Knowledge-Based Systems 203 (2020) 106087

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

A weighted hybrid ensemble method for classifying imbalanced data

Jiakun Zhao [a], Ju Jin [a,*], Si Chen [a], Ruifeng Zhang [a], Bilin Yu [b], Qingfang Liu [c]

[a] School of Software Engineering, Xi'an Jiaotong University, 710049, China
[b] School of Management, University of Science and Technology of China, 230026, China
[c] School of Mathematics and Statistics, Xi'an Jiaotong University, 710049, China

ARTICLE INFO

Article history:
Received 26 February 2020
Received in revised form 6 May 2020
Accepted 28 May 2020
Available online 8 June 2020

Keywords:
Data imbalance
Binary classification
Boosting algorithm
Data sampling methods
Base classifiers

ABSTRACT

In real datasets, most are unbalanced. Data imbalance can be defined as the number of instances in some classes greatly exceeds the number of instances in other classes. Whether in the field of data mining or machine learning, data imbalance can have adverse effects. At present, the methods to solve the problem of data imbalance can be divided into data-level methods, algorithm-level methods and hybrid methods. In this paper, we propose a weighted hybrid ensemble method for classifying imbalanced data in binary classification tasks, called WHMBoost. In the framework of the boosting algorithm, the presented method combines two data sampling methods and two base classifiers, and each sampling method and each base classifier is assigned corresponding weights, which makes them have better complementary advantages. The performance of WHMBoost has been evaluated on 40 benchmark imbalanced datasets with state of the art ensemble methods like AdaBoost, RUSBoost, SMOTEBoost using AUC, F-Measure and Geometric Mean as the performance evaluation criteria. Experimental results show significant improvement over the other methods and it can be concluded that WHMBoost is a promising and effective algorithm to deal with imbalance datasets.

1. Introduction

If the number of instances is inconsistent among the various classes of the dataset, the dataset is considered imbalanced. In actual data mining and machine learning, basically all data sets are skewed. For example, there is an imbalance between classes in the credit card fraud detection problem [1]. In all transactions, the normal transaction (majority class) occupies the majority of the dataset, and the fraud transaction (minority class) occupies only the minority of the dataset. When training a model on such a dataset, it is difficult for the classifier or discriminator to identify the minority class. And the classifier will bias the majority class that occupies the majority in the dataset and ignore the minority class that is important to us. In order to address the problem of data imbalance in the dataset, researchers have proposed many methods. Johnson et al. [2] divided these approaches into three categories: data-level methods, algorithm-level methods and hybrid methods.

The data-level methods are mainly to reduce the data imbalance by changing the data distribution so that the classifier will not be too biased towards a certain class. In the data-level methods, the method of solving the data imbalance is mainly the data sampling method which can be divided into undersampling and oversampling. They all reduce the data imbalance by changing the data distribution. In the undersampling methods, the simplest random undersampling achieves data balance by randomly discarding instances of the majority class. It can be further divided into random undersampling without replacement and random undersampling with replacement. The former only samples instances of the majority class once, while the latter may have duplicate instances of the majority class. The random undersampling is simple, but it may throw away more important data. Later researchers have proposed many other undersampling methods to effectively avoid the disadvantages of the random undersampling, such as NearMiss family [3], Condensed Nearest Neighbor (CNN) [4], Edited Nearest Neighbor (ENN) [5], Tomek Link Removal [6] and so on. Yen et al. [7] also presented a cluster-based undersampling method for selecting representative data. It can solve the imbalance between classes as well as the imbalance within classes, which can effectively improve the classification performance of the minority class. In the oversampling approaches, the simplest random oversampling is to achieve a balance of data distribution by randomly repeating instances of the minority class. It is likely to result in overfitting of models in traditional machine learning. In order to avoid over fitting problems in oversampling, some intelligent oversampling techniques have been proposed, such as SMOTE [8], Borderline-SMOTE [9], AdaSyn [10], Safe-Level-SMOTE [11] and so on. In order to combine the advantages of undersampling and oversampling, some methods of combining undersampling and oversampling have also been investigated, such as the combination of Tomek Link

* Corresponding author.
E-mail address: xjtujinju@stu.xjtu.edu.cn (J. Jin).

https://doi.org/10.1016/j.knosys.2020.106087
0950-7051/

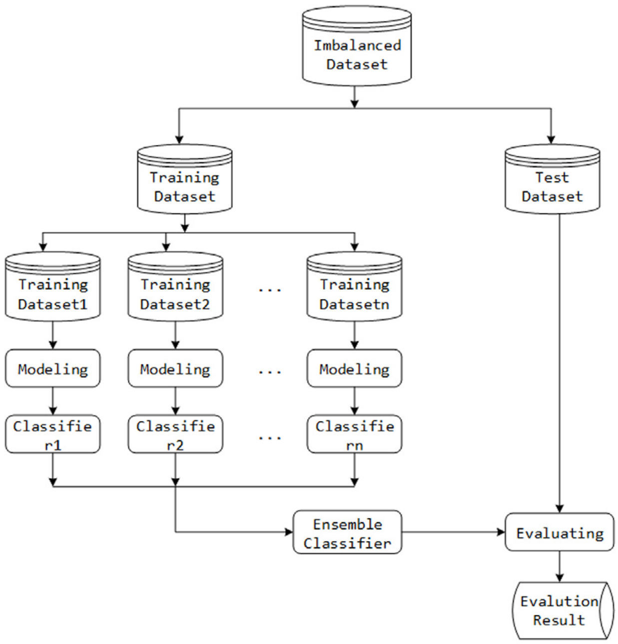

Fig. 1. The process of ensemble methods for classifying unbalanced data.

Removal [6] and SMOTE [8], the combination of ENN [5] and SMOTE [8]. Leery et al. [12] also classified feature selection as a data-level method to solve the problem of data imbalance. It can be utilized to select the most influential features that are beneficial to distinguish between the classes, which will reduce the negative impact of data imbalance on classification performance. Zhou et al. [13] proposed an online feature selection algorithm based on K nearest neighbor dependencies for high-dimensional unbalanced datasets, which uses nearest neighbor information to select relevant features.

The algorithm-level methods do not need to modify the distribution of the dataset. It reduces the sensitivity of the algorithm to the data distribution by changing the learning process or discrimination method, so that the classifier pays more attention to the minority class. Such methods mainly include new loss function, cost-sensitive learning, threshold moving, one-class classification and ensemble methods. The new loss function allows instances of the minority class to contribute more to the final loss. For example, Lin et al. [14] solved the extreme imbalance problem of foreground-background class encountered in the target detection problem by reforming the standard cross-entropy loss so as to reduce the loss allocated to good classification examples. Cost-sensitive learning assigns different costs to the misclassification of instances of different classes. Generally, the misclassification cost of the majority class is small and the misclassification cost of the minority class is large, so that the model pays more attention to the minority class. However, this cost allocation is difficult and may require knowledge in the domain. If the allocation is not good, the model may be unstable. For example, Huang et al. [15] proposed a dynamic cost-sensitive ensemble classification method in order to obtain lower classification costs and improve classification performance. Threshold moving changes the class probability of model output by adjusting the decision threshold of the model. Threshold moving does not affect model training and is generally only used in the test phase. For example, Buda et al. [16] used three datasets to study the impact of data imbalance on classification problems and to perform a comparison of threshold moving and some other methods in order to verify the effectiveness of these methods in solving data imbalance problems. One-class classification only identifies positive examples in the dataset and is generally used in cases of extreme data imbalance. For example, Zhu et al. [17] presented a new one-class support vector machine based on hidden information. Experimental results show that this method has very good effectiveness and advantages. The ensemble methods combine multiple base classifiers together to determine the output of the ensemble model in order to improve the performance of classification, such as bagging and boosting. In [18], the author applied bagging and boosting to a learning decision tree system and tested it on some representative datasets. It was concluded that both methods can improve prediction accuracy, but boosting is better. In [19], the authors propose online versions of bagging and boosting in order to overcome the situation that can only be used in batch mode.

The hybrid approaches are the approaches that combine data-level methods and algorithm-level methods. By modifying the data distribution to reduce the data imbalance and changing the learning process to reduce the sensitivity of the algorithm to the data distribution, the hybrid methods pay more attention to the minority class, so that the classifier can distinguish the majority class and the minority class well. For example, various sampling techniques can be coupled with cost-sensitive learning or threshold moving to jointly improve classification performance. In [20], researchers combined ensemble methods with sampling techniques and cost-sensitive learning to achieve superior performance in classification tasks. The methods of combining sampling technology and ensemble methods include CUSBoost [21], SMOTEBoost [22], RUSBoost [23] and so on.

In this paper, a weighted hybrid ensemble method is presented to solve the problem of data imbalance in binary classification tasks. In the framework of the boosting algorithm, our method combines two data sampling techniques and two base classifiers and assigns weights to each sampling method and each base classifier. Our intuition is that different sampling techniques and base classifiers have different advantages, and there may be certain limitations. Through this hybrid approach, these sampling techniques and base classifiers can complement each other. The proposed method produces a better classification performance than the method using a sampling method and a base classifier. We choose random undersampling and adjustable random balance as our collection of sampling methods. In terms of base classifiers, we choose decision tree classifiers and support vector machines as the base classifier set. We used AUC, F-Measure and Geometric Mean as the performance evaluation method and compared the presented method with the previously proposed ensemble methods (AdaBoost, RUSBoost [23], RBBoost [24], RHS-Boost [25], SMOTEBoost [22], CUSBoost [21], MEBoost [26]) on 40 datasets with different imbalance rates. Experimental results show that when using AUC, F-Measure and Geometric Mean as the performance evaluation method, the method in this paper is superior to other methods.

The main contributions of this paper are summarized as: (1) The random balance has been improved, an adjustable random balance has been proposed and its feasibility has been verified through experiments. (2) A weighted hybrid ensemble method is put forward, which uses two sampling methods and two base classifiers. (3) Through experiments, it is verified that the performance of the proposed ensemble method is better than that of other ensemble methods. (4) Through experiments, it is proved that the performance of the presented ensemble model is better than that of some single models.

