8915	0.8773	0.5833
10	0.8844	0.8981	0.7765	0.8150	0.8783	0.8699	0.9059	0.8967	0.7980	0.7595	0.8621	0.7606	0.8761	**0.9672**	**0.9672**	0.8967	0.9617	0.9643	0.4490
11	**0.9545**	0.9526	0.9049	0.9183	0.9408	0.9110	0.9129	0.9172	0.8859	0.8689	0.8528	0.8373	0.7967	0.8185	0.8271	0.8101	0.7990	0.5655	0.2747
12	0.9827	**0.9849**	0.9772	0.9804	0.9799	0.9810	0.9653	0.9705	0.9538	0.9629	0.8808	0.8907	0.8985	0.8760	0.6600	0.8691	0.0583	0.2391	0.1512
13	**0.9055**	0.8764	0.8953	0.8966	0.8364	0.8875	0.8639	0.8565	0.8408	0.8311	0.8548	0.7881	0.7995	0.7779	0.7180	0.7028	0.1915	0.0603	0.4830
14	0.9755	0.9751	0.9772	0.9742	0.9753	0.9740	0.9750	0.9690	0.9663	0.3688	0.9787	**0.9793**	0.9744	0.9029	0.9096	0.9690	0.9755	0.9670	0.9061
15	0.8625	0.8220	0.8105	0.8217	0.8224	0.8086	0.8517	0.8647	0.7757	0.0000	0.8049	0.7218	0.7984	0.8978	**0.9086**	0.7942	0.6953	0.6675	0.7872
16	**0.9938**	0.9936	0.9897	0.9917	0.9910	0.9916	0.9842	0.9899	0.9824	0.9815	0.9537	0.9384	0.9556	0.2476	0.7407	0.9355	0.0000	0.5249	0.0505
17	0.2792	0.1292	0.4337	0.4543	0.4076	0.5815	0.5474	0.5580	0.3752	**0.6619**	0.3603	0.3629	0.4095	0.5763	0.5549	0.1185	0.2447	0.0831	0.1241
18	0.5684	0.5383	0.5023	0.5665	**0.5851**	0.5502	0.5639	0.5443	0.5287	0.5347	0.4195	0.3732	0.4559	0.4716	0.4819	0.4015	0.4117	0.3969	0.0000
19	0.8347	0.8217	0.7012	**0.8356**	0.8345	0.8183	0.8033	0.8272	0.7652	0.8163	0.5010	0.5217	0.5073	0.4750	0.4939	0.7122	0.5122	0.4391	0.2182
AVG	**0.8602**	0.8359	0.8205	0.8437	0.8476	0.8514	0.8498	0.8464	0.7976	0.7558	0.7772	0.7444	0.7754	0.7417	0.7680	0.7321	0.5532	0.6124	0.3349

Table A3
The results of different algorithms on F-measure.

#dataset	DECOC	DOAO	HDD +ECOC	AdaBoost.M1	AdaC2.-M1	SAMME	imECOC+sparse	imECOC+dense	AdaBoost.NC	PIBoost	MC-HDDT	HDD +OVA	CART	Multi-IM +OVO	Multi-IM +A&O	imECOC+OVA	Multi-IM +OAHO	Multi-IM +OVA	IFROW-ANN +ECOC
1	**0.7513**	0.7467	0.7368	0.7481	0.7385	0.7403	0.7190	0.7214	0.7124	0.7261	0.6150	0.5893	0.6242	0.6353	0.6149	0.5740	0.5677	0.5607	0.2393
2	0.5309	0.5334	**0.5610**	0.5331	0.5442	0.5403	0.5088	0.5467	0.4364	0.4717	0.3999	0.4113	0.4426	0.4474	0.4307	0.3703	0.3518	0.3715	0.0638
3	0.5627	0.5473	0.5125	0.5270	0.5195	0.5398	0.5340	0.5367	0.4686	0.2799	0.4834	0.4585	0.4597	0.4162	0.3927	0.4421	**0.5766**	0.4159	0.1918
4	**0.9004**	**0.9004**	0.5436	0.6121	0.6133	0.6286	0.5699	0.5114	0.5717	0.6225	0.5452	0.5436	0.5199	0.5496	0.5257	0.5114	0.5745	0.5451	0.4791
