Journal of Data Intelligence, Vol. 2, No. 4 (2021) 462–475
© Rinton Press

COMBINING MULTI-RATIO UNDERSAMPLING AND METRIC LEARNING FOR IMBALANCED CLASSIFICATION

TAKAHIRO KOMANIZU
Nagoya University
Nagoya, Japan
taka-coma@acm.org

In classification, class imbalance is a factor that degrades the classification performance of many classification methods. Resampling is one widely accepted approach to the class imbalance; however, it still suffers from an insufficient data space, which also degrades performance. To overcome this, in this paper, an undersampling-based imbalanced classification framework, MMEnsemble, is proposed that incorporates metric learning into a multi-ratio undersampling-based ensemble. This framework also overcomes a problem with determining the appropriate sampling ratio in the multi-ratio ensemble method. It was evaluated by using 12 real-world datasets. It outperformed the state-of-the-art approaches of metric learning, undersampling, and oversampling in recall and ROC-AUC, and it performed comparably with them in terms of Gmean and F-measure metrics.

Keywords: imbalanced classification, undersampling, ensemble, metric learning

1. Introduction

Class imbalance [15] is a crucial problem in real-world applications that degrades classification performance, especially with minority classes. Class imbalance refers to a situation with datasets in which the number of examples in a class is much larger than that in other classes. The large difference in terms of the numbers of examples causes classifiers to be biased toward the majority class. Class imbalance has been observed and dealt with in various domains, such as the clinical domain [7], economic domain [25] and agricultural domain [28], and in software engineering [26] and computer networks [12].

Resampling is an effective solution to class imbalance, and it has been widely studied [8, 19, 5, 6]. Resampling techniques can be roughly classified into two categories: oversampling (e.g., SMOTE [8] and SWIM [6]) and undersampling (e.g., EasyEnsemble [19] and RUSBoost [27]). Undersampling is a simple and powerful resampling technique for dealing with class imbalance [10]. Not only using single-shot undersampling but also combining multiple undersampled datasets in an ensemble manner have done for the problem [19, 16, 27].

A preliminary survey of various datasets on the effects of different undersampling ratios, shown in Figure 1, indicated that different undersampling ratios have different preferences toward classes. Sampling ratio refers to the ratio of the sampled majority size over the minority size. In this paper, the minority class and majority class are regarded as the positive class and negative class, respectively. A sampling ratio of 1.0 means that the majority examples are randomly selected so that the number of sampled examples equals that of the minority examples. A ratio below 1.0 means that the number of sampled majority examples is below that of the minority examples. In this paper, this is called *excessive undersampling* and

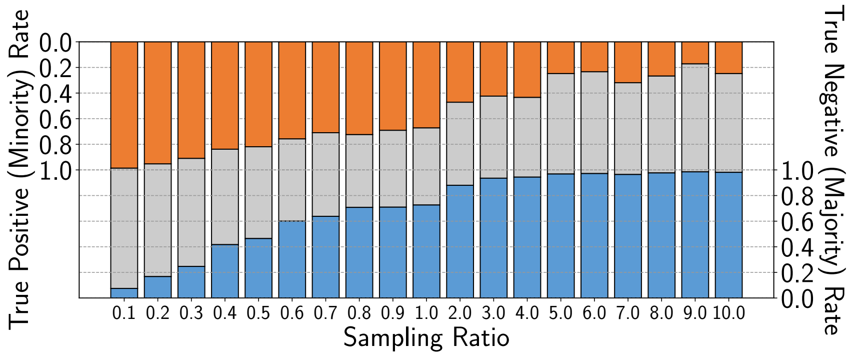

Fig. 1. Small sampling ratios prefer minority, and vice versa.

its antonym is *moderate undersampling*. The figure depicts the true positive and negative ratios for different sampling ratios in the Abalone dataset. It indicates that classifiers learned with excessively undersampled datasets favor the minority class and those by moderately undersampled datasets favor the majority class, so 1.0 may not be best balanced ratio.

This paper proposes a novel undersampling-based ensemble framework, **MMEnsemble**, that is composed of *metric learning*, *multi-ratio ensemble*, and *asset-based weighting*.

- **Metric Learning**: Metric learning methods such as LMNN [34] learn a data transformation so that instances in different classes can be distinguishable. Recent metric learning approaches [33] have shown that selecting subsets of training instances for metric learning improves the classification performance of an imbalanced classification. On the basis of this idea, in MMEnsemble, metric learning is incorporated into an undersampling-based ensemble.

- **Multi-ratio Ensemble**: When applying undersampling, the sampling ratio is an important parameter. It determines the number of drawn majority instances. A recent study [17] has shown that incorporating multiple sampling ratios in an ensemble manner improves the classification performance.

- **Asset-based Weighting**: When applying a multi-ratio undersampling-based ensemble, weak classifiers for different sampling ratios have different assets. A classifier with a large sampling ratio may correctly classify the majority class, and another classifier with a small sampling ratio may correctly classify the minority class. To capture the assets, MMEnsemble introduces a weighting scheme that weighs on classifiers that can correctly classify instances that are hard for the other classifiers to classify.

The contributions of this paper are summarized as follows.

- **MMEnsemble – a novel framework**: MMEnsemble is a framework composed of metric learning, multi-ratio undersampling-based ensemble, and asset-based weighting. This framework overcomes the weakness of metric learning regarding the class imbalance by applying undersampling beforehand, and it releases users from the burden of choosing sampling ratios in undersampling by using ensemble of base classifiers in various sampling ratios and automatic weighting schemes.