The remainder of the paper is organized as follows: Section 2 presents related work. Section 3 shows the details of our proposed method. Section 4 presents the results and analysis of the experiments. Finally, Section 5 concludes this paper.

Algorithm 1 Random Balance

Require: Set S of example (x_1,y_1), ..., (x_m,y_m) where $x_i \in X \subseteq R^n$ and $y_i \in Y = \{-1, +1\}$(+1: positive or minority class, -1: negative or majority class)
  neighbours used in SMOTE, k
Ensure: New set S' of examples with Random Balance

totalSize $\leftarrow |S|$
$S_N \leftarrow \{(x_i, y_i) \in S | y_i = -1\}$
$S_P \leftarrow \{(x_i, y_i) \in S | y_i = +1\}$
majoritySize $\leftarrow |S_N|$
minoritySize $\leftarrow |S_P|$
newMajoritySize $\leftarrow$ Random integer between 2 and totalSize - 2
newMinoritySize $\leftarrow$ totalSize - newMajoritySize
if newMajoritySize < majoritySize **then**
  $S' \leftarrow S_P$
  Take a random sample of size newMajoritySize from S_N, add the sample to S'.
  Create newMinoritySize - minoritySize artificial examples from S_P using SMOTE, add these examples to S'.
else
  $S' \leftarrow S_N$
  Take a random sample of size newMinoritySize from S_P, add the sample to S'.
  Create newMajoritySize - majoritySize artificial examples from S_N using SMOTE, add these examples to S'.
end if
return S'

Algorithm 2 Adjustable Random Balance

Require: Set S of example (x_1,y_1), ..., (x_m,y_m) where $x_i \in X \subseteq R^n$ and $y_i \in Y = \{-1, +1\}$(+1: positive or minority class, -1: negative or majority class)
  neighbours used in SMOTE, k
  scale factor, ratio
Ensure: New set S' of examples with Adjustable Random Balance

totalSize $\leftarrow |S|$
$S_N \leftarrow \{(x_i, y_i) \in S | y_i = -1\}$
$S_P \leftarrow \{(x_i, y_i) \in S | y_i = +1\}$
majoritySize $\leftarrow |S_N|$
minoritySize $\leftarrow |S_P|$
rangeMinimum $\leftarrow$ (1.0-ratio)*majoritySize
if rangeMinimum < 2 **then**
  rangeMinimum $\leftarrow$ 2
end if
rangeMaximum $\leftarrow$ majoritySize+ratio * minoritySize
if rangeMaximum > (totalSize - 2) **then**
  rangeMaximum $\leftarrow$ totalSize-2
end if
newMinoritySize $\leftarrow$ totalSize-newMajoritySize
if newMajoritySize < majoritySize **then**
  $S' \leftarrow S_P$
  Take a random sample of size newMajoritySize from S_N, add the sample to S'.
  Create newMinoritySize - minoritySize artificial examples from S_P using SMOTE, add these examples to S'.
else
  $S' \leftarrow S_N$
  Take a random sample of size newMinoritySize from S_P, add the sample to S'.
  Create newMajoritySize - majoritySize artificial examples from S_N using SMOTE, add these examples to S'.
end if
return S'

2. Related work

2.1. Other ensemble methods

Improving the performance of classifiers trained with imbalanced datasets has become a huge challenge in the field of machine learning. At present, a growing number of researchers pay attention to the problem of data imbalance. Many ensemble methods based on boosting have been proposed to solve various data imbalance problems encountered during data classification. Fig. 1 illustrates the process of an ensemble method for classifying unbalanced data. In the picture, *Training Dataset 1*, *Training Dataset 2*, …, *Training Dataset n* and *Training Dataset* may be the same or different. The classification algorithms used by *Classifier 1*, *Classifier 2*, …, *Classifier n* may be the same or different.

Freund et al. [27,28] proposed the Adaboost algorithm which is an improvement on the boosting algorithm. The principle of this algorithm is to adjust the weights of samples and weak classifiers and combine the trained weak classifiers to form a final strong classifier in order to jointly determine the output of the ensemble model. The base classifier is trained based on the training set and each time the next base classifier is obtained by training on different weight sets of samples. The weight of each sample is determined by the classification difficulty which is estimated by the output of the classifier in the previous step.

Chawla et al. [22] presented the SMOTEBoost algorithm which combines the synthetic minority sampling technique (SMOTE) and the boosting algorithm to improve the prediction accuracy of models learned from imbalanced datasets. According to the new dataset generated by SMOTE, the boosting algorithm learns a strong classifier composed of multiple weak classifiers, so as to improve the accuracy of the whole dataset. Because SMOTE is used, SMOTEBoost can effectively avoid overfitting of the classifier, but this makes it more complicated and the model training time will increase due to more samples.

Seiffert et al. [23] presented a hybrid sampling/boosting method called RUSBoost based on SMOTEBoost, which combines random undersampling and adaboost algorithm. The undersampling method randomly samples instances from the majority class, thus making the data distribution more balanced. RUSBoost is simpler than SMOTEBoost and because the data used to train the model is reduced, model training is faster. However, random undersampling may discard those important data, resulting in poor model performance.

Díez-Pastor et al. [24] proposed an ensemble method called RBBoost which combines Adaboost and the random balance put forward by them. In random balance, the dataset used to train each weak classifier in the ensemble model is generated by randomly undersampling one class and oversampling another class using SMOTE. The class being undersampled and the proportion of undersampling are randomly determined, as is the oversampling. The specific process of the random balance is shown in Algorithm 1. Although the random balance changes the data distribution, the size of the dataset does not change so that the model training time does not increase. At the same time, it increases the diversity of the dataset, so the classification effect of the ensemble model may become better. However, the random balance is unstable and the imbalance rate of the dataset generated by it may be higher than that of the original dataset, resulting in worse classification results.

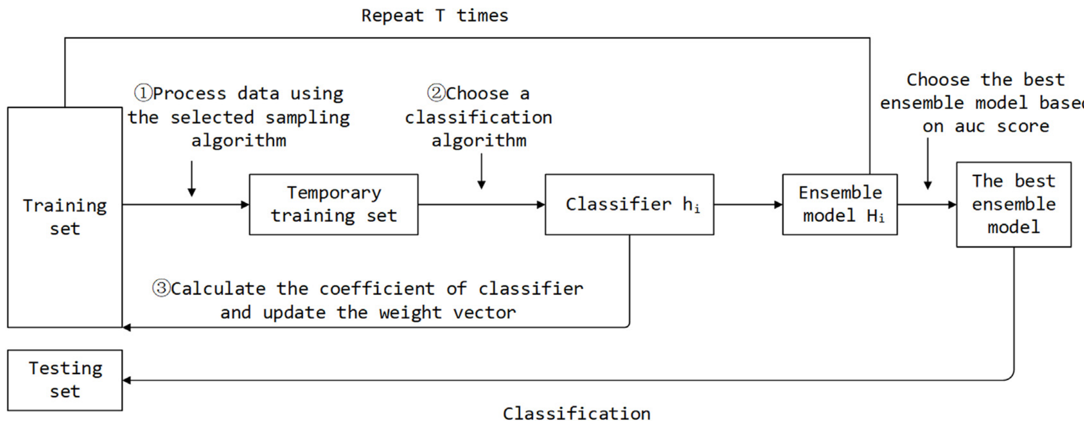

Fig. 2. The process of WHMBoost algorithm.

Algorithm 3 the weighted hybrid ensemble method

Require: Set S of example (x_1,y_1), ..., (x_m,y_m) where $x_i \in X \subseteq R^n$ and $y_i \in Y = \{-1, +1\}$(+1: positive or minority class, -1: negative or majority class)
Sampling method weight set, sampleWeightSet
Basic classifier weight set, baseClassifierWeightSet
Number of iterations, T
Number of neighbours used in SMOTE, k
Scale factor, ratio

Ensure: WHMBoost is built

initialize weights, $W_0(i) \leftarrow \frac{1}{m}$ for $i = 1, 2, ..., m$
for $i = 1$ to T **do**
  According to *sampleWeightSet*, select a sampling algorithm from the *sampleMethodSet*, and use this sampling algorithm to generate a temporary training set S_i from the original training set S.
  $w_i^{'}(j) \leftarrow w_i(k)$ if $S_i(j) == S(k)$ else $\frac{1}{m}$, for $i = 1, 2, ..., m$
  According to the *baseClassifierWeightSet*, a classification algorithm is selected from the *baseClassifierSet*. A base classifier h_i is trained using S_i and weights $w_i^{'}$.
  Calculate pseudo-loss of h_i, e_i=$\sum_{j=1}^{m} w_i(j)*I(h_i(x_j) \neq y_j)$
  α_i=$\frac{1}{2}$*ln $\frac{1-e_i}{e_i}$
  Update w_i: $w_{i+1}(j)$=$w_i(j)$*exp$(-\alpha_i$*y_j*$h_i(x_j))$
  Normalize w_{i+1}: Let $Z_i = \sum_j w_{i+1}(j)$, $w_{i+1}(j) = \frac{w_{i+1}(j)}{Z_i}$
  A ensemble model: $H_i(x)$=$argmax_{y \in Y} \sum_{t=1}^{i} \alpha_t * h_t(x, y)$
  score = roc_auc_score(H_i, X_{test}, Y_{test})
  if $score_{best}$ < score **then**
    $score_{best}$ = score
    H_{test} = H_i
  end if
end for
return H_{best}

Rayhan et al. [21] proposed a method called CUSBoost that combines cluster-based undersampling and Adaboost. Cluster-based undersampling can effectively avoid discarding important instances during the sampling process, and it can also effectively solve the problem of imbalance within the class. CUSBoost is more complicated than RUSBoost, but its benefits become obvious when the imbalance rate of the dataset is high. In their experiments, only 13 datasets were used and the evaluation method only used AUC, which could not effectively prove that their method was better than others.