5	0.8016	**0.8028**	0.7793	0.7975	0.7850	0.7928	0.7118	0.7710	0.7223	0.7374	0.4527	0.4846	0.5075	0.3001	0.2791	0.4480	0.1287	0.1743	0.1248
6	0.9866	0.9829	0.9784	0.9879	0.9871	0.9853	0.9811	0.9566	**0.9898**	0.9744	0.9682	0.9638	0.9817	0.9588	0.9693	0.9569	0.9715	0.9748	0.1168
7	0.9418	0.9242	0.8984	0.8854	0.9407	0.8696	0.9083	0.8451	0.8617	**0.9422**	0.8934	0.8984	0.8781	0.8943	0.8931	0.8451	0.8772	0.9238	0.7562
8	0.6513	0.5988	0.5832	0.6669	0.6326	0.6674	0.6086	0.5838	0.6159	**0.6702**	0.5550	0.5364	0.5514	0.5553	0.5355	0.4599	0.5240	0.4971	0.3279
9	0.9317	**0.9406**	0.9223	0.9083	0.8605	0.8724	0.9028	0.8740	0.8193	0.9178	0.8466	0.7067	0.8458	0.8640	0.8563	0.7107	0.7756	0.7593	0.5582
10	0.8456	0.8498	0.7816	0.7510	0.8039	0.8223	**0.8621**	0.8070	0.7694	0.7899	0.7777	0.7565	0.8294	0.7361	0.7361	0.8070	0.7822	0.8183	0.3600
11	**0.8655**	0.8592	0.7772	0.8243	0.8311	0.7977	0.7836	0.7927	0.7404	0.7470	0.6650	0.6266	0.6067	0.5553	0.5711	0.6001	0.6110	0.2677	0.1382
12	0.9226	**0.9295**	0.8967	0.9103	0.9077	0.9124	0.8500	0.8694	0.8087	0.8403	0.6008	0.6250	0.6416	0.6446	0.3509	0.5744	0.1269	0.1502	0.1699
13	**0.8731**	0.7854	0.8410	0.8338	0.7444	0.8150	0.7481	0.7628	0.7338	0.7084	0.7219	0.6501	0.6269	0.5814	0.5067	0.5077	0.4007	0.1887	0.3328
14	0.9572	0.9540	0.9511	0.9551	**0.9593**	**0.9593**	0.9362	0.9210	0.9421	0.4064	0.9498	0.9540	0.9469	0.6381	0.6617	0.9210	0.9222	0.9218	0.6741
15	**0.6377**	0.6091	0.5793	0.5914	0.5589	0.5767	0.5841	0.5879	0.5368	0.0140	0.5526	0.5276	0.5534	0.4227	0.4284	0.4927	0.5174	0.4521	0.2657
16	**0.9698**	0.9689	0.9502	0.9599	0.9565	0.9591	0.9261	0.9514	0.9183	0.9139	0.8057	0.7554	0.8118	0.3624	0.4088	0.7458	0.1562	0.2415	0.1746
17	0.4549	0.4458	0.4545	**0.4734**	0.4349	0.4528	0.4067	0.4224	0.3531	0.3970	0.3584	0.3750	0.4007	0.3658	0.3306	0.3391	0.3537	0.1922	0.2534
18	**0.3936**	0.3813	0.3437	0.3862	0.3774	0.3725	0.3680	0.3390	0.3610	0.3504	0.2522	0.2488	0.2735	0.2858	0.2785	0.2386	0.2406	0.2395	0.2204
19	**0.7466**	0.7321	0.5804	0.7436	0.6736	0.7362	0.6160	0.6460	0.6501	0.7254	0.3789	0.4230	0.3831	0.1079	0.1401	0.4450	0.3026	0.2905	0.2062
AVG	**0.7750**	0.7628	0.7195	0.7419	0.7300	0.7390	0.7118	0.7077	0.6848	0.6439	0.6222	0.6071	0.6255	0.5432	0.5216	0.5784	0.5137	0.4729	0.2975

Table A4
The results of different algorithms on AUC.