- **Superior Classification Performance**: In an experiment using 12 real-world datasets, MMEnsemble outperforms the state-of-the-art approaches, especially for recall and ROC-AUC metrics, and it performs comparably to them on Gmean and F-measure metrics. This experiment indicates that this approach can achieve higher recall scores, which would be useful for many real-world applications.

The rest of this paper is organized as follows. Section 2 introduces the related work on resampling-based approaches. Section 3 explains MMEnsemble in detail, and Section 4 shows the experimental evaluation using 12 real-world datasets. Finally, Section 5 concludes this paper.

2. Related Work: Resampling Approaches

To deal with class imbalance, there are basically three groups of approaches, namely, resampling, cost-adjustment [9], and algorithm modification [32]. Resampling is commonly used because it has shown robust performance and applicability to any classifiers. Resampling approaches can be roughly classified into two categories, namely, oversampling and undersampling.

2.1. *Oversampling-based Approaches*

A simple oversampling approach is to randomly copy minority examples so that the numbers of minority and majority examples become the same. This approach easily causes overfitting. To cope with the overfitting problem, oversampling approaches generate synthetic minority examples that are close to the minority. SMOTE [8] is the most popular synthetic oversampling method. It generates synthetic minority examples on the basis of the nearest neighbor technique. Since SMOTE does not take majority examples into consideration, the generated examples can easily overlap with majority examples. This degrades the classification performance. To overcome the weakness of SMOTE, more recent approaches have incorporated majority examples into the resampling process. SMOTE-Tomek [4] and SMOTE-ENN [5] employ data cleansing techniques including the removal of Tomek links [30] and Edit Nearest Neighbours [35]. Along this line, there are more advanced approaches (e.g., ADASYN [14], borderline-SMOTE [13], and SVM-SMOTE [23]). One of the state-of-the-art synthetic oversampling approaches is SWIM [6]. SWIM utilizes the density of each minority example with respect to the distribution of majority examples in order to generate synthetic minority examples. Comprehensive experiments by [18] were conducted to investigate a large number of SMOTE variants and to compare these variants with diverse kinds of datasets. In this investigation, PolyFitSMOTE [11] and ProWSyn [2] showed the best performances.

2.2. *Undersampling-based Approaches*

Undersampling-based approaches can be classified into three categories: example selection and boosting and bagging ensembles. Example selection is an approach to choosing majority examples that are expected to contribute to better classification. Major approaches choose majority examples hard to distinguish from minority examples. NearMiss [22] samples majority examples close to minority examples. Instance hardness [29] is a hardness property that indicates the likelihood that an example will be misclassified.

Boosting ensemble is a learning method that gradually changes majority examples. For each iteration, boosting approaches remove a part of the majority examples. BalanceCascade [19] is a boosting approach that removes correctly classified majority examples. RUSBoost [27] is a weighted random undersampling approach for removing majority examples that are likely to be classified correctly. EUSBoost [21] introduces a cost-sensitive weight modification and an adaptive boundary decision strategy to improve model performance. Trainable Undersampling [24] is the state-of-the-art in this category. It trains a classifier by reinforcement learning.

Bagging ensemble combines multiple weak classifiers, each of which is learned on individual pieces of undersampled training data in a voting manner. Ensemble of Undersampling [16] is one of the earlier bagging approaches using undersampled training data. EasyEnsemble [19] is an ensemble-of-ensemble approach that ensembles AdaBoost classifiers for each piece of undersampled training data in a bagging manner. In [10], a comprehensive experiment on boosting and bagging approaches is reported. It shows that RUSBoost and EasyEnsemble are the best performing approaches, and they outperform oversampling-based approaches. MUEnsemble [17] extends EasyEnsemble to incorporate multiple sampling ratios. DDAE [36] is the state-of-the-art bagging-based approach that takes metric learning and cost-sensitive learning into account.

The proposed method, MMEnsemble, is classified in the bagging category. A major distinction of MMEnsemble from the others (except DDAE) is that it incorporates metric learning to overcome the issue of insufficient data spaces in resampling methods. There are two major differences between DDAE and MMEnsemble. One is the control of undersampling (called *data block*); MMEnsemble undersamples data with respect to the sampling ratio, while DDAE undersamples on the basis of the number of blocks, which is not dependent on the imbalance ratio of datasets. The other is the choice of weak classifier; DDAE uses the nearest neighbor classifier, which is considered to fit metric learning, while MMEnsemble uses the AdaBoost classifier. Additionally, but importantly, DDAE has (at least) three hyper-parameters that need to be tuned, while MMEnsemble has only one, which is a much smaller parameter space.

In terms of the multi-ratio ensemble, MMEnsemble uses assets of weak classifiers that are obtained from the process of validating weak classifiers, while MUEnsemble uses a heuristic weighting (i.e., Gaussian function-based weighting). Therefore, to capture the characteristics of weak classifiers, a comprehensive hyper-parameter tuning is required. The experimental evaluation in this paper shows the superiority of the asset-based weighting scheme over the heuristic weighting in MUEnsemble.