Gong et al. [25] proposed a random hybrid sampling boosting algorithm called RHSBoost. Under the boosting algorithm, they used a hybrid sampling technique combining random undersampling and ROSE sampling [29]. After random undersampling and ROSE sampling processing, the size of the obtained dataset is the same as that of the original dataset, but there is a balance between the majority class and the minority class. RHSBoost makes the data distribution more balanced without increasing the number of instances in the dataset, does not increase the training time of the model, and avoids the risk of overfitting. However, it utilizes two sampling methods sequentially, resulting in an increase in the complexity of the method and the time required for the entire process.

Rayhan et al. [26] presented a boosting algorithm composed of multiple estimators for classification tasks, called MEBoost. In the framework of the boosting algorithm, they used two weak classifiers, a decision tree classifier and an extra tree classifier. In each iteration, either the decision tree classifier or the extra tree classifier is selected as its weak estimator. And in the training process, it will discard some classifiers that get relatively poor scores according to AUC. MEBoost combines the advantages of two base classifiers to make the ensemble model more robust. Nevertheless their algorithm does not use sampling methods, and the performance of models trained on unbalanced data sets may still be poor. And the single evaluation method and the small number of datasets are also the main problems.

In addition to the ensemble methods mentioned above, Li et al. [30] also proposed an AdaBoost algorithm with an SVM-based component classifier, which is called AdaBoostSVM. It has better generalization performance than SVM on imbalanced datasets. Sun et al. [31] also presented a novel ensemble method for classifying unbalanced data. The experimental results show that their method is better than the previous method for dealing with unbalanced data. DataBoost-IM algorithm [32], EUSBoost algorithm [33], and dynamic AdaBoost algorithm [34] using 10 different estimators have also been presented to solve the data imbalance problem in classification problems.

2.2. Performance metrics for imbalanced data

The *ROC curve* is a popular performance assessment method that plots true positive rates over false positive rates, creating a visual graph that describes the trade-offs between correctly classified positive samples and incorrectly classified negative samples. *AUC* is considered to be a more robust classification performance metric, independent of the imbalance rate of the dataset, and can be used to compare performance between models. *Precision* is used to measure the proportion of true positive samples

Table 1
Results of the illustrated experiment.

Examples	Features	IR	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	RB
1000	8	1.46	0.8879	0.9163	0.93	0.9355	0.9351	0.9352	0.94	0.9412	0.9384	**0.9425**	0.939
1000	8	1.82	0.939	0.9464	0.9549	0.9567	0.9554	0.9553	0.9573	0.9592	0.9607	0.9595	**0.9619**
1000	8	2.34	0.9264	0.9608	0.9671	0.9697	0.974	0.975	**0.9769**	0.9754	0.9761	0.976	0.9739
1000	8	2.92	0.9236	0.9474	0.9498	0.9506	0.951	0.9574	0.9583	**0.9616**	0.9588	0.9589	0.9589
1000	8	3.95	0.9394	0.96	0.9654	0.9688	0.9706	0.9706	0.9741	**0.9781**	0.9739	0.9742	0.9767
1000	8	5.62	0.9317	0.9635	0.9699	0.9722	0.9714	0.9749	0.9716	0.9707	**0.9756**	0.9718	0.9712
1000	8	5.94	0.8329	0.902	0.9352	0.9386	0.9519	0.9559	**0.9586**	0.9581	0.9553	0.9515	0.9475
1000	8	6.58	0.8753	0.9421	0.9478	0.9577	0.9608	0.9605	0.9664	**0.9668**	0.9624	0.9649	0.9587
1000	8	7.2	0.7675	0.7931	0.7985	0.8053	**0.8127**	0.8121	0.8081	0.8105	0.8112	0.8088	0.7965
1000	8	8.8	0.9284	0.93	0.9308	0.9465	0.948	0.9594	0.9553	0.9597	**0.9645**	0.9597	0.9563
1000	8	17.18	0.7956	0.8803	0.9096	0.9026	0.9192	**0.9249**	0.9166	0.9193	0.9228	0.9147	0.8997
2000	12	1.49	0.9115	0.9385	0.9436	0.9517	0.9561	0.9603	0.9598	0.9657	0.9673	**0.9678**	0.9672
2000	12	1.83	0.8347	0.8664	0.8692	0.8767	0.8779	0.8835	0.8862	0.8869	0.8849	**0.8908**	0.8888
2000	12	2.28	0.9751	0.9785	0.9802	0.9814	0.9841	0.9839	0.9837	0.9857	0.9837	0.9835	**0.9873**
2000	12	2.93	0.7955	0.8198	0.8297	0.8388	0.8529	0.8542	0.8649	0.8671	**0.8681**	0.8638	0.8586
2000	12	3.95	0.8664	0.9205	0.9379	0.9522	0.9635	0.9646	0.9689	0.9699	0.9731	0.9763	**0.977**
2000	12	6.27	0.7319	0.8049	0.8155	0.831	0.8367	0.8328	0.8409	0.849	**0.8568**	0.8531	0.8424
2000	12	7.3	0.7725	0.8354	0.8599	0.876	0.8823	0.9018	0.9081	0.9105	0.9146	**0.9204**	0.9139
2000	12	7.81	0.8398	0.9208	0.954	0.9586	0.9603	0.9622	0.9672	**0.9696**	0.9691	0.967	0.9678
2000	12	8.43	0.8046	0.852	0.8856	0.9077	0.9179	0.9243	0.9266	0.9198	0.9225	**0.9307**	0.9254
2000	12	9.99	0.8396	0.8624	0.8633	0.88	0.8845	0.8902	0.8958	0.8914	0.8942	**0.8988**	0.8911
2000	12	18.05	0.7783	0.8698	0.8933	0.912	0.9186	0.9211	0.9089	0.9239	0.9174	**0.9241**	0.9207
2500	14	1.5	0.8455	0.883	0.8925	0.9035	0.9041	0.9028	0.905	0.9074	0.9104	**0.9111**	0.9087
2500	14	1.84	0.8636	0.8888	0.9051	0.9094	0.9176	0.926	0.9346	0.931	0.9383	0.9368	**0.9398**
2500	14	2.31	0.8916	0.906	0.9173	0.9211	0.928	0.9329	0.9439	0.9481	0.9491	**0.9513**	0.9469
2500	14	2.97	0.827	0.8679	0.8766	0.8814	0.8915	0.8929	0.902	0.9073	0.9084	**0.9125**	0.9087
2500	14	3.97	0.8381	0.8699	0.8803	0.8824	0.8863	0.8934	0.8911	0.8958	0.8977	**0.9008**	0.8984
2500	14	5.48	0.8095	0.8848	0.8973	0.9053	0.9075	0.9085	0.9118	0.913	**0.918**	0.9134	0.9131
2500	14	5.89	0.7665	0.8248	0.8423	0.8521	0.8605	0.8713	0.8762	0.8864	0.889	**0.8938**	0.8858
2500	14	9.64	0.8673	0.9023	0.9117	0.9223	0.9462	0.9443	0.9481	0.949	**0.9553**	0.9531	0.9542
2500	14	10.74	0.7474	0.8101	0.8175	0.8351	0.8408	0.8558	0.8516	**0.8615**	0.86	0.8577	0.8457
2500	14	12.51	0.7856	0.8371	0.8487	0.8654	0.8828	0.8877	0.8983	0.9043	**0.9165**	0.9157	0.9139
2500	14	14.53	0.7693	0.8481	0.8584	0.8805	0.8927	0.8936	0.8982	0.9019	0.9016	**0.9129**	0.9085
2500	14	42.1	0.7006	0.7795	0.8351	0.8437	0.8366	0.8544	0.8604	0.8677	0.8728	**0.8735**	0.8729
2500	14	72.53	0.6196	**0.7745**	0.7533	0.7551	0.7408	0.7518	0.7397	0.7536	0.7463	0.6955	0.7231
3000	16	1.49	0.8582	0.8847	0.9056	0.9145	0.9169	0.916	0.9216	0.9234	**0.9236**	0.9233	0.9169
3000	16	1.87	0.8903	0.8961	0.9014	0.9128	0.9187	0.9274	0.9361	0.9393	0.9376	**0.9438**	0.9399
3000	16	2.96	0.9301	0.9507	0.9593	0.9593	0.9636	0.967	0.9693	0.97	0.9728	**0.9736**	0.9715
3000	16	3.93	0.8636	0.8865	0.8913	0.8985	0.9102	0.9138	0.9193	**0.9252**	0.9236	0.9243	0.9203
3000	16	5.52	0.8272	0.8988	0.9326	0.9461	0.9535	0.956	0.9618	0.9584	**0.9628**	0.9613	0.9602
3000	16	6.52	0.8486	0.9088	0.9273	0.9335	0.9493	0.9495	0.9502	0.9531	0.9571	**0.9583**	0.9528
3000	16	7.88	0.7828	0.8608	0.8904	0.9018	0.9137	0.9186	0.921	0.9255	0.9305	**0.9365**	0.9362
3000	16	8.65	0.8091	0.8804	0.9002	0.911	0.9164	0.9277	0.9323	0.933	0.9323	**0.9341**	0.93
3000	16	37.96	0.6556	0.7736	0.7827	0.8003	0.8081	**0.8125**	0.8077	0.7922	0.8049	0.7863	0.7771
3000	16	74.0	0.6319	0.715	0.7169	0.7105	0.728	0.742	0.7152	**0.7436**	0.7324	0.7296	0.7391

among the samples predicted to be positive by the model, and is defined as $\frac{TP}{TP+FP}$. *Recall* or *TPR* is used to measure the proportion of samples that are predicted to be positive by the model among true positive examples, and is defined as $\frac{TP}{TP+FN}$. *Selectivity* or *TNR* represents the proportion of negative examples predicted by the model among true negative examples, and is defined as $\frac{TN}{TN+FP}$. *F-Measure* combines *precision* and *recall*. It is a trade-off between precision and recall and is defined as $\frac{2*recall*precision}{recall+precision}$. *Geometric Mean* measures classification performance by combining *TPR* and *TNR* and is defined as $\sqrt{TPR * TNR}$. In the above, TP represents the number of True Positives, FP represents the number of False Positives, TN represents the number of True Negatives and FN represents the number of False Negatives. In this paper, we choose *AUC*, *F-Measure* and *Geometric Mean* as the classification performance measures which are commonly used performance measures for data imbalance.