#dataset	DECOC	DOAO	HDD + -ECOC	AdaBoos-t.M1	AdaC2.-M1	SAMME	imECOC- + sparse	imECOC- + dense	AdaBoos-t.NC	PIBoost	MC-HDDT	HDD + -OVA	CART	Multi-IM + OVO	Multi-IM + A&O	imECOC- + OVA	Multi-IM + OAHO	Multi-IM + OVA	IFROW-ANN + -ECOC
1	**0.8933**	0.8886	0.8858	0.8894	0.8911	0.8876	0.8822	0.8844	0.8737	0.8797	0.8328	0.8184	0.8388	0.8465	0.8466	0.8268	0.8276	0.8232	0.7301
2	0.8542	0.8619	0.8610	0.8519	**0.8846**	0.8517	0.8529	0.8809	0.8020	0.8416	0.7854	0.7735	0.8271	0.8421	0.8528	0.8075	0.8019	0.8097	0.7373
3	0.7777	0.7590	0.7543	0.7656	0.7728	0.7551	0.8083	0.8065	0.7086	0.8218	0.7654	0.7037	0.7496	**0.8503**	0.8492	0.7945	0.7721	0.7731	0.7794
4	**0.9471**	**0.9471**	0.7029	0.7586	0.7445	0.7687	0.7222	0.6852	0.7273	0.7520	0.7084	0.7032	0.6974	0.7238	0.7004	0.6852	0.7400	0.6882	0.5000
5	**0.9713**	0.9680	0.9644	0.9674	0.9659	0.9672	0.9504	0.9612	0.9532	0.9544	0.8648	0.8572	0.8879	0.8617	0.8083	0.8723	0.7275	0.7948	0.6875
6	0.9933	0.9933	0.9869	0.9944	0.9893	0.9919	0.9948	0.9817	**0.9962**	0.9886	0.9885	0.9775	0.9928	0.9846	0.9942	0.9786	0.9851	0.9880	0.7221
7	0.9477	0.9415	0.9398	0.9088	0.9575	0.9040	0.9281	0.8806	0.8929	**0.9642**	0.9288	0.9398	0.9092	0.9033	0.9067	0.8806	0.9212	0.9334	0.8800
8	0.8791	0.8607	0.8594	0.8951	0.8801	**0.9015**	0.8688	0.8177	0.8695	0.8922	0.8710	0.8671	0.7987	0.8336	0.7970	0.7817	0.8526	0.8457	0.8003
9	0.9872	**0.9882**	0.9766	0.9747	0.9684	0.9760	0.9692	0.9732	0.9388	0.9817	0.9608	0.8870	0.9467	0.9733	0.9767	0.9231	0.9608	0.9448	0.9133
10	0.9041	0.8918	0.8161	0.8235	0.8841	0.8663	0.9337	0.9145	0.8002	0.9275	0.8976	0.8105	0.9017	0.9591	0.9591	0.9145	0.9631	**0.9644**	0.6972
11	**0.9712**	0.9696	0.9330	0.9335	0.9623	0.9330	0.9278	0.9311	0.9249	0.9205	0.9143	0.8836	0.8628	0.8999	0.9005	0.8563	0.8863	0.8291	0.7719
12	0.9866	**0.9896**	0.9834	0.9863	0.9852	0.9864	0.9743	0.9767	0.9674	0.9711	0.9161	0.9123	0.9266	0.9333	0.8589	0.9110	0.8149	0.8139	0.7703
13	**0.9610**	0.9470	0.9388	0.9427	0.9278	0.9434	0.9389	0.9330	0.9267	0.9201	0.9394	0.8340	0.8949	0.8680	0.8597	0.8056	0.8429	0.8242	0.8440
14	0.9881	0.9879	0.9870	0.9876	0.9880	0.9873	0.9855	0.9788	0.9837	0.6736	**0.9896**	0.9881	0.9876	0.9539	0.9566	0.9788	0.9873	0.9812	0.9543
15	0.8530	0.8179	0.8112	0.8289	0.8271	0.8097	0.8490	0.8622	0.7866	0.5607	0.8157	0.7738	0.8204	0.8952	**0.9040**	0.8318	0.8692	0.8575	0.8404
16	**0.9967**	0.9957	0.9938	0.9960	0.9946	0.9949	0.9908	0.9947	0.9892	0.9899	0.9694	0.9480	0.9705	0.9064	0.8859	0.9534	0.7633	0.8572	0.7767