3. MMEnsemble

Figure 2 shows an overview of MMEnsemble, which consists of three phases: the multi-ratio undersampling phase, metric learning phase, and multi-ratio ensemble phase. The first phase is imitated from MUEnsemble [17], that is, for each sampling ratio $r_i \in R$ (R is a predefined set of sampling ratios), multiple undersampled sets of instances with r_i are drawn from the training data. In the second phase, for each drawn set, metric learning is performed and a base ensemble classifier, called *MLEnsemble*, is trained using this drawn set that is transformed by the metric learning. In the last phase, given $|R|$ base classifiers from the

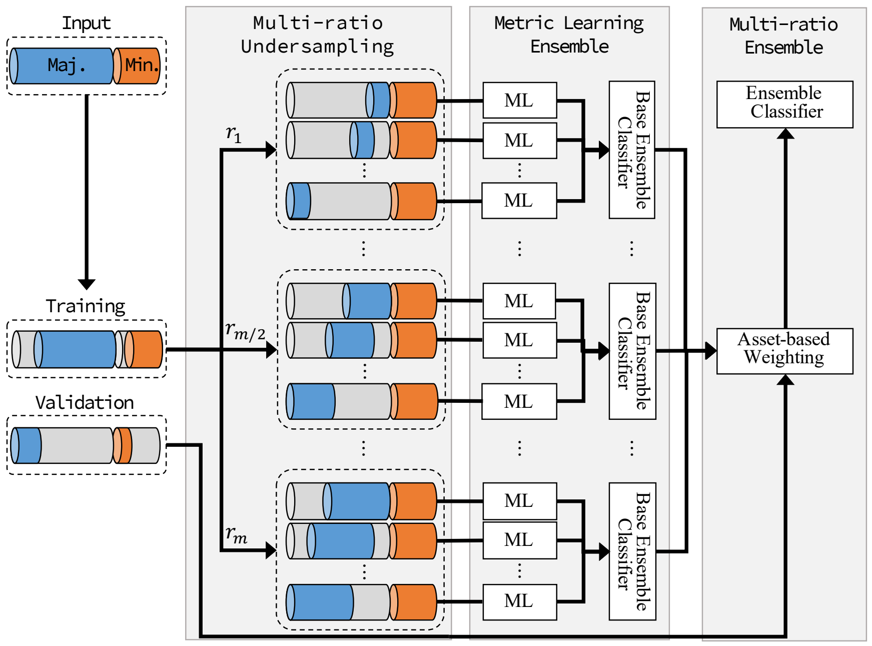

Fig. 2. MMEnsemble – Proposed Framework

Algorithm 1 MLEnsemble

```
Input: Training data $D^{(train)} = \{D_{maj}, D_{min}\}$, sampling ratio $r$, the number of weak classifiers $n$
Output: Base ensemble classifier with metric learner $C_r = ((c_1, m_1), (c_2, m_2), \ldots, (c_n, m_n))$
1: for $i = 1$ to $n$ do
2:     $D'_{maj} \leftarrow$ Randomly sample $D_{maj}$ s.t. $\frac{|D'_{maj}|}{|D_{min}|} = r$
3:     Train metric learner $m_i$ using $\{D'_{maj}, D_{min}\}$
4:     $D' \leftarrow$ Transform $\{D'_{maj}, D_{min}\}$ using $m_i$
5:     Train weak classifier $c_i$ using $D'$
6: end for
```

previous phase, the final ensemble classifier is constructed by the asset-based weighting. In the following sections, the technical details of MLEnsemble and the ensemble with the asset-based weighting are introduced.

3.1. *Base Ensemble Classifier – MLEnsemble*

MLEnsemble is a bagging classifier with metric learning. Its procedure is summarized in Algorithm 1. The training data are sampled multiple times with replacement to obtain particular sets of instances (Line 2). For each set, a metric learner is trained by using the set so that it transforms the set into a sufficient data space for distinguishing instances of different classes (Lines 3-4). Using the transformed set, a weak classifier is trained (Line 5).

3.2. *Ensemble using Asset-based Weighting*

Typical ensemble methods use the weighted voting strategy. These methods often use equal weights for all base classifiers, and they are not aware of class imbalance. In contrast, for the case of an ensemble of base classifiers in different sampling ratios, the weights of base classifiers are more sensitive, and thus need to be carefully designed. [17] showed that a heuristic weighting using a Gaussian function is superior to the equal weighting. The Gaussian-based weighting is calculated as follows.

$$W_{gauss}(r) = a \cdot \exp\left(-\frac{(r-\mu)^2}{2\sigma^2}\right), \tag{1}$$

where μ and σ^2 are tunable parameters, and a is a normalization constant such that $a = \frac{1}{\sum_{r \in R} W_{gauss}(r)}$. When $\mu = 1.0$, most of the weight is on the base classifier trained using the balanced data, and the weights gradually decrease as r increases and decrease from μ.

