Journal of Data Intelligence and Predictive Systems (JDIPS) **2025, 1(1): 1-10**
https://www.longmanpublishers.com/

An encapsulated framework for imbalanced classification based on multi-ratio undersampling

Riku Tsude
Hirosaki Gakuin University, Hirosaki, Aomori, Japan

Abstract: Real-world data frequently exhibits class imbalance, which is troublesome since it impairs classification ability because of biased supervision. One efficient resampling technique for the class imbalance is undersampling. One fixed sample ratio is used in the traditional undersampling-based methods. Nonetheless, preferences for classes vary depending on the sampling ratio. This study proposes MUEnsemble, an ensemble architecture based on undersampling. A versatile design for weighting weak classifiers in various sampling ratios is made possible by this framework, which incorporates weak classifiers of various sampling ratios. This paper presents a uniform weighting function and a Gaussian weighting function to illustrate the design principle. According to a thorough experimental review, MUEnsemble performs better than state-of-the-art techniques that rely on undersampling or oversampling in terms of recall, gmean, F-measure, and ROC-AUC metrics. Furthermore, the evaluation demonstrates the superiority of the Gaussian weighting function over the uniform weighting function. This suggests that the various preferences of sampling ratios toward classes can be captured by the Gaussian weighting function. An analysis of the impacts of the Gaussian weighting function's parameters reveals that recall can be used to determine the function's parameters, which is desirable in many real-world applications.
Keywords: Imbalanced classification, Resampling, Undersampling, Ensemble.

1. Introduction

In real-world applications, class imbalance is a significant issue that impairs classification performance, particularly when minority classes are involved. When there are significantly more examples in one class than in other classes, this is referred to as class imbalance in datasets. Classifiers are biased in favor of the majority class due to the significant disparity in the number of examples. Class imbalance has been noted and addressed in a number of fields, including computer networks [13], software engineering [29], and the fields of medicine [7], economics [28], and agriculture [32].

Resampling has been extensively researched as a successful remedy for class imbalance [8, 22, 5, 33]. Resampling methods can be broadly divided into two groups: undersampling (e.g., EasyEnsemble [22] and RUSBoost [31]) and oversampling (e.g., SMOTE [8] and SWIM [33]).

A straightforward and effective resampling method for addressing class imbalance is undersampling [11]. The problem has been solved by integrating multiple undersampled datasets in an ensemble fashion in addition to employing single-shot undersampling [22, 17, 31].

According to Figure 1, a preliminary analysis of multiple datasets on the effects of various undersampling ratios revealed that preferences for classes vary depending on the undersampling ratio. The ratio of the sampled majority size to the minority size is known as the sampling ratio. The majority class is considered the negative class in this study, whereas the minority class is considered the positive class. The majority examples are chosen at random when the sampling ratio is 1.0, ensuring that the number of sampled examples is equal to that of the minority examples. When the ratio is less than 1.0, it indicates that there are fewer sampled majority cases than minority examples. This is referred to as severe undersampling in this work, while moderate undersampling is its opposite. The Abalone dataset's true positive and negative ratios for various sample ratios are shown in the figure. It suggests that 1.0 might not be the optimal balanced ratio because classifiers trained with severely undersampled datasets favor the minority class, while those trained with moderately undersampled datasets favor the majority class.

The goal of this paper is to use the ensemble framework MUEnsemble to combine classifiers in various sampling ratios. First, a variety of sampling ratios undersample the bulk of the cases. For every sampling ratio, several undersampled datasets are created in order to use a significant number of majority examples. The various datasets are then used to develop a base ensemble classifier. An ensemble of the base classifiers then makes up the overall classifier. MUEnsemble introduces weighting functions based on the voting technique in ensemble learning to express the preferences of sample ratios toward classes. The weighting functions in this paper are a Gaussian function and a uniform weighting function. The latter uses a Gaussian function to weight classifiers in

a sampling ratio, whereas the former is for analogous voting. When it comes to weighing on a central sample ratio and the other sampling ratios, the Gaussian function is adaptable.

Precision, recall, gmean, f-measure, ROC-AUC, and other common assessment metrics for the class imbalance problem were used to assess MUEnsemble on 31 publically accessible real-world datasets. Numerous dimensionalities, record counts, and imbalance ratios are present in the datasets (the dataset statistics are displayed in Section 4). In terms of recall-preferred metrics (i.e., recall, gmean, and f-measure), MUEnsemble fared better than the state-of-the-art techniques.