17	0.8852	0.8648	0.8892	0.8796	0.8767	**0.9001**	0.8800	0.8890	0.8478	0.8613	0.8620	0.7877	0.8477	0.8754	0.8504	0.8036	0.8363	0.8377	0.8094
18	**0.7294**	0.7161	0.6908	0.7235	0.7286	0.7151	0.7195	0.6995	0.7087	0.7103	0.6370	0.6212	0.6571	0.6594	0.6592	0.6246	0.6214	0.6383	0.5029
19	0.9106	0.9015	0.8219	**0.9250**	0.9015	0.9026	0.8973	0.9222	0.8792	0.9064	0.7220	0.7008	0.7370	0.6764	0.7137	0.8296	0.7635	0.7520	0.6626
AVG	**0.9177**	0.9100	0.884	0.8964	0.9016	0.897	0.8986	0.8933	0.8725	0.8694	0.8615	0.8309	0.8555	0.8656	0.8568	0.8452	0.8388	0.8398	0.7568

Table A5
The running time of different algorithms (in seconds).

#dataset	DOAO	HDD + -ECOC	AdaBoo-st.M1	AdaC2.M1	SAMME	imECOC- + sparse	imECOC- + dense	AdaBoost.NC	PIBoost	MC-HDDT	HDD + OVA	CART	Multi-IM + OVO	Multi-IM + A&O	imECOC- + OVA	Multi-IM + OAHO	Multi-IM + OVA	IFROW-ANN + ECOC
1	700.40	104.97	14.18	21,820.00	14.84	14.91	14.92	14.80	1018.30	20.18	17.24	0.53	3.53	12.03	2.13	2.62	8.49	13,492.98
2	391.40	29.97	3.97	6086.60	4.37	9.79	7.55	4.46	2461.80	5.39	3.82	0.14	3.38	14.49	0.69	0.91	11.10	1381.18
3	184.50	13.31	3.31	1530.60	3.48	8.55	5.20	3.66	61.14	2.41	2.78	0.10	41.43	94.26	0.55	1.50	52.79	9178.18
4	53.45	1.78	2.84	1386.50	2.90	0.73	0.31	2.86	251.64	1.34	1.72	0.10	0.33	0.91	0.26	0.28	0.58	31.78
5	1082.00	129.07	15.35	9752.10	16.87	20.15	18.42	15.84	1264.00	38.30	22.87	0.50	5.75	33.91	3.06	9.66	28.16	46,944.73
6	288.90	8.83	3.39	242.95	3.71	6.31	1.13	3.40	378.25	5.18	4.41	0.11	1.65	5.28	0.50	0.54	3.63	10,875.09
7	11.89	0.10	0.17	18.44	0.18	0.33	0.07	0.17	22.58	0.04	0.04	0.01	0.04	0.10	0.02	0.03	0.07	2.64
8	38.20	4.03	0.46	87.79	0.46	5.54	3.59	0.49	490.94	0.14	0.15	0.02	0.18	0.46	0.06	0.08	0.28	22.87
9	89.77	6.61	0.64	208.91	0.66	5.26	4.84	0.66	852.24	0.43	0.32	0.02	0.28	0.88	0.09	0.13	0.61	70.31
10	25.89	0.19	0.47	29.63	0.42	0.39	0.10	0.44	172.75	0.15	0.15	0.01	1.24	2.79	0.05	0.27	1.55	56.82
11	116.60	12.07	2.28	950.39	2.31	6.96	4.43	2.40	96.53	1.96	2.40	0.09	1.68	4.99	0.31	0.35	3.30	1298.04
12	2545.00	181.52	15.74	10,068.00	17.25	18.78	19.36	15.99	538.59	53.32	29.34	0.55	4.92	21.86	2.63	6.81	16.93	16,321.08
13	404.20	17.06	2.87	5863.70	3.10	6.14	5.87	2.78	219.26	2.23	1.92	0.09	0.53	1.62	0.28	0.31	1.09	138.32
14	111.40	2.27	2.18	249.05	2.31	0.65	0.25	2.30	40.33	2.07	2.22	0.07	0.86	2.21	0.20	0.32	1.35	510.65
15	301.20	12.25	3.14	1139.36	3.24	8.49	5.09	3.25	175.13	2.30	2.49	0.11	40.48	88.57	0.54	1.53	48.07	14,498.74
16	1253.00	103.60	9.36	6789.05	10.03	17.85	15.04	9.23	572.39	28.50	16.89	0.30	4.65	30.35	2.43	8.61	25.70	50,499.86