The heuristic weighting approach does not take the classification performances of base classifiers into consideration. There are typically some instances that can be correctly classified by only a few base classifiers. To improve the classification performance with the ensemble mechanism, base classifiers classifying such instances correctly are important. Also, these base classifiers are expected to not incorrectly classify instances that are correctly classified by the other base classifiers. In this paper, this is called an *asset* of a base classifier. Formally, given set $C = \{C_j\}_{j=1}^{s}$ of base classifiers with size s and validation set $D^{(val)}$, for each instance $(d_i, \ell_i) \in D^{(val)}$, where d_i is a feature vector, and ℓ_i is a class label of the i-th instance, the number T_i of base classifiers that correctly classify the i-th instance is obtained. That is, $T_i = |\{C_j | C_j \in C, C_j.predict(d_i) = \ell_i\}|$. This number indicates how hard (or easy) an instance is to classify. The intuition behind using this number for weighting base classifiers is that the lower the number, the more weights on a base classifier if it correctly classifies the instance. This intuition is formalized by the following formula.

$$W_{asset}(r) = a \cdot \sum_{(d_i,\ell_i) \in D^{(val)}} \delta(C_r.predict(d_i), \ell_i) \cdot T_i^{-k}, \tag{2}$$

where k is a tunable parameter for emphasizing the importance of the classifiers that correctly classify instances that other classifiers cannot, δ function is the Kronecker delta (i.e., 1 if the two arguments are equal, 0 otherwise), and a is a normalization constant such that $a = \frac{1}{\sum_{r \in R} W_{asset}(r)}$.

4. Experimental Evaluation

In this experiment, MMEnsemble was evaluated to answer the following questions.

Q1 Does MMEnsemble outperform the state-of-the-art imbalanced classification methods of metric learning, oversampling and undersampling?

Q2 Is the combination of metric learning and multi-ratio ensemble effective?

Q3 Does the asset-based weighting help improve the classification performance? and what is the effect of choice of its hyper-parameter k (Equation 2)?

4.1. *Settings*

Table 1. Datasets

ID	Name	#records	#minor	#dim	IR
D1	cm1	498	49	21	9.2
D2	kc3	458	43	39	9.7
D3	mw1	403	31	37	12.0
D4	pc1	1,109	77	21	13.4
D5	pc3	1,563	160	37	8.8
D6	pc4	1,458	178	37	7.2
D7	yeast1-7	459	30	7	14.3
D8	abalone9-18	731	42	8	16.4
D9	yeast6	1,484	35	8	41.4
D10	abalone19	4,174	32	8	129.4
D11	wine3-5	691	10	11	68.1
D12	abalone20	1,916	26	8	72.7

Datasets: The datasets for the experiment were obtained from the OpenML dataset [31] and KEEL repository [1]. Table 1 shows the total number of records (#records), the number of minority instances (#minor), dimensionality (#dim), and the imbalance ratio (IR), which is $\frac{\#major}{\#minor}$. D1-D6 were obtained from the OpenML dataset, and the rest were obtained from the KEEL repository.

Evaluation: The evaluation metrics were *Recall*, *Gmean*, F_2, and AUC. Let *TP*, *FN*, *TN*, and *FP* be the true positives, false negatives, true negatives, and false positives. $Recall = \frac{TP}{TP+FN}$ measures how many positive (minority) instances are correctly classified. $Gmean = \sqrt{Recall \cdot TNR}$ is the geometric mean of the recalls of both classes, where $TNR = \frac{TN}{TN+FP}$. $F_\beta = \frac{(1+\beta^2) Recall \cdot Precision}{Recall + \beta^2 Precision}$ is the harmonic mean of the recall and precision, where $Precision = \frac{TP}{TP+FP}$, and β determines the weight on the recall. In this experiment, β was set to 2 because the higher recalls are preferred in many real-world applications. AUC is the area under the receiver operation characteristic curve.

To accurate estimate these evaluation metric values, the experimental process was repeated 50 times. In the process, a dataset was randomly separated into 70% for training and 30% for testing, and the classifiers were trained on the training set and evaluated using the test set. The overall metric scores were the macro average of the 50 trials.

Baseline Methods: MMEnsemble was compared with the state-of-the-art methods of metric learning, resampling (oversampling and undersampling) and the ensemble approach. **IML** [33] is a state-of-the-art approach of metric learning and copes with the class imbalance. IML incorporates LMNN [34] and iteratively selects training samples to improve the data transformation. For resampling, **ProWSyn** [2] was selected for the oversampling approach on the basis of the comprehensive experiment in [18] and a preliminary evaluation on the datasets in this experiment. For the undersampling and ensemble method, **DDAE** [36] is the state-of-the-art and also includes metric learning. Since this experiment uses the same dataset as DDAE, the results of IML and DDAE were copied from the DDAE paper [36] (though, in [36], MWMOTE [3] is selected as the state-of-the-art oversampling method, [18] and the preliminary evaluation showed the superiority of ProWSyn).

To answer Q2, MMEnsemble was compared with **EasyEnsemble** [20], **MUEnsemble** [17], and MLEnsemble (in this paper), which are an undersampling-based ensemble, a multi-ratio undersampling-based ensemble, and a metric learning incorporating EasyEnsemble, respectively. The difference between EasyEnsemble and MMEnsemble shows the benefit of integrating both the metric learning and the multi-ratio ensemble. Similarly, the difference between MMEnsemble and MUEnsemble shows the benefit of metric learning to improve the performance for the imbalanced classification.

Parameters: The parameters of ProWSyn, EasyEnsemble, MLEnsemble, and MUEnsemble were set as follows. The sampling ratio in ProWSyn, EasyEnsemble and MLEnsemble was set to 1.0. The metric learning method was LMNN with the k parameter of kNN set to 3. For MUEnsemble, the predefined set R of sampling ratios is set to $\{0.2, 0.4, \ldots, 2.0\}$, and Gaussian weighting was used with parameters, μ and σ^2, of 1.0 and 0.2, where μ was fixed to 1.0 have the parameter be the same as the former methods, and the best σ^2 was experimentally explored from $\{0.1, 0.2, \ldots, 1.0\}$. For MMEnsemble, the base classifier, MLEnsemble, was set the same as above, R is the same as MUEnsemble, and k of the asset-based weighting was chosen from $\{0.1, 0.2, \ldots, 5.0\}$.