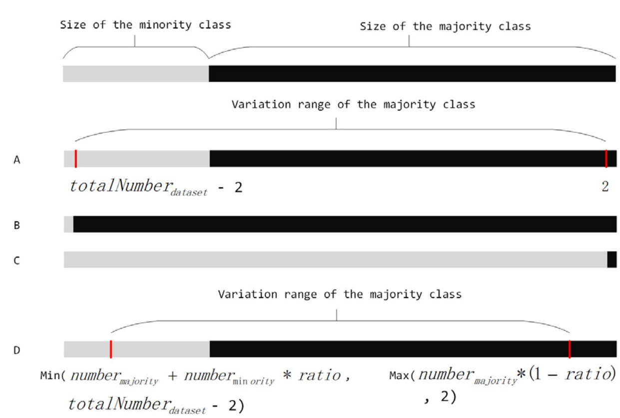

Fig. 3. Examples of random balance and adjustable random balance.

3. Proposed algorithm

3.1. Adjustable random balance

In the proposed algorithm, we used the random balance algorithm proposed by Díez-Pastor et al. [24] and improved it so that better results can be achieved. In the original random balance algorithm, the proportion between the majority class and the minority class was chosen randomly. The maximum number of instances in a certain class can be equal to $totalNumber_{dataset} - 2$ and the minimum number is equal to 2, as shown in Fig. 3A. Selecting the ratio between classes randomly within such a large

Table 2
Average ranks(AUC) of the illustrated experiment.

Scale factor	Average rank
90%	2.6667
80%	2.8444
70%	3.2111
RB	3.9778
60%	4.5778
50%	5.3
40%	6.3333
30%	7.5667
20%	8.7444
10%	9.7778
0%	11.0

Table 3
Characteristics of the data sets.

Data set	Attr	Ins	IR	Source
pima	8	768	1.87	KEEL
yeast1	8	1484	2.46	KEEL
vehicle2	18	846	2.88	KEEL
vehicle1	18	846	2.9	KEEL
vehicle3	18	846	2.99	KEEL
vehicle0	18	846	3.25	KEEL
segment0	19	2308	6.02	KEEL
yeast3	8	1484	8.1	KEEL
page-blocks0	10	5472	8.79	KEEL
vowel0	13	988	9.98	KEEL
abalone9–18	8	731	16.4	KEEL
yeast-1-4-5-8_vs_7	8	693	22.1	KEEL
yeast4	8	1484	28.1	KEEL
yeast-1-2-8-9_vs_7	8	947	30.57	KEEL
yeast5	8	1484	32.73	KEEL
yeast6	8	1484	41.4	KEEL
abalone19	8	4174	129.44	KEEL
yeast-0-2-5-7-9_vs_3-6–8	8	1004	9.14	KEEL
yeast-0-2-5-6_vs_3-7-8–9	8	1004	9.14	KEEL
flare-F	11	1066	23.79	KEEL
car-good	6	1728	24.04	KEEL
car-vgood	6	1728	25.58	KEEL
kr-vs-k-zero–one_vs_draw	6	2901	26.63	KEEL
winequality-red-4	11	1599	29.17	KEEL
kr-vs-k-three_vs_eleven	6	2935	35.23	KEEL
abalone-17_vs_7-8-9–10	8	2338	39.31	KEEL
winequality-white-3_vs_7	11	900	44.0	KEEL
abalone-19_vs_10-11-12–13	8	1622	49.69	KEEL
kr-vs-k-zero_vs_eight	6	1460	53.07	KEEL
winequality-white-3-9_vs_5	11	1482	58.28	KEEL
abalone-20_vs_8-9–10	8	1916	72.69	KEEL
segment	19	2310	6.0	HDDT
satimage	36	6430	9.29	HDDT
phoneme	5	5404	2.41	HDDT
page	10	5473	8.77	HDDT
pendigits	16	10992	8.63	HDDT
ism	6	11180	42.0	HDDT
compustat	20	13657	25.26	HDDT
estate	12	5322	7.37	HDDT
oil	49	937	21.85	HDDT

range can introduce greater diversity and obtain some excellent results. However, in some extreme cases, the number of instances of the minority class may become extremely small, as shown in Fig. 3B. There may be cases where the minority class is over-represented, as shown in Fig. 3C. The performance of the classifiers trained on such a data distribution must be very poor.

In view of the situation described above, we propose an adjustable random balance algorithm. We use a scale factor to control the range in which the number of class instances can vary, so that the number of instances of a class is not too small or too large, as shown in Fig. 3D. With this scale factor, we can ensure that each class retains a certain number of instances, and by doing so we can get better results than the original random balance.

The adjustable random balance algorithm is shown in Algorithm 2. The most important of these is the scale factor *ratio* which can be used to control the range of random changes in the number of class instances. As shown in lines 6 to 13 in the algorithm, it can reach a maximum of $number_{majority} + number_{minority} * ratio$, but cannot exceed $totalNumber_{dataset} - 2$. It can be at least equal to $number_{majority} * (1 - ratio)$, but not lower than 2. This can ensure that in the end, both the majority class and the minority class have certain instances for classification purpose. The other parts are consistent with the original random balance algorithm, as shown in Algorithm 1.

3.2. The weighted hybrid ensemble method

Most of the previously proposed ensemble algorithms use a sampling method or a base classifier. Our intuition is that different sampling methods and base classifiers have different advantages, and the way of mixing multiple sampling methods and multiple base classifiers can make these sampling methods and base classifiers complement each other to produce a better classification performance. Therefore, in this paper, a weighted hybrid ensemble method called WHMBoost is proposed to solve the problem of data imbalance in binary classification. In the framework of boosting algorithms, our method combines two data sampling methods and two base classifiers. The sampling methods include random undersampling and adjustable random balance. Base classifiers include decision tree classifier and support vector machine. The tree algorithm is unstable. It can generate different trees on the same dataset, covering different subspaces, so that it can obtain better results under the ensemble algorithm. Support vector machine has good generalization ability, and it can get better classification effect on a small number of samples. each sampling method and each base classifier are assigned corresponding weights, so that they have different contributions in model training. For example, in [35], the author also the used PEO-based NNCT weight integration strategy to determine the weight coefficient of the ensemble model. The sampling method set is $sampleMethodSet = \{random\ undersampling, adjustable\ random\ balance\}$, and its corresponding weight set is $sampleWeightSet = \{p_1, p_2\}$, which means that the random undersampling is chosen as the sampling method with the probability of p_1, where $\sum_i p_i \leq 1$. There is a certain probability that the data will not be processed in any way and the original imbalance rate will be maintained. In different datasets, the weight p_i assigned to each sampling method is different, which is mainly considered that the advantages of different sampling methods on different datasets are different. In the adjustable random balance, the scale factor can be adjusted according to the imbalance rate of datasets to ensure that the majority class and the minority class have enough data to participate in model training. Similarly, the base classifier set is $baseClassifierSet = \{decision\ tree\ classifier, support\ vector\ machine\}$, and its corresponding weight set is $baseclassifierWeightSet = \{p_1, p_2\}$, where $\sum_j p_j = 1$. By doing so, they can participate in the training process with a given weight during the model training process, so that the model can obtain the best results. The biggest difference between our proposed method and the previous ensemble method is that we combine two sampling methods and two base classifiers, and let them participate in model training with a given weight, so that they can complement each other and produce better Performance. The pseudo-code of the proposed weighted hybrid ensemble method is shown in Algorithm 3.

The process of WHMBoost algorithm is shown in Fig. 2. It can be described as follows. First, a weight $w_0(i)$ is initialized for each instance in the dataset. During each iteration, according to *sampleWeightSet*, the algorithm selects a sampling method

Table 4
Scores from the proposed method and other ensemble methods according to AUC.