17	46.57	4.90	0.36	1161.52	0.38	5.09	4.63	0.38	101.69	0.09	0.10	0.01	0.34	0.88	0.06	0.07	0.54	29.55
18	923.10	173.30	39.23	19,470.00	40.85	32.76	20.53	40.68	315.03	32.52	48.88	1.47	8.51	24.59	4.99	5.61	16.07	43,006.99
19	77.69	5.74	1.56	1000.42	1.56	7.04	4.26	1.50	58.23	0.49	0.69	0.04	12.26	27.91	0.30	0.99	15.65	4852.96
AVG	455.00	42.72	6.39	4623.95	6.79	9.25	7.14	6.59	478.46	10.37	8.34	0.22	6.95	19.37	1.01	2.14	12.42	11,221.73

References

[1] H. He, E.A. Garcia, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (9) (2009) 1263–1284 Sep..
[2] N. Chawla, N. Japkowicz, A. Kolcz, Editorial: special issue on learning from imbalanced data sets, ACM SIGKDD Explor. Newsl. 6 (1) (2004) 1–6.
[3] G. Haixiang, et al., , Learning from class-imbalanced data: review of methods and applications, Expert Syst. Appl. 73 (2017) 220–239.
[4] N.V. Chawla, L.O. Hall, K.W. Bowyer, W.P. Kegelmeyer, SMOTE: synthetic minority oversampling technique, J. Artif. Intell. Res. (JAIR) (2002) 321–357.
[5] H.B. He, Y. Bai, E.A. Garcia, S.T. Li, ADASYN adaptive synthetic sampling approach for imbalanced learning, Proceedings of International Joint Conference on Neural Networks (IJCNN), 2008, pp. 1322–1328.
[6] X.Y. Liu, J.X. Wu, Z.H. Zhou, Exploratory under-sampling for class-imbalance learning, Proceedings of International Conference on Data Mining (ICDM), 2006.
[7] S. Wang, X. Yao, Multi-class imbalance problems: analysis and potential solutions, IEEE Trans. Syst. Man Cybernetics Part B 42 (4) (2012) 1119–1130.
[8] T.R. Hoens, Q. Qian, N.V. Chawla, et al., Building Decision Trees for the Multi-Class Imbalance Problem, Advances in Knowledge Discovery and Data Mining, Springer, Berlin Heidelberg, 2012, pp. 122–134.
[9] A.S. Ghanem, S. Venkatesh, G. West, Multi-class pattern classification in imbalanced data, International Conference on Pattern Recognition, IEEE Computer Society, 2010, pp. 2881–2884.
[10] X.Y. Liu, Q.Q. Li, ZH. Zhou, Learning imbalanced multi-class data with optimal dichotomy weights, IEEE 13th International Conference on Data Mining (IEEE ICDM), 2013, pp. 478–487.
[11] S. Kang, S. Cho, P. Kang, Constructing a multi-class classifier using one-against-one approach with different binary classifiers, Neurocomputing 149 (2015) 677–682.
[12] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1) (1997) 119–139.
[13] Y. Sun, M.S. Kamel, Y. Wang, Boosting for learning multiple classes with imbalanced class distribution, Proceedings of the 6th International Conference on Data Mining, 2006, pp. 592–602.