4.2. *Results*

To answer the questions, the experimental results are shown from three perspectives: an overall comparison (corr. Q1), the ablation study (corr. Q2), and a comparison over the k parameter and other weighting schemes (corr. Q3).

4.2.1. *Overall Comparison*

Table 2 showcases the metric scores of MMEnsemble with the state-of-the-art methods. In the table, the highest scores in a row are boldfaced. MMEnsemble totally outperformed IML and ProWSyn, and it outperformed DDAE in recall and AUC, and it was comparable with DDAE in terms of Gmean and F_2 metrics. It is noteworthy that MMEnsemble totally outperformed the others on the AUC metric, and it achieved almost the best performance on the recall metric. For the real-world applications, a high recall is preferable; therefore, this superiority of MMEnsemble is practically useful. On the contrary, the Gmean and F_2 scores were comparable with DDAE. On datasets, D5, D6, D8, D9, and D11, MMEnsemble clearly outperformed DDAE, however, on the other datasets, MMEnsemble was inferior to DDAE or comparable. This was caused by the low TNR and precision scores for MMEnsemble, coming from the weighting scheme design (i.e., the asset-based weighting). Asset-based weighting is designed to emphasize the base classifiers that correctly classify instances that others cannot. This increases the chance of increasing the number of false positives.

Table 2. Comparison with State-of-the-art Methods of Metric Learning (IML), Oversampling (ProWSyn) and Undersampling-based Ensemble (DDAE) – † means that scores were copied from the DDAE paper [36]. MMEnsemble achieved the best performance in the recall and AUC, and it was comparable with DDAE in the Gmean and F_2.

	IML†				ProWSyn				DDAE†				MMEnsemble			
data	*Rec*	*Gm*	F_2	*AUC*	*Rec*	*Gm*	F_2	*AUC*	*Rec*	*Gm*	F_2	*AUC*	*Rec*	*Gm*	F_2	*AUC*
D1	.313	.520	.287	.589	.411	.583	.350	.728	.813	**.775**	**.580**	.776	**.863**	.756	.546	**.819**
D2	.692	.805	**.652**	.814	.375	.571	.353	.735	.846	**.823**	.625	.823	**.952**	.750	.534	**.868**
D3	.500	.635	.345	.653	.406	.595	.364	.753	.750	**.815**	**.588**	.817	**.793**	.772	.528	**.866**
D4	.852	.657	.408	.679	.433	.620	.378	.821	**.963**	**.819**	**.573**	.830	.944	**.819**	.548	**.895**
D5	.510	.578	.342	.582	.444	.623	.404	.796	.735	.743	.536	.744	**.867**	**.794**	**.598**	**.854**
D6	.814	.725	.574	.730	.685	.790	.646	.920	.932	.804	.676	.813	**.963**	**.873**	**.748**	**.934**
D7	.667	.716	.471	.718	.502	.648	.387	.757	.833	**.841**	**.649**	.841	**.933**	.808	.512	**.883**
D8	.600	.709	.375	.719	.537	.695	.452	.803	.700	.814	.603	.824	**.886**	**.877**	**.650**	**.941**
D9	.700	.798	.407	.805	.669	.796	.532	.884	.900	.883	.421	.883	**.931**	**.920**	**.585**	**.976**
D10	.667	.626	.037	.628	.436	.612	.118	.754	**1.000**	**.839**	.075	.852	.935	.835	**.128**	**.876**
D11	.000	.000	NA	.500	.153	.248	**.287**	.691	.333	.550	.156	.620	**.894**	**.842**	.188	**.939**
D12	.800	.802	.252	.802	.680	.801	.471	.891	**1.000**	**.964**	**.556**	.965	.992	.943	.451	**.982**

Table 3. Ablation Study – EE, ML, and MR stand for EasyEnsemble [20], metric learning, and multi-ratio ensemble, respectively. MLEnsemble is extension of EE with ML, MUEnsemble [17] is extension of EE with MR, and MMEnsemble is extension of EE with ML and MR. For this evaluation, MMEnsemble used Gaussian-based weighting (Equation 1) with the same parameters as MUEnsemble for fair comparison with MUEnsemble. MMEnsemble achieved best performance on all metrics.