Dataset	AdaBoost	RUSBoost	RBBoost	RHSBoost	SMOTEBoost	CUSBoost	MEBoost	WHMBoost
pima	0.6334	0.6605	0.5966	0.633	0.664	0.6508	0.6564	**0.6717**
yeast1	0.7571	0.7754	0.7582	0.4933	0.7698	0.773	0.7694	**0.7976**
vehicle2	0.8302	0.9283	0.9378	0.7615	0.876	0.9062	0.9215	**0.9399**
vehicle1	0.7472	0.7863	0.7663	0.4915	0.7633	0.7703	0.796	**0.8215**
vehicle3	0.7085	**0.7762**	0.7189	0.536	0.7534	0.7665	0.7592	0.7728
vehicle0	0.9159	0.9586	0.9706	0.8775	0.9397	0.9493	0.9713	**0.9763**
segment0	0.9268	0.9391	0.9899	0.9283	0.9357	0.9813	0.9786	**0.9975**
yeast3	0.8744	0.9298	0.9425	0.6649	0.92	0.9176	0.9204	**0.9557**
page-blocks0	0.8782	0.9512	0.9562	0.9313	0.9356	0.9444	0.9489	**0.9654**
vowel0	0.886	0.9715	0.9718	0.9318	0.954	0.9642	0.9538	**0.9837**
abalone9–18	0.6095	0.7848	0.7906	0.667	0.7692	0.7004	0.6631	**0.8485**
yeast-1-4-5-8_vs_7	0.5803	0.5929	0.5719	0.4919	0.5874	0.5919	0.6036	**0.6809**
yeast4	0.7305	0.87	0.8636	0.6153	0.8401	0.7962	0.8042	**0.8801**
yeast-1-2-8-9_vs_7	0.6028	0.698	0.6735	0.4929	0.646	0.6664	0.6647	**0.7427**
yeast5	0.8452	0.9836	0.9692	0.975	0.9323	0.946	0.9593	**0.9847**
yeast6	0.8518	**0.9718**	0.8919	0.8446	0.8794	0.8676	0.8858	0.921
abalone19	0.6759	0.7544	0.7469	0.6712	0.7075	0.7258	0.7227	**0.7737**
yeast-0-2-5-7-9_vs_3-6–8	0.8717	0.912	0.9027	0.9077	0.9007	0.9055	0.9216	**0.9259**
yeast-0-2-5-6_vs_3-7-8–9	0.7511	0.7999	0.7773	0.605	0.7794	0.7877	0.8076	**0.8281**
flare-F	0.8015	0.8515	0.8599	0.7391	0.8685	0.8312	0.8571	**0.89**
car-good	0.7601	0.8704	0.8514	0.7806	0.777	0.8501	0.8735	**0.9336**
car-vgood	0.8831	0.9702	0.9377	0.9213	0.9026	0.9237	0.969	**0.9816**
kr-vs-k-zero–one_vs_draw	0.7505	0.983	0.9879	0.9725	0.9764	0.9457	0.9772	**0.9892**
winequality-red-4	0.5873	0.668	0.5527	0.6595	0.5981	0.5859	0.6149	**0.7252**
kr-vs-k-three_vs_eleven	0.9573	0.994	0.9808	0.9715	0.9481	0.9642	0.9831	**0.9974**
abalone-17_vs_7-8-9–10	0.7311	0.8661	0.7984	0.7702	0.7613	0.7728	0.8093	**0.8782**
winequality-white-3_vs_7	0.6039	0.8021	0.729	0.6621	0.7281	0.6843	0.6715	**0.8035**
abalone-19_vs_10-11-12–13	0.544	0.6103	0.5985	0.5693	0.5712	0.5849	0.5878	**0.6746**
kr-vs-k-zero_vs_eight	0.6679	0.9738	0.9184	0.8689	0.9137	0.8164	0.8824	**0.98**
winequality-white-3-9_vs_5	0.5246	0.5823	0.524	0.5945	0.5567	0.5592	0.5753	**0.6011**
abalone-20_vs_8-9–10	0.6237	0.8021	0.7929	0.7237	0.7225	0.7223	0.6416	**0.8737**
segment	0.9187	0.9423	0.9887	0.9076	0.932	0.9843	0.9809	**0.9982**
satimage	0.8535	0.8879	0.8959	0.8407	0.8708	0.8753	0.9012	**0.9157**
phoneme	0.8038	0.8153	0.8297	0.7974	0.8082	0.8127	0.8405	**0.8669**
page	0.8783	0.9486	0.9564	0.9342	0.9375	0.947	0.9483	**0.9642**
pendigits	0.9063	0.9552	0.9654	0.9057	0.9361	0.9224	0.9518	**0.9732**
ism	0.8298	0.8796	**0.8958**	0.8388	0.8677	0.8634	0.8679	0.8897
compustat	0.7448	0.8102	0.7799	0.6893	0.7888	0.7911	0.8022	**0.8232**
estate	0.5939	0.623	0.5513	0.5019	0.6035	0.6144	0.6041	**0.6231**
oil	0.711	0.85	0.8134	0.6497	0.8293	0.8467	0.7662	**0.853**

from *sampleMethodSet* and uses this sampling method to generate a new dataset S_i. The weight of each instance in the original dataset remains unchanged. the artificially generated samples are assigned a new weight, $\frac{1}{m}$. Then, according to the *baseClassifierWeightSet*, the WHMBoost algorithm selects a classifier from the *baseClassifierSet*, and the selected classifier is trained using S_i and weights w_i'. The pseudo-loss of the base classifier h_i is calculated from $\sum_{j=1}^{m} w_i(j)^* I(h_i(x_j) \neq y_j)$. According to e_i, a weight α_i occupied by the base classifier in the ensemble model is calculated. The weight vector of data instances is updated so that the weight of misclassified instances increases and the weight of correctly classified instances decreases. In order to discard the base classifier with poor performance, the ensemble model with the best auc score is saved as H_{best}. When the iteration is over, H_{best} is the final ensemble model.

It needs to be emphasized here. In the proposed algorithm, we choose the best ensemble model based on AUC. However, in actual applications, different applications may have different requirements for the model. At this time, in order to obtain the best model, you may need to choose a specific evaluation method according to specific need.

4. Experimental study

4.1. An illustrated experiment

In order to find the scale factor in the adjustable random balance that can produce the best effect, we did an illustrated experiment with the generated data. We modify the RB-Boost algorithm proposed in [24] and replace the random balance in it with our improved adjustable random balance. We call this new algorithm ARB-Boost. ARB-Boost is the same as RB-Boost except that it uses the adjustable random balance to change the data distribution. We used *make_classification* in *sklearn* to generate binary classification data with different imbalance rates. In the settings, the value of *n_informative* is half of *n_features*, the value of *n_redundant* is half of *n_informative*, and the value of *n_clusters_per_class* is 1. By modifying the value of *n_samples* in the parameters of *make_classification*, we can obtain any number of binary classification datasets. Similarly, we can also manually set the value of *weights* in the parameters of *make_classification*, so that we can obtain a dataset with the expected imbalance rate. The scale factors of the adjustable random balance are set to 0, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%. When the scale factor is equal to 100%, it is the original random balance. We use RB to represent the case where the scale factor is 100%. In this illustrated experiment, we chose AUC as the performance metric and conducted experiments on 45 generated data sets. Each experiment was repeated fifty times, and the result was the average of fifty experiments. The experimental results are shown in Table 1.

From the experimental data, we can see that 35 of the best experimental results appear in 70%, 80% and 90%, accounting for 77.78% of the total datasets. When the imbalance rate is low, a high scale factor is most likely to achieve the best results, such as 80% or 90%. When the imbalance rate is high, a relatively low scale factor is most likely to achieve the best results. The scale factor ranges from 0% to 100% (RB is equivalent to the case where

Table 5
Scores from the proposed method and other methods according to F1.

Dataset	AdaBoost	RUSBoost	RBBoost	RHSBoost	SMOTEBoost	CUSBoost	MEBoost	WHMBoost
pima	0.3346	**0.5543**	0.3976	0.5093	0.5498	0.5543	0.2478	0.5409
yeast1	0.5644	**0.6004**	0.5227	0.0	0.5942	0.5635	0.4264	0.5973
vehicle2	0.6425	0.7776	0.7049	0.5526	0.6605	0.7144	0.6242	**0.8058**
vehicle1	0.1345	0.579	0.4917	0.184	0.5621	0.5593	0.0371	**0.5841**
vehicle3	0.2145	0.5547	0.4551	0.2438	0.5368	0.5491	0.1237	**0.5661**
vehicle0	0.8566	0.7844	0.8139	0.6611	0.7694	0.7555	0.8555	**0.8664**
segment0	0.8199	0.7213	0.8142	0.6173	0.7172	0.8485	0.8313	**0.9689**
yeast3	0.7029	0.6081	0.6131	0.2583	0.6217	0.6933	0.5719	**0.7072**
page-blocks0	0.745	0.5788	0.6274	0.579	0.5471	0.743	0.7314	**0.7487**
vowel0	0.7244	0.676	0.7176	0.5587	0.6625	0.7319	0.7419	**0.7891**
abalone9–18	0.1883	0.2392	0.2315	0.1859	0.2395	0.295	0.1917	**0.3294**
yeast-1-4-5-8_vs_7	0.0	0.1018	0.1039	0.0	0.0976	0.0483	0.0033	**0.1261**
yeast4	0.3139	0.2179	0.3074	0.0889	0.3384	**0.4149**	0.059	0.3657
yeast-1-2-8-9_vs_7	0.162	0.1029	0.114	0.0	0.1159	**0.2513**	0.101	0.1326
yeast5	0.4417	0.532	0.605	0.4475	0.5256	0.5879	0.2168	**0.6158**
yeast6	0.0824	0.178	0.3438	0.1693	0.3362	**0.4645**	0.0138	0.3618
abalone19	**0.9959**	0.8294	0.8397	0.6533	0.8548	0.9946	0.9958	0.9898
yeast-0-2-5-7-9_vs_3-6-8	0.6914	0.6032	0.6638	0.5464	0.7323	0.7403	0.719	**0.7525**
yeast-0-2-5-6_vs_3-7-8-9	0.4376	0.4464	0.398	0.2014	0.4963	0.5215	0.4442	**0.531**
flare-F	0.1128	0.2462	0.2909	0.1953	0.2898	0.1809	0.0414	**0.2952**
car-good	0.0	0.2249	0.2175	0.1507	0.1631	0.0	0.0	**0.369**
car-vgood	0.0	0.5713	0.3899	0.27	0.3217	0.0	0.0	**0.5797**
kr-vs-k-zero-one_vs_draw	0.6365	0.5692	0.6529	0.5821	0.4732	**0.7599**	0.6492	0.7337
winequality-red-4	0.0495	0.107	0.1035	**0.159**	0.1212	0.0939	0.0314	0.1524
kr-vs-k-three_vs_eleven	0.801	0.6113	0.8639	0.7126	0.7101	**0.9081**	0.7222	0.8742
abalone-17_vs_7-8-9-10	0.2347	0.162	0.1686	0.0981	0.2047	**0.3231**	0.1672	0.2911
winequality-white-3_vs_7	0.3069	0.0857	0.139	0.1326	0.0953	0.2617	**0.3838**	0.2322
abalone-19_vs_10-11-12-13	0.0176	0.0594	0.0821	0.0459	0.0543	0.0247	0.0124	**0.1081**
kr-vs-k-zero_vs_eight	0.3802	0.2779	0.3877	0.1698	0.2487	0.5355	0.4819	**0.5789**
winequality-white-3-9_vs_5	0.0551	0.0416	0.0473	0.0795	0.0465	**0.1674**	0.0806	0.0812
abalone-20_vs_8-9-10	**0.3455**	0.0909	0.1068	0.0654	0.077	0.3363	0.3324	0.3
segment	0.8196	0.7261	0.7955	0.6198	0.7142	0.856	0.8354	**0.9634**
satimage	0.0624	0.4793	0.4631	0.3561	0.48	**0.5508**	0.0048	0.544
phoneme	0.6191	0.6412	0.6091	0.6338	0.6408	0.6153	0.5667	**0.6808**
page	0.7583	0.5749	0.6493	0.564	0.5542	0.7405	0.7338	**0.7647**
pendigits	0.7319	0.5798	0.6381	0.5764	0.6223	0.7193	0.7307	**0.7412**
ism	0.4009	0.236	0.2236	0.2326	0.2761	**0.4491**	0.0857	0.4152
compustat	0.008	0.1841	0.1781	0.2114	0.1784	0.0012	0.0023	**0.2172**
estate	0.0205	**0.2614**	0.131	0.0056	0.2348	0.0339	0.0261	0.223
oil	0.1484	0.2346	0.2917	0.1652	0.2664	**0.2934**	0.0514	0.2597

the scale factor is 100%) has a trend of rising first and then falling. In most cases, the highest value occurs when the scale factors are 70%, 80%, and 90%.