[14] J. Zhu, H. Zou, S. Rosset, et al., Multi-class AdaBoost, Stat. Interface 2 (3) (2006) 349–360.
[15] S. Wang, H. Chen, X. Yao, Negative correlation learning for classification ensembles, Proceedings of International Joint Conference on Neural Networks, WCCI, 2010, pp. 2893–2900.
[16] B.A. Fernández, L. Baumela., Multi-class boosting with asymmetric binary weak-learners, Pattern Recognit. 47 (5) (2014) 2080–2090.
[17] M. Kubat, R. Holte, S. Matwin, Learning When Negative Examples Abound, Machine Learning: ECML-97, Springer, Berlin Heidelberg, 1997, pp. 146–153.
[18] V. Rijsbergen, Information Retrieval, Butterworths, London, U.K., 1979.
[19] A.P. Bradley, The use of the area under the ROC curve in the evaluation of machine learning algorithms, Pattern Recognit. 30 (7) (1997) 1145–1159.
[20] T. Dietterich, G. Bakiri, Error-correcting output codes: A general method for improving multiclass inductive learning programs, AAAI (1994) 395 395.
[21] P.N. Garcia, B.D. Ortiz, Improving multiclass pattern recognition by the combination of two strategies, IEEE Trans. Pattern Anal Mach. Intell. 28 (6) (2006) 1001–1006.
[22] Y.L. Murphey, H. Wang, G. Ou, et al., OAHO: an effective algorithm for multi-class learning from imbalanced data, International Joint Conference on Neural Networks, IEEE, 2007, pp. 406–411.
[23] S. Ghanem, S. Venkatesh, G. West, Learning in imbalanced relational data, International Conference on Pattern Recognition, 2008, pp. 1–4.
[24] S.Y. Sohn, Meta Analysis of classification algorithms for pattern recognition, IEEE Trans. Pattern Anal. Mach. Intell. 21 (11) (1999) 1137–1144.
[25] T.S. Lim, W.Y. Loh, Y.S. Shih, A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms, Mach. Learn. 40 (3) (2000) 203–228.
[26] M.Y. Kiang, A comparative assessment of classification methods, Decis. Support Syst. 35 (4) (2003) 441–454.
[27] K. Woods, W.P. Kegelmeyer, K. Bowyer, Combination of multiple classifiers using local accuracy estimates, IEEE Trans. Pattern Anal. Mach. Intell. 19 (4) (1997) 405–410.
[28] P.R. Cavalin, R. Sabourin, C.Y. Suen, Dynamic selection approaches for multiple classifier systems, Neural Comput. Appl. 22 (3) (2013) 673–688.
[29] L. Xu, A. Krzyzak, C.Y. Suen, Methods of combining multiple classifiers and their applications to handwriting recognition, IEEE Trans. Cybernetics 22 (3) (1992) 418–435.
[30] T.K. Ho, J.J. Hull, S.N. Srihari, Decision combination in multiple classifier systems, IEEE Trans. Pattern Anal. Mach. Intell. 16 (1) (1994) 66–75.
[31] J. Kittler, M. Hatef, R.P.W. Duin, et al., On combining classifiers, IEEE Trans. Pattern Anal. Mach. Intell. 20 (3) (1998) 226–239.
[32] R. Lysiak, M. Kurzynski, T. Woloszynski, Optimal selection of ensemble classifiers using measures of competence and diversity of base classifiers, Neurocomputing 126 (126) (2014) 29–35.
[33] D.A. Cieslak, N.V. Chawla, Learning Decision Trees for Unbalanced Data, European Conference on Machine Learning and Knowledge Discovery in Databases, Springer-Verlag, 2008, pp. 241–256.
[34] R.E. Schapire, Y. Singer, Improved boosting algorithms using confidence-rated predictions, Mach. Learn. 37 (3) (1999) 297–336.
[35] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.
[36] S. Wang, X. Yao, Negative correlation learning for class imbalance problems, IEEE Trans. Neural Netw. (2011).