	EasyEnsemble (EE)				MLEnsemble (EE + ML)				MUEnsemble (EE + MR)				MMEnsemble (EE + ML + MR)			
data	*Rec*	*Gm*	F_2	*AUC*	*Rec*	*Gm*	F_2	*AUC*	*Rec*	*Gm*	F_2	*AUC*	*Rec*	*Gm*	F_2	*AUC*
D1	.705	.667	.441	.738	.751	.695	.475	.754	.812	.698	**.484**	**.783**	**.820**	**.699**	.483	**.783**
D2	.775	.706	.475	.805	.854	**.742**	**.518**	.831	.821	.718	.490	.826	**.891**	.731	.509	**.862**
D3	.730	.705	.444	.805	.790	.720	.461	.817	.761	.700	.439	.820	**.864**	**.761**	**.506**	**.860**
D4	.833	.778	.500	.848	.875	.804	.533	.871	**.880**	.788	.509	.860	.873	**.816**	**.548**	**.885**
D5	.799	.749	.540	.816	.821	.760	.554	.821	.828	.753	.546	.828	**.844**	**.781**	**.581**	**.837**
D6	.920	.873	.750	.925	.921	.844	.707	.907	.946	**.883**	**.764**	**.934**	**.971**	.873	.747	.921
D7	.771	.732	.427	.812	.787	.746	**.444**	.830	.792	.743	.438	.818	**.860**	**.749**	**.444**	**.859**
D8	.752	.751	.436	.830	.835	.822	**.537**	.913	.769	.757	.440	.840	**.911**	**.835**	.531	**.959**
D9	.820	.844	.418	.928	**.893**	.874	.438	.951	.850	.857	.427	.935	.885	**.890**	**.508**	**.973**
D10	.804	.735	.090	.807	.835	.762	.101	.828	.911	.770	.096	.834	**.999**	**.828**	**.112**	**.887**
D11	.587	.633	.152	.752	.735	.697	.144	.797	**.785**	**.753**	**.178**	**.841**	.765	.724	.160	.795
D12	.835	.823	.235	.911	.882	.875	.330	.951	.870	.840	.248	.931	**.987**	**.923**	**.363**	**.985**

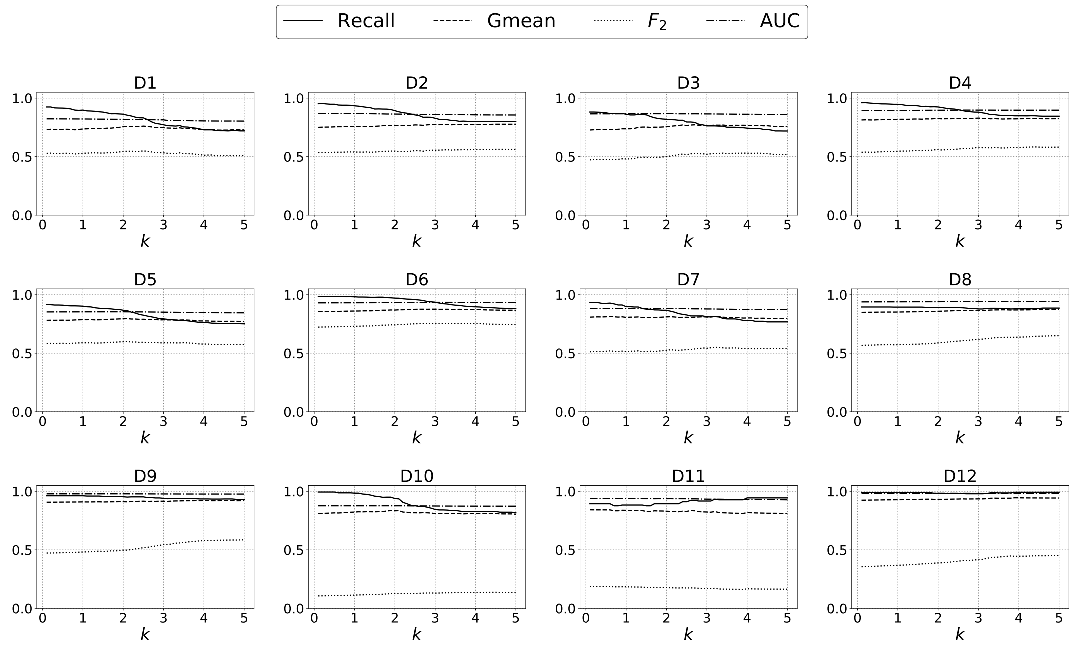

Fig. 3. Effect of k in Asset-based Weighting – As k increases, recall scores decrease, and the Gmean, F_2, and AUC scores increase.

4.2.2. *Impact of the Combination*

Table 3 shows a comparison of MMEnsemble and its basic approaches. The comparisons of EasyEnsemble to MLEnsemble and MUEnsemble show that the classification performance could be slightly increased by adding either the metric learning or the multi-ratio ensemble. The architectural difference between MUEnsemble and MMEnsemble is whether metric learning is involved; therefore, to observe the performance improvement caused by the difference, MMEnsemble was incorporated with the Gaussian weighting (Equation1). On the basis of this comparison, MMEnsemble showed its superiority to MUEnsemble, that is, the metric learning successfully improved the data space in the sets of data for each sampling ratio. In addition, as it can be seen by comparing the columns of MMEnsemble in Table 2 and Table 3, MMEnsemble with the asset-based weighting was superior to that with the Gaussian weighting; therefore, MMEnsemble clearly outperformed MUEnsemble.

4.2.3. *Effect of the Asset-based Weighting*

Figure 3 shows the effect of the hyper-parameter k on the asset-based weighting. A basic finding is that the recall scores dropped as k increased. This is because the higher the k, the more weights are given to the base classifiers that can correctly classify instances that are incorrectly classified by the other base classifiers. This leads to a higher TNR and precision; therefore, as k increases, the Gmean and F_2 scores increase, and similarly, AUC scores gradually increase.