The following is a summary of this paper's contributions.
• **Multi-ratio Undersampling-based Ensemble Framework:** This research proposes MUEnsemble, an ensemble framework for unbalanced data. Several undersampling ratios, including extreme undersampling, are used by MUEnsemble. For every sample ratio, an ensemble classifier is learned, and the ensemble classifiers are then combined across sampling ratios. It has two weighting functions for ensemble classifiers that correlate to sampling ratios: Gaussian and uniform.
• **Comprehensive Testing on a Range of Public Datasets:** MUEnsemble is tested on a range of publicly accessible datasets to compare it to the most advanced resampling techniques. The datasets cover a wide range of topics and feature different imbalance ratios. Precision, recall, gmean, f-measure, and ROC-AUC are the standard evaluation metrics used for the class imbalance problem. In terms of recall-preferred metrics, the experiment shows that MUEnsemble performs better than the state-of-the-art techniques. The potential parameter combinations are examined in order to demonstrate the Gaussian weighting function's flexibility. This is how the remainder of the paper is structured. Related resampling-based techniques for unbalanced data are introduced in Section 2. The four balancing functions, MUEnsemble, and the suggested framework are explained in Section 3. In addition to discussing the impacts of excessive undersampling, balancing functions, and parameter optimization, Section 4 shows how successful MUEnsemble is in comparison to state-of-the-art techniques. This paper is finally concluded in Section 5.

2. Related Work: Resampling Approaches

There are essentially three categories of methods for addressing class imbalance: sampling, cost-adjustment [10, 20], and algorithm change [2, 37]. Since it can be applied to any classifier and has been demonstrated to have strong performance, resampling is the most widely used method. Oversampling and undersampling are the two broad categories into which resampling techniques fall.

2.1. Methods Based on Oversampling

Copying minority cases at random to equalize the numbers of minority and majority examples is a straightforward oversampling technique. This method is prone to overfitting. Oversampling techniques produce artificial minority instances that are near the minority in order to address the overfitting issue. The most often used synthetic oversampling technique is SMOTE [8]. It uses the nearest neighbor technique to create artificial minority examples. The generated examples may readily overlap with majority instances because SMOTE does not account for majority cases. The categorization performance suffers as a result. More recent methods have included majority instances to the resampling procedure in order to get over SMOTE's drawback. Data purification methods used by SMOTE-Tomek [4] and SMOTE-ENN [5] include Edit Nearest Neighbours [38] and Tomek link removal [35]. Similarly, there are more sophisticated methods (e.g., SVM-SMOTE [26], borderline-SMOTE [14], and ADASYN [15]). SWIM is one of the most advanced synthetic over-sampling techniques [33, 6]. To create synthetic minority instances, SWIM makes use of each minority example's density in relation to the distribution of majority examples. Extensive studies were carried out by [18] to examine a wide range of SMOTE variations and compare them with various dataset types. PolyFitSMOTE [12] and ProWSyn [3] performed the best in this experiment.

2.2. Techniques Based on Undersampling

Three types of undersampling-based methods can be distinguished: boosting and bagging ensembles, as well as example selection. The process of selecting majority examples that are anticipated to improve categorization is known as example selection. Major techniques select examples of the majority that are difficult to differentiate from instances of the minority. Majority examples that are near minority instances are sampled by NearMiss [25]. A hardness property called instance hardness [34] shows how likely it is that an example would be incorrectly labeled.
A learning technique called "boosting ensemble" progressively modifies the bulk of samples. Boosting approaches eliminate a portion of the majority samples for every iteration. A boosting technique called BalanceCascade [22] eliminates majority samples that are correctly classified. A weighted random

undersampling technique called RUS-Boost [31] is used to exclude majority examples that are probably going to be correctly identified. To enhance model performance, EUSBoost [23] presents an adaptive boundary choice technique and a cost-sensitive weight change. The best in this category is Trainable Undersampling [27]. It uses reinforcement learning to train a classifier.

Multiple weak classifiers are combined in a bagging ensemble, each of which is trained in a voting fashion on separate pieces of undersampled training data. One of the earliest bagging techniques that used undersampled training data was Ensemble of Undersampling [17]. An ensemble-of-ensemble method called EasyEnsemble [22] groups AdaBoost classifiers for every piece of undersampled training data in a bagging fashion. A thorough experiment on bagging and boosting strategies is documented in [11]. It demonstrates that the top-performing methods are RUSBoost and EasyEnsemble, which outperform methods based on oversampling.

MUEnsemble falls under the category of bagging ensembles. The following are the ways that MUEnsem-ble differs from current undersampling techniques. The first ensemble method that combines several sampling ratios is MUEnsemble. Second, among these methods, it incorporates significant under-sampling, which has not been taken into account. Third, it uses a Gaussian function to determine the weights assigned to weak classifiers.

3. MUEnsemble

Complex Multiple sample ratios are used in the ensemble framework MUEnsemble. An overview of MUEnsemble is shown in Figure 2. The sampling phase and the ensemble phase are its two stages. A collection of sampling ratios with size n is initially determined in the sample phase, given that the user specified the number n of sampling ratios. Second, a batch of undersampled datasets is created by repeatedly undersampling the input data for each sampling ratio. Each batch of undersampled datasets is utilized to train a base ensemble classifier during the ensemble phase. Lastly, an ensemble classifier is created by combining the base ensemble classifiers for every sampling ratio. The set of sample ratios and the weights assigned to the base classifiers are the two main parameters of MUEnsemble. In contrast to a set of sample ratios, the weights are regarded in this work as the primary factors in classification performance.