[37] B. Zhu, B. Baesens, A. Backiel, et al., , Benchmarking sampling techniques for imbalance learning in churn prediction Customer churn prediction, J. Oper. Res. Soc. (2017) 1–17.
[38] G.E. Batista, R.C. Prati, M.C. Monard, A study of the behavior of several methods for balancing machine learning training data, Proceedings of ACM SIGKDD Explorations Newsletter, 6 2004, pp. 20–29.
[39] Z. Sun, Q. Song, X. Zhu, et al., A novel ensemble method for classifying imbalanced data, Pattern Recognit. 48 (5) (2015) 1623–1637.
[40] L. Nanni, C. Fantozzi, N. Lazzarini, Coupling different methods for overcoming the class imbalance problem, Neurocomputing 158 (2015) 48–61.
[41] Y. Li, H. Guo, X. Liu, et al., Adapted ensemble classification algorithm based on multiple classifier system and feature selection for classifying multi-class imbalanced data, Knowl-Based Syst. 94 (2016) 88–104.
[42] Z. Zhang, B. Krawczyk, S. Garcìa, et al., Empowering one-vs-one decomposition with ensemble learning for multi-class imbalanced data, Knowl-Based Syst. 106 (2016) 251–263.
[43] J.A. Sáez, B. Krawczyk, M. Woźniak, Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets, Pattern Recognit. 57 (2016) 164–178.
[44] N.F. Fernández, M.C. Hervás, G.P. Antonio, A dynamic over-sampling procedure based on sensitivity for multi-class problems, Pattern Recognit. 44 (8) (2011) 1821–1833.
[45] J. Sun, H. Fujita, P. Chen, et al., Dynamic financial distress prediction with concept drift based on time weighting combined with Adaboost support vector machine ensemble, Knowl-Based Syst. 120 (2016) 4–14.
[46] L. Zhou, H. Fujita, Posterior probability based ensemble strategy using optimizing decision directed acyclic graph for multi-class classification, Inf. Sci. 400 (401) (2017) 142–156.
[47] J. Sun, J. Lang, H. Fujita, et al., , Imbalanced Enterprise Credit Evaluation with DTE-SBD: Decision Tree Ensemble Based on SMOTE and Bagging with Differentiated Sampling Rates, Inf. Sci. 425 (2018) 76–91.
[48] E. Ramentol, Y. Caballero, R. Bello, et al., SMOTE-RSB, * : a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using SMOTE and rough sets theory, Knowl. Inf. Syst. 33 (2) (2012) 245–265.
[49] E. Ramentol, S. Vluymans, N. Verbiest, et al., IFROWANN: imbalanced fuzzy-rough ordered weighted average nearest neighbor classification, IEEE Trans. Fuzzy Syst. 23 (5) (2015) 1622–1637.
[50] Jie Sun, Jie Lang, Hamido Fujita, Hui Li, Imbalanced enterprise credit evaluation with DTE-SBD: Decision tree ensemble based on SMOTE and bagging with differentiated sampling rates, Inf. Sci. 425 (2018) 76–91.
[51] Jie Sun, Hamido Fujita, Peng Chen, Hui Li, Dynamic financial distress prediction with concept drift based on time weighting combined with Adaboost support vector machine ensemble, Knowl-Based Syst. 120 (2017) 4–14.
[52] Ligang Zhou, KwoPing Tam, Hamido Fujita, Predicting the listing status of Chinese listed companies with multi-class classification models, Inf. Sci. 328 (2016) 222–236.
[53] Sarah Vluymans, Alberto Fernández, YvanSaeysChris Cornelis, Francisco Herrera, Dynamic affinity-based classification of multi-class imbalanced data with one-versus-one decomposition: a fuzzy rough set approach, Knowl. Inf. Syst. 1 (2018) 1–30.
[54] C. Zhang, C. Liu, X. Zhang, et al., An up-to-date comparison of state-of-the-art classification algorithms, Expert Syst. Appl. 82 (2017) 128–150.
[55] O.J. Dunn, Multiple comparisons using rank sums, Technometrics 6 (3) (1964) 241–252.
[56] W.W. Daniel, Applied Nonparametric Statistics, (2nd ed.), Cengage Learning, 2000.