In order to make a better comparison between different scale factors, we use average ranks [36]. As in [24], in the same way, for a given dataset, the methods are sorted from best to worst. The best method assigns rank 1, and subsequent ranks increase by 1. If they are in the same position, average rank is assigned to them. Finally, average ranks of all datasets are obtained. As shown in Table 2, when the scale factor is equal to 90%, the minimum average ranks of 2.6667 are obtained, which is similar to the average ranks of 80%. 100% or the original random balance is in the fourth position. In summary, the adjustable random balance we proposed is an improvement on the original random balance. It has better adaptability to datasets with different imbalance rates and can obtain satisfactory results. Moreover, the adjustable random balance can ensure that the generated dataset has enough instances of the majority class and the minority class, so as not to have a bad effect on model training, but the random balance may cause instances of the majority class and the minority class to become few or many. Even when the data set is extremely unbalanced, by adjusting the scale factor, the adjustable random balance can still achieve good results. In the following experiments we set the scale factor to 80%.

4.2. Another experiment and analysis

4.2.1. Dataset

In this section, we collected a total of 40 datasets with different imbalance ratio to evaluate our presented method and other ensemble methods. A part of the datasets required for our experiments comes from the Knowledge Extraction based on Evolutionary Learning tool, referred to as KEEL [37]. KEEL is open source software that supports data management and experimental designers. It pays special attention to solving data mining problems based on evolutionary learning and soft computing technologies, including regression, classification, clustering, and pattern mining. From KEEL, we collected a total of 31 datasets with an imbalance rate from 1.87 to 129.44. Another part of the datasets comes from the unbalanced binary classification data used in [38]. We collected a total of 9 datasets with an imbalance rate from 2.41 to 42. The datasets used in the experiment are shown in Table 3.

In Table 3, the first column represents the name of the datasets, the second column represents the number of attributes, the third column represents the number of instances, and the fourth column represents the imbalance rate between classes (It can be defined as $\frac{the\ number\ of\ instances\ in\ the\ majority\ class}{the\ number\ of\ instances\ in\ the\ minority\ class}$), the last column indicates the source of the data set, either KEEL or HDDT.

4.2.2. Experimental results and analysis

In each experiment, the dataset is randomly divided into the training set and the test set, where the training set accounts for 70% of the original data. In order to maintain the effect of class imbalance, the ratio of the majority class to the minority class in the training set is the same as the ratio of the majority class to the minority class in the original data. For experiments run on the same dataset, all ensemble models have the same number of

Table 6
Scores from the proposed method and other methods according to gmean.

Dataset	AdaBoost	RUSBoost	RBBoost	RHSBoost	SMOTEBoost	CUSBoost	MEBoost	WHMBoost
pima	0.4313	**0.6152**	0.4272	0.603	0.6016	0.4892	0.3621	0.6098
yeast1	0.6906	0.7067	0.6283	0.0	0.7114	0.6816	0.5418	**0.7125**
vehicle2	0.7158	0.8588	0.8068	0.6901	0.7931	0.8372	0.6915	**0.8709**
vehicle1	0.2225	0.7261	0.6315	0.2365	0.6945	0.6981	0.0752	**0.7284**
vehicle3	0.151	**0.7127**	0.6284	0.3823	0.6962	0.7039	0.1272	0.7098
vehicle0	0.9008	0.8899	0.8815	0.8313	0.8682	**8775.0**	0.8775	0.8932
segment0	0.9069	0.9277	0.9285	0.8561	0.9241	0.9416	0.8413	**0.9776**
yeast3	0.812	0.8726	0.8464	0.3784	0.8615	0.8507	0.6656	**0.878**
page-blocks0	0.8478	0.8883	0.8803	0.8866	0.8778	0.876	0.7871	**0.8986**
vowel0	0.795	0.9246	0.9224	0.8963	0.9204	0.8642	0.7946	**0.9311**
abalone9–18	0.3097	**0.7147**	0.6433	0.6179	0.6551	0.4766	0.3316	0.6753
yeast-1-4-5-8_vs_7	0.0066	0.5466	0.4603	0.0	0.4998	0.118	0.0	**0.5546**
yeast4	0.4605	**0.7921**	0.768	0.3171	0.7725	0.6561	0.0982	0.7818
yeast-1-2-8-9_vs_7	0.2426	0.6593	0.516	0.0	0.5955	0.4034	0.2139	**0.6605**
yeast5	0.5734	0.9419	0.8937	**0.9492**	0.8986	0.8014	0.3695	0.939
yeast6	0.1141	0.828	0.8119	0.6339	0.8159	0.6715	0.0085	**0.8364**
abalone19	0.0	**0.6944**	0.6315	0.5746	0.6847	0.0	0.0	0.6712
yeast-0-2-5-7-9_vs_3-6–8	0.7766	0.8673	0.8518	0.8685	0.8734	0.8652	0.7788	**0.8832**
yeast-0-2-5-6_vs_3-7-8–9	0.5953	0.7588	0.7031	0.3524	0.738	0.7006	0.541	**0.7642**
flare-F	0.1106	0.8191	0.7994	0.6785	**0.8377**	0.3723	0.1193	0.8255
car-good	0.0	**0.8353**	0.7452	0.7216	0.7522	0.0	0.0	0.7934
car-vgood	0.0	0.9401	0.879	0.8788	0.8776	0.0141	0.0	**0.9428**
kr-vs-k-zero–one_vs_draw	0.6999	0.9419	0.9483	0.919	0.9376	0.8068	0.7061	**0.9526**
winequality-red-4	0.0768	**0.6225**	0.4929	0.6129	0.5302	0.1772	0.096	0.5997
kr-vs-k-three_vs_eleven	0.8911	0.9624	0.9614	0.9641	0.9557	0.9495	0.7201	**0.9654**
abalone-17_vs_7-8-9–10	0.4145	**0.7333**	0.6431	0.6737	0.6537	0.4921	0.308	0.7156
winequality-white-3_vs_7	0.4203	0.6644	0.6424	0.4743	0.6708	0.5133	0.5402	**0.6846**
abalone-19_vs_10-11-12–13	0.0819	**0.5929**	0.4608	0.5277	0.4462	0.0505	0.0063	0.5616
kr-vs-k-zero_vs_eight	0.5172	**0.9221**	0.8163	0.7751	0.8656	0.6683	0.5573	0.9067
winequality-white-3-9_vs_5	0.1088	0.4968	0.43	0.4195	0.4835	0.3527	0.1245	**0.549**
abalone-20_vs_8-9–10	0.4142	**0.7445**	0.6515	0.5803	0.5925	0.4736	0.4744	0.73
segment	0.9103	0.9261	0.9288	0.8631	0.9256	0.942	0.8595	**0.9757**
satimage	0.1617	**0.8337**	0.7981	0.7647	0.8261	0.8075	0.0212	0.8286
phoneme	0.7226	0.7463	0.716	0.732	0.7488	0.7119	0.6711	**0.7836**
page	0.8531	0.8888	0.8577	0.8904	0.8788	0.8781	0.789	**0.9011**
pendigits	0.8226	0.8749	0.8813	0.878	0.869	0.8514	0.7516	**0.8955**
ism	0.5605	0.8022	0.7937	**0.8061**	0.7804	0.6605	0.1578	0.7945
compustat	0.0287	**0.754**	0.7292	0.6169	0.7476	0.0135	0.0071	0.7112
estate	0.0927	**0.5955**	0.4121	0.0553	0.5411	0.1272	0.1004	0.5233
oil	0.3002	**0.7968**	0.7071	0.4925	0.7837	0.407	0.104	0.7775

base classifiers. Each experiment is repeated fifty times, and the final result is the average of fifty experiments.

We compare the proposed ensemble method with the previously published ensemble methods (Adaboost, RUSBoost, RBBoost, RHSBoost, SMOTEBoost, CUSBoost, and MEBoost) on 40 datasets. In the *sampleWeightSet*, there are six optional weight assignments for random undersampling and adjustable random balancing, namely (0.8,0.1), (0.6,0.3), (0.45,0.45), (0.3,0.6), (0.1,0.8), (0.1,0.1). During training process of the model, there is a 10% chance that no sampling method will be used and the original data distribution will be maintained. Because the decision tree is unstable so that different trees can be generated on the same dataset, and the training speed is fast. Support vector machines are more complicated than decision trees. On the same dataset, the training speed of support vector sets is much slower than that of decision trees. So in the experiment, we tend to set a greater weight for the decision tree classifier. By observing the experimental results, we find that in most cases when the weight of decision tree classifier is assigned from 0.6 to 0.75, good results can be obtained. On some datasets, the weight of the support vector machine needs to be set larger to obtain good results.

In order to better compare our proposed method with other ensemble methods, we used average ranks used in the illustrated experiment. We also used the Hochbery test [39] to check the pairwise differences between the proposed method and other ensemble methods.

Using AUC as a performance metric, we compared the proposed method with other ensemble methods, and the performance scores are shown in Table 4. It can be seen from the experimental data that the performance of the weighted hybrid ensemble method is better than that of other ensemble methods on 37 datasets which account for 92.5% of the total datasets. Table 8 shows the average rank of our proposed method and other methods based on the AUC score. The adjusted Hochbery *p*-value of our proposed method compared to other methods is showed in Table 8, too. According to average rank, our proposed method is ranked first, followed by RUSBoost, RBBoost and MEBoost. All adjusted Hochbery *p*-value is less than 0.05, which means that with the significance of a = 0.05, our proposed method is significantly different from other methods. In order to better compare our proposed method with other methods, we give the roc curve on some datasets, as shown in Fig. 4. Both the table and the graph show that our proposed method can obtain a better AUC.