Table 4 shows a comparison of the asset-based weighting with uniform weighting. Uniform

Table 4. Comparison between Uniform and Asset-based Weighting

	Uniform				Gauss				Asset			
data	*Rec*	*Gm*	F_2	AUC	*Rec*	*Gm*	F_2	AUC	*Rec*	*Gm*	F_2	AUC
D1	**.893**	.637	.456	.781	.820	.699	.483	.783	.863	**.756**	**.546**	**.819**
D2	.950	.711	.502	.818	.891	.731	.509	.862	**.952**	**.750**	**.534**	**.868**
D3	.813	.692	.435	.815	**.864**	.761	.506	.860	.793	**.772**	**.528**	**.866**
D4	**.954**	.788	.505	.891	.873	.816	**.548**	.885	.944	**.819**	**.548**	**.895**
D5	**.923**	.748	.550	.840	.844	.781	.581	.837	.867	**.794**	**.598**	**.854**
D6	**.972**	.846	.710	.925	.971	**.873**	.747	.921	.963	**.873**	**.748**	**.934**
D7	.915	.742	.432	.882	.860	.749	.444	.859	**.933**	**.808**	**.512**	**.883**
D8	.900	.817	.509	.931	**.911**	.835	.531	**.959**	.886	**.877**	**.650**	.941
D9	.910	.872	.413	.954	.885	.890	.508	.973	**.931**	**.920**	**.585**	**.976**
D10	.924	.758	.091	.837	**.999**	.828	.112	**.887**	.935	**.835**	**.128**	.876
D11	.633	.666	.152	.810	.765	.724	.160	.795	**.894**	**.842**	**.188**	**.939**
D12	.873	.858	.303	.953	.987	.923	.363	**.985**	**.992**	**.943**	**.451**	.982

weighting gave equal weights for all base classifiers. MMEnsemble with uniform weighting tended to achieve high recall scores, but low scores for the other metrics. This indicates that taking the average performance among the base classifiers trained using datasets of different sampling ratios increases the number of instances classified to the minority class.

Although the details are omitted due to space limitations, it is noteworthy that MLEnsemble with the asset-based weighting showed a similar classification performance to that with the uniform weighting. This is because the base classifiers in MLEnsemble are close in terms of classification tendency, that is, correctly classified instances are almost common among these classifiers. Thus, the weights on these classifiers calculated by Equation 2 become similar values. This fact indicates that the asset-based weighting is effective for ensemble classifiers of which the classification tendencies of the base classifiers differ from each other. MMEnsemble is this kind of ensemble classifier, that is, base classifiers are trained for different sampling ratios; thus, the tendencies of these base classifiers differ from each other.

4.3. *Lessons Learned*

Q1: Does MMEnsemble outperform the state-of-the-art imbalanced classification methods of metric learning, oversampling and undersampling? — In terms of recall and AUC, MMEnsemble achieved the state-of-the-art, while MMEnsemble was comparable with DDAE in Gmean and F_2. This indicates that MMEnsemble can achieve a higher recall, but its performance in TNR and precision is limited. Though many real-world applications expect a higher recall, a high TNR and precision with high recall is ideal; therefore, improving MMEnsemble for these metrics without sacrificing high recall is a promising next direction.

Q2: Is the combination of metric learning and multi-ratio ensemble effective? — Yes, the combination contributes to improving the classification performance for all metrics. The comparison between MMEnsemble and MLEnsemble revealed that the multi-ratio ensemble improved the performance, and that between MMEnsemble and MUEnsemble revealed that the metric learning improved the performance.

Q3: Does the asset-based weighting help improve the classification performance? and what is the effect of choice of its hyper-parameter k (Equation 2)? —

Asset-based weighting improved classification performance compared with the two weighting schemes (uniform and Gaussian) for all metrics; however, the hyper-parameter k must be carefully determined because it is sensitive to recall. As k increases, recall decreases, while the Gmean and F_2 increase. This indicates that a higher k improves classifiers in terms of the TNR and precision. Thus, k can be tuned in terms of users' preferences on recall or precision.

5. Conclusion

In this paper, a novel undersampling-based ensemble framework, MMEnsemble was proposed. MMEnsemble integrates three techniques, metric learning, multi-ratio ensemble, and asset-based weighting to overcome the insufficient data space issue in the previous undersampling-based ensemble approaches. An experimental evaluation revealed the superiority of MMEnsemble to the state-of-the-art methods, especially for the recall and AUC metrics. The major limitation of MMEnsemble (also in the other methods) is that it can achieve higher recall scores but sacrifices precision.

Acknowledgements
This work was partly supported by JSPS KAKENHI JP21H0355 and the Kayamori Foundation of Informational Science Advancement.

References

1. J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, and S. García. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. *J. Multiple Valued Log. Soft Comput.*, 17(2-3):255–287, 2011.
2. S. Barua, M. M. Islam, and K. Murase. ProWSyn: Proximity Weighted Synthetic Oversampling Technique for Imbalanced Data Set Learning. In *PAKDD 2013*, pages 317–328, 2013.
3. S. Barua, M. M. Islam, X. Yao, and K. Murase. MWMOTE-Majority Weighted Minority Oversampling Technique for Imbalanced Data Set Learning. *IEEE Trans. Knowl. Data Eng.*, 26(2):405–425, 2014.
4. G. E. A. P. A. Batista, A. L. C. Bazzan, and M. C. Monard. Balancing Training Data for Automated Annotation of Keywords: a Case Study. In *II Brazilian Workshop on Bioinformatics*, pages 10–18, 2003.
5. G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations*, 6(1):20–29, 2004.
6. C. Bellinger, S. Sharma, N. Japkowicz, and O. R. Zaane. Framework for extreme imbalance classification: SWIM-sampling with the majority class. *KAIS*, 2019.
7. S. Bhattacharya, V. Rajan, and H. Shrivastava. ICU Mortality Prediction: A Classification Algorithm for Imbalanced Datasets. In *AAAI 2017*, pages 1288–1294, 2017.
8. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Intell. Res.*, 16:321–357, 2002.
9. C. Elkan. The Foundations of Cost-Sensitive Learning. In *IJCAI 2001*, pages 973–978, 2001.
10. M. Galar, A. Fernández, E. B. Tartas, H. B. Sola, and F. Herrera. A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. *IEEE Trans. Systems, Man, and Cybernetics, Part C*, 42(4):463–484, 2012.
11. S. Gazzah and N. E. B. Amara. New Oversampling Approaches Based on Polynomial Fitting for Imbalanced Data Sets. In *DAS 2008*, pages 677–684, 2008.
12. S. E. Gómez, L. Hernández-Callejo, B. C. Martínez, and A. J. Sánchez-Esguevillas. Exploratory study on Class Imbalance and solutions for Network Traffic Classification. *Neurocomputing*, 343:100–119, 2019.