Table 5 shows the scores of our proposed method and other ensemble methods using F1 as a performance metric. On 23 datasets which account for 57.5% of the total datasets, the scores of our proposed method exceed those of other methods. Table 9 shows the average rank of our proposed method and other methods based on F1 scores and the adjusted Hochbery *p*-value of our proposed method compared to other methods. Under the F1 evaluation criteria, the average rank of our method is still the smallest. At the same time, we can also see that with F1 as the evaluation standard CUSBoost can also get better results. According to adjusted Hochbery *p*-value, we can conclude that our proposed method is different from other methods. From the above analysis, we can know that our proposed method can outperform other methods and obtain good results.

Table 7
Scores from the presented method and other single models according to auc.

Dataset	Decision tree	Extra tree	Naive Bayes	SVM	WHMBoost
pima	0.6267	0.5892	0.6439	0.6025	**0.6624**
yeast1	0.75	0.6638	0.7744	0.7782	**0.7984**
vehicle2	0.8196	0.7808	0.8759	0.4115	**0.9023**
vehicle1	0.7124	0.6919	0.7553	0.3843	**0.7602**
vehicle3	0.7064	0.6872	0.7431	0.3838	**0.7493**
vehicle0	0.9222	0.8736	0.8984	0.8421	**0.9598**
segment0	0.9252	0.8798	0.9515	0.7417	**0.9601**
page-blocks0	0.918	0.8979	**0.9627**	0.9522	0.9598
vowel0	0.9394	0.8584	0.9425	0.9461	**0.9747**
abalone9–18	0.6949	0.6189	0.7024	0.4188	**0.7564**
yeast4	0.7896	0.7645	0.8636	0.4122	**0.87**
yeast-1-2-8-9_vs_7	0.6367	0.6177	0.7561	0.2577	**0.7611**
yeast5	0.9231	0.9117	0.9825	0.7837	**0.9834**
yeast6	0.8254	0.8286	**0.9326**	0.2634	0.9221
abalone19	0.6809	0.6534	0.7417	0.4184	**0.7476**
yeast-0-2-5-6_vs_3-7-8-9	0.7546	0.7321	0.8259	0.6555	**0.8308**
flare-F	0.8166	0.7562	0.8508	0.4111	**0.8726**
car-good	0.773	0.7415	**0.9009**	0.5584	0.8999
kr-vs-k-zero–one_vs_draw	0.9509	0.8796	0.9854	**0.9883**	0.9849
winequality-red-4	0.6212	0.6065	0.6342	0.4034	**0.6459**
kr-vs-k-three_vs_eleven	0.9635	0.9195	0.9961	**0.9995**	0.9969
abalone-17_vs_7-8-9–10	0.7575	0.7483	0.7882	0.5576	**0.813**
winequality-white-3_vs_7	0.686	0.6969	0.8303	0.4766	**0.8424**
kr-vs-k-zero_vs_eight	0.8571	0.8034	0.939	0.7929	**0.9705**
abalone-20_vs_8-9–10	0.7062	0.6297	0.7563	0.3222	**0.7971**
segment	0.9242	0.8786	0.9559	0.7461	**0.956**
satimage	0.8502	0.8337	0.888	0.8913	**0.9084**
phoneme	0.802	0.7748	0.8415	0.8755	**0.8763**
ism	0.8494	0.8247	0.8489	0.8339	**0.8805**
compustat	0.7662	0.7351	0.7669	0.7374	**0.8176**
estate	0.5926	0.5623	0.5933	0.6118	**0.6259**
oil	0.7738	0.6846	0.7695	0.4121	**0.8281**

Table 6 shows the scores of our proposed method and other methods using G-mean as a performance metric. According to the experimental data, we can know that our proposed method outperforms other methods on 21 datasets which account for 52.5% of the total datasets and obtains relatively good results. At the same time, we can also know that RUSBoost performs well under the G-mean evaluation criteria. Table 10 shows the average rank obtained by our proposed method and other methods based on the G-mean score and the adjusted Hochbery p-value of our proposed method compared to other methods. As can be seen from the table, our proposed method still has the smallest Average rank, ranking first. This is followed by RUSBoost, which is consistent with the conclusions drawn from the experimental data. Our proposed method and RUSBoost's Adjusted Hochbery p-value is 1, exceeding 0.05, which means that under Gmean as the evaluation criteria, they have no important differences and may have similar classification effects. Except for RUSBoost, our method has significant differences from other methods.

From the above analysis, we can know that there are important differences between our proposed method and other ensemble methods. Our method can get good classification performance in most cases. Although when we use Gmean as the evaluation criteria, our proposed method does not have significant differences from RUSBoost, but it cannot be said that our method is not good. After all, with AUC and F1 as the evaluation criteria, our proposed method is still different from all other methods in general, and the scores obtained are valuable.

4.3. A comparison experiment between ensemble model and single model

In order to prove that the ensemble model can indeed produce better performance than the single model, we select 32 datasets from Table 3, use AUC as the evaluation criterion and compare the proposed method with decision tree, extra tree, naive bayes and support vector machine. Before training, the data is randomly under-sampled, so that the majority class and the minority class are in balance. The number of the base classifier in the ensemble model is set to 20. The result is the average of fifty repeated experiments.

The results of the experiment are shown in Table 7. From the experimental results, it can be seen that on the 27 datasets, which account for 84.38% of the total datasets, the results obtained by the ensemble model proposed by us exceed those obtained by other single models. Table 11 shows the average rank obtained from the auc score of our ensemble model and other single models and the adjusted Hochbery p-value of our presented method relative to other single models. As can be seen from the table, the average rank of the ensemble model we proposed is 1.1875, which is the smallest compared to other single models. All adjusted Hochbery p-values are less than 0.05, which means that with the importance of a = 0.05, the ensemble model we proposed has significant differences from other single models. From the above analysis, we can know that the presented ensemble model can outperform other single models and obtain better results.

5. Conclusion

In real datasets, data distribution is almost always unbalanced. The classifier tends to over-represent the majority class, while the underrepresented minority class cannot be well classified. However, in practical applications, the minority class is usually the class we are interested in. In this case, how to correctly classify the minority class becomes a huge challenge. In this paper, in order to eliminate the impact of data imbalance and maximize the probability that the model correctly classifies instances of the minority class, we propose a weighted hybrid ensemble method for classifying binary classification datasets, called WHMBoost. This algorithm combines the advantages of multiple sampling methods and multiple base classifiers and improves the overall

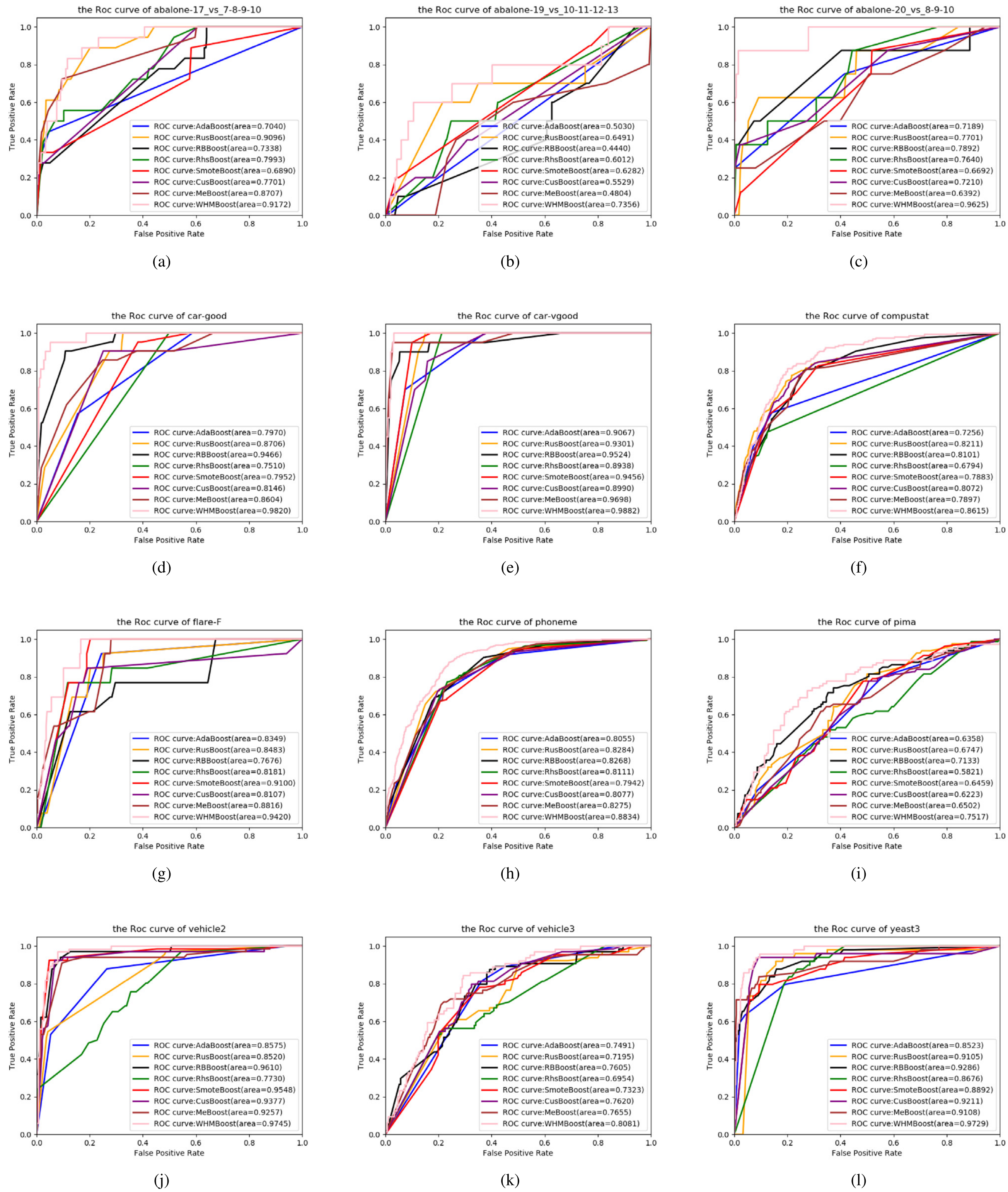

Fig. 4. Roc curve on some datasets: (a) abalone-17_vs_7-8-9-10, (b) abalone-19_vs_10-11-12-13, (c) abalone-20_vs_8-9-10, (d) car-good, (e) car-vgood, (f) compustat, (g) flare-F, (h) phoneme, (i) pima, (j) vehicle2, (k) vehicle3, (l) yeast3.

performance compared to using a single sampling method and a single base classifier. The performance of WHMBoost has been evaluated on 40 benchmark imbalanced datasets with state of the art ensemble methods like AdaBoost, RUSBoost, SMOTEBoost using AUC, F-Measure and Geometric Mean as the performance evaluation criteria. From the experimental results, our proposed method outperforms other ensemble methods and obtains excellent evaluation results. It can replace other ensemble methods to solve the data imbalance problem encountered in real scenarios.

In the future, we will use more evaluation criteria and experiment with our method and more methods on more imbalanced data sets to verify the effectiveness of our presented

Table 8
Average ranks and Hochbery test(AUC).

Algorithm	Average rank	Adjusted Hochbery *p*-value
WHMBoost	1.075	
RUSBoost	2.725	0.0181
RB-Boost	3.875	1.9118e−06
MEBoost	3.95	7.6460e−07
CUSBoost	4.9	1.1520e−11
SMOTEBoost	5.35	1.7986e−14
RHSBoost	6.825	0.0
ADABoost	7.3	0.0

Table 9
Average ranks and Hochbery test(F1).

Algorithm	Average rank	Adjusted Hochbery *p*-value
WHMBoost	1.725	
CUSBoost	3.2125	0.0463
RBBoost	4.55	1.4998e−06
RUSBoost	4.7125	2.4569e−07
SMOTEBoost	4.725	1.7282e−07
ADABoost	4.8625	3.0439e−08
MEBoost	5.725	5.6311e−13
RHSBoost	6.4875	0.0

Table 10
Average ranks and Hochbery test(gmean).

Algorithm	Average rank	Adjusted Hochbery *p*-value
WHMBoost	1.7	
RUSBoost	2.05	1
SMOTEBoost	3.55	0.0044
RBBoost	4.225	2.0134e−05
RHSBoost	5.1125	1.8616e−09
CUSBoost	5.25	2.7266e−10
ADABoost	6.7125	0.0
MEBoost	7.4	0.0

Table 11
Average ranks and Hochbery test(auc).

Algorithm	Average rank	Adjusted Hochbery *p*-value
WHMBoost	1.1875	
Naive Bayes	2.1875	0.0456
Decision Tree	3.2188	8.2993e−07
SVM	4.125	2.1494e−13
Extra Tree	4.2812	5.1070e−15

algorithm. And we will extend our ideas to the multi-class imbalance problem. We will also conduct more research on issues such as with-class imbalance and feature selection so that we can make breakthroughs in these areas.

CRediT authorship contribution statement

Jiakun Zhao: Conceptualization, Resources, Supervision. **Ju Jin:** Methodology, Investigation, Writing - original draft, Writing - review & editing. **Si Chen:** Formal analysis, Writing - original draft. **Ruifeng Zhang:** Software, Writing - original draft. **Bilin Yu:** Validation, Data curation. **Qingfang Liu:** Funding acquisition.

Declaration of competing interest

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

Acknowledgments

This work was supported by the Natural Science Foundation of China (NSFC) under Grant 11972275 and 11772242. We wish to thank the donors of the different datasets, too.

References

[1] N. Shirodkar, P. Mandrekar, R. Mandrekar, R. Sakhalkar, K.M.C. Kumar, S. Aswale, Credit card fraud detection techniques – A survey, in: 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), 2020, pp. 1–7.
[2] J.M. Johnson, T.M. Khoshgoftaar, Survey on deep learning with class imbalance, J. Big Data 6 (2019) 1–54.
[3] J. Zhang, I. Mani, KNN approach to unbalanced data distributions: A case study involving information extraction, in: Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Datasets, 2003.
[4] P.E. Hart, The condensed nearest neighbor rule (corresp.), IEEE Trans. Inform. Theory 14 (1968) 515–516.
[5] D.L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Trans. Syst. Man Cybern. 2 (1972) 408–421.
[6] I. Tomek, Two Modifications of CNN, 1976.
[7] S.-J. Yen, Y.-S. Lee, Cluster-based under-sampling approaches for imbalanced data distributions, Expert Syst. Appl. 36 (2009) 5718–5727.
[8] K.W. Bowyer, N.V. Chawla, L.O. Hall, W.P. Kegelmeyer, SMOTE: Synthetic minority over-sampling technique, J. Artificial Intelligence Res. 16 (2002) 321–357.
[9] H. Han, W. Wang, B. Mao, Borderline-SMOTE: A new over-sampling method in imbalanced data sets Learning, in: ICIC, 2005.
[10] H. He, Y. Bai, E.A. Garcia, S. Li, ADASYN: Adaptive synthetic sampling approach for imbalanced learning, in: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), 2008, pp. 1322–1328.
[11] C. Bunkhumpornpat, K. Sinapiromsaran, C. Lursinsap, Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem, in: PAKDD, 2009.
[12] J.L. Leevy, T.M. Khoshgoftaar, R.A. Bauder, N. Seliya, A survey on addressing high-class imbalance in big data, J. Big Data 5 (2018) 1–30.
[13] P. Zhou, X. Hu, P.-P. Li, X. Wu, Online feature selection for high-dimensional class-imbalanced data, Knowl.-Based Syst. 136 (2017) 187–199.
[14] T.-Y. Lin, P. Goyal, R.B. Girshick, K. He, P. Dollár, Focal loss for dense object detection, in: 2017 IEEE International Conference on Computer Vision, ICCV, 2017, pp. 2999–3007.
[15] Y. Huang, Dynamic cost-sensitive ensemble classification based on extreme learning machine for mining imbalanced massive data streams, 2015.
[16] M. Buda, A. Maki, M.A. Mazurowski, A systematic study of the class imbalance problem in convolutional neural networks, Neural Netw. 106 (2018) 249–259.
[17] W. Zhu, P. Zhong, A new one-class SVM based on hidden information, Knowl.-Based Syst. 60 (2014) 35–43.
[18] J.R. Quinlan, Bagging, Boosting, and C4.5, Vol. 1, AAAI/IAAI, 1996.
[19] N.C. Oza, S.J. Russell, Online bagging and boosting, in: 2005 IEEE International Conference on Systems, Man and Cybernetics, vol. 3, 2001, pp. 2340–2345.
[20] H. He, E.A. Garcia, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (2009) 1263–1284.
[21] F. Rayhan, S. Ahmed, A. Mahbub, M.R. Jani, S. Shatabda, D.M. Farid, CUSBoost: Cluster-based under-sampling with boosting for imbalanced classification, in: 2017 2nd International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS), 2017, pp. 1–5.
[22] N.V. Chawla, A. Lazarevic, L.O. Hall, K.W. Bowyer, Smoteboost: Improving prediction of the minority class in boosting, in: PKDD, 2003.
[23] C. Seiffert, T.M. Khoshgoftaar, J.V. Hulse, A. Napolitano, RUSboost: A hybrid approach to alleviating class imbalance, IEEE Trans. Syst. Man Cybern. 40 (2010) 185–197.
[24] J.-F. Díez-Pastor, J.J.R. Diez, C.I. García-Osorio, L.I. Kuncheva, Random balance: Ensembles of variable priors classifiers for imbalanced data, Knowl.-Based Syst. 85 (2015) 96–111.
[25] J. Gong, H. Kim, Rhsboost: Improving classification performance in imbalance data, Comput. Statist. Data Anal. 111 (C) (2017) 1–13, http://dx.doi.org/10.1016/j.csda.2017.01.00, https://ideas.repec.org/a/eee/csdana/v111y2017icp1-13.html.
[26] F. Rayhan, S. Ahmed, A. Mahbub, M.R. Jani, S. Shatabda, D.M. Farid, C.M. Rahman, MEBoost: Mixing estimators with boosting for imbalanced data classification, in: 2017 11th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), 2017, pp. 1–6.
[27] Y. Freund, R.E. Schapire, Experiments with a new boosting algorithm, in: ICML, 1996.
[28] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. System Sci. 55 (1997) 119–139.
[29] N. Lunardon, G. Menardi, N. Torelli, ROSE: a package for binary imbalanced learning, 2014.
[30] X. Li, L. Wang, E. Sung, Adaboost with SVM-based component classifiers, Eng. Appl. AI 21 (2008) 785–795.

[31] Z. Sun, Q. Song, X. Zhu, H. Sun, B. Xu, Y. Zhou, A novel ensemble method for classifying imbalanced data, Pattern Recognit. 48 (2015) 1623–1637.
[32] H. Guo, H.L. Viktor, Learning from imbalanced data sets with boosting and data generation: the databoost-IM approach, SIGKDD Explor. 6 (2004) 30–39.
[33] M. Galar, A. Fernández, E.B. Tartas, F. Herrera, EUSBoost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling, Pattern Recognit. 46 (2013) 3460–3471.
[34] E.N. de Souza, S. Matwin, Extending adaboost to iteratively vary its base classifiers, in: Canadian Conference on AI, 2011.
[35] F. Zhao, G. Zeng, K.-D. Lu, EnLSTM-WPEO: short-term traffic flow prediction by ensemble LSTM, NNCT weight integration, and population extremal optimization, IEEE Trans. Veh. Technol. 69 (2020) 101–113.
[36] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[37] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework, Multiple-Valued Log. Soft Comput. 17 (2011) 255–287.
[38] D.A. Cieslak, T.R. Hoens, N.V. Chawla, W.P. Kegelmeyer, Hellinger distance decision trees are robust and skew-insensitive, Data Min. Knowl. Discov. 24 (2011) 136–158.
[39] Y. Hochberg, A sharper Bonferroni procedure for multiple tests of significance, 1988.