13. H. Han, W. Wang, and B. Mao. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In *ICIC 2005*, pages 878–887, 2005.
14. H. He, Y. Bai, E. A. Garcia, and S. Li. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *IJCNN 2008*, pages 1322–1328, 2008.
15. H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Trans. Knowl. Data Eng.*, 21(9):1263–1284, 2009.
16. P. Kang and S. Cho. EUS SVMs: Ensemble of Under-Sampled SVMs for Data Imbalance Problems. In *ICONIP 2006*, pages 837–846, 2006.
17. T. Komamizu, R. Uehara, Y. Ogawa, and K. Toyama. MUEnsemble: Multi-ratio Undersampling-Based Ensemble Framework for Imbalanced Data. In *DEXA 2020*, pages 213–228, 2020.
18. G. Kovács. An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets. *Applied Soft Computing*, 83:105662, 2019. (IF-2019=4.873).
19. X. Liu, J. Wu, and Z. Zhou. Exploratory Undersampling for Class-Imbalance Learning. *IEEE Trans. Systems, Man, and Cybernetics, Part B*, 39(2):539–550, 2009.
20. X. Liu, J. Wu, and Z. Zhou. Exploratory Undersampling for Class-Imbalance Learning. *IEEE Trans. Systems, Man, and Cybernetics, Part B*, 39(2):539–550, 2009.
21. W. Lu, Z. Li, and J. Chu. Adaptive Ensemble Undersampling-Boost: A novel learning framework for imbalanced data. *Journal of Systems and Software*, 132:272–282, 2017.
22. I. Mani and I. Zhang. kNN approach to unbalanced data distributions: a case study involving information extraction. In *ICML'2003 Workshop on Learning from Imbalanced Datasets*, volume 126, 2003.
23. H. M. Nguyen, E. W. Cooper, and K. Kamei. Borderline over-sampling for imbalanced data classification. *IJKESDP*, 3(1):4–21, 2011.
24. M. Peng, Q. Zhang, X. Xing, T. Gui, X. Huang, Y. Jiang, K. Ding, and Z. Chen. Trainable Undersampling for Class-Imbalance Learning. In *AAAI 2019*, pages 4707–4714, 2019.
25. A. D. Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi. Calibrating Probability with Undersampling for Unbalanced Classification. In *SSCI 2015*, pages 159–166, 2015.
26. D. Rodríguez, I. Herraiz, R. Harrison, J. J. Dolado, and J. C. Riquelme. Preliminary Comparison of Techniques for Dealing with Imbalance in Software Defect Prediction. In *EASE 2014*, pages 43:1–43:10, 2014.
27. C. Seiffert, T. M. Khoshgoftaar, J. V. Hulse, and A. Napolitano. RUSBoost: A Hybrid Approach to Alleviating Class Imbalance. *IEEE Trans. Systems, Man, and Cybernetics, Part A*, 40(1):185–197, 2010.
28. A. Sharififar, F. Sarmadian, and B. Minasny. Mapping imbalanced soil classes using Markov chain random fields models treated with data resampling technique. *Computers and Electronics in Agriculture*, 159:110–118, 2019.
29. M. R. Smith, T. R. Martinez, and C. G. Giraud-Carrier. An instance level analysis of data complexity. *Machine Learning*, 95(2):225–256, 2014.
30. I. Tomek. Two Modifications of CNN. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(11):769–772, 1976.
31. J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo. OpenML: networked science in machine learning. *SIGKDD Explor.*, 15(2):49–60, 2013.
32. H. Wang, Y. Gao, Y. Shi, and H. Wang. A Fast Distributed Classification Algorithm for Large-Scale Imbalanced Data. In *ICDM 2016*, pages 1251–1256, 2016.
33. N. Wang, X. Zhao, Y. Jiang, and Y. Gao. Iterative Metric Learning for Imbalance Data Classification. In *IJCAI 2018*, pages 2805–2811, 2018.
34. K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance Metric Learning for Large Margin Nearest Neighbor Classification. In *NIPS 2005*, pages 1473–1480, 2005.
35. D. L. Wilson. Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. *IEEE Trans. Systems, Man, and Cybernetics*, 2(3):408–421, 1972.
36. J. Yin, C. Gan, K. Zhao, X. Lin, Z. Quan, and Z. Wang. A Novel Model for Imbalanced Data Classification. In *AAAI 2020*, pages 6680–6687, 2020.