Therefore, MUEnsemble uses a straightforward splitting technique to determine a set of sample ratios. In this research, two functions are proposed for the weights of the basic classifiers: a Gaussian weighting function and a uniform weighting function.

3.1. Calculating the Sampling Ratio

It is preferable to automatically determine the sampling ratios to be ensembled since human users find this to be non-intuitive. The user provides an interval value to split the imbalance ratio (IR) into a sequence of sampling ratios at intervals of the value, which is one method of simplifying the parameter to a ratio interval. Assume, for example, that the ratio interval is 0.2 and the IR is 2.0. The sample ratios are therefore {0.2, 0.4, 0.6,..., 2.0}. Nevertheless, the IR varies from dataset to dataset, and an excessively high IR results in an excessively high number of sample ratios. Because so many base classifiers are learned, the ensemble grows hefty and the training cost rises excessively.

MUEnsemble uses a splitting approach to calculate the sampling ratio in order to get around this problem. To find the number |N ′| of sampled majority examples, two numbers are required: nP and nN. The number of blocks that divide the minority (or majority) example size equally is nP (resp. nN). An example of the dividing approach is shown in Figure 3. The majority example size is separated into blocks of size nP + nN + 1 (in the example, nP = nN = 4).

The blue-shaded regions in Figure 3 represent the sample ratios of the associated iteration c. The training uses the proportions of orange-shaded and blue-shaded areas for the majority and minority cases.

3.2. Function of Weighing

For determining the weights on base classifiers of sampling ratios, the weighting function is a programmable function. This study introduces the Gaussian weighting function and the previously described uniform weighting function in order to examine the basis choices for the weight function.

4. Experimental Evaluation

According Three questions were assessed for MUEnsemble in this paper:
Q1: Does MUEnsemble perform better than standard techniques? — Using datasets with a broad range of domains, dimensionalities, record counts, and imbalance ratios, MUEnsem-ble was compared with cutting-edge resampling-based techniques in order to address this topic.
Q2: How successful are ensemble classifications when multi-ratio undersampling is taken into account? — EasyEnsemble [22], a single sampling ratio variant of MUEnsemble, was compared with MUEnsemble in order to address this query.
Question 3: Could the Gaussian weighting function be superior to the uniform weighting function? This question was addressed by comparing MUEnsemble with the uniform weighting function and that with the Gaussian weighting function. The study demonstrated the relationship between the parameters of the Gaussian weighting and uniform weighting functions by comparing both the optimal and potential parameters.

4.1. Configuration

4.1.1. Assessment

Gmean, F2, AUC, Precision, Recall, and Recall were the evaluation metrics. The true positives, false negatives, true negatives, and false positives are denoted by TP, FN, TN, and FP.

Table 1. Classification Datasets

ID	Name	#dim	#major	#minor	IR
D1	Abalone (9 v. 18)	8	689	42	16.4
D2	Anuran Calls (Lept. v. Bufo.)	22	4,420	68	65.0
D3	Covertype (2 v. 5)	54	283,301	9,493	29.8
D4	default of credit card clients	23	23,364	6,636	3.5
D5	HTRU2	8	16,259	1,639	9.9
D6	Online Shoppers Purchasing Intention	17	10,422	1,908	5.5
D7	Polish companies bankruptcy	64	5,500	410	13.4
D8	Spambase	57	2,788	1,813	1.5
D9	Wine Quality – Red ((3, 4) v. others)	11	1,536	63	24.4
D10	Wine Quality – White (7 v. 3)	11	880	20	44.0
D11	Churn Modelling	9	7,963	2,037	3.9
D12	Credit Card Fraud Detection	29	284,315	492	577.9
D13	ECG Heartbeat – Arrhythmia (N v. F)	187	90,589	803	112.8
D14	Financial Distress	85	3,536	136	26.0
D15	LoanDefault LTFS AV	39	182,543	50,611	3.6
D16	Mafalda Opel– Driving Style	14	9,530	2,190	4.4
D17	Mafalda Peugeot – Driving Style	14	12,559	678	18.5
D18	Rain in Australia	20	110,316	31,877	3.5
D19	Surgical	24	10,945	3,690	3.0
D20	Paysim1	10	6,354,407	8,213	773.7
D21	cm1	21	449	49	9.2
D22	kc3	39	415	43	9.7
D23	mw1	37	372	31	12.0
D24	pc1	21	1,032	77	13.4
D25	pc3	37	1,403	160	8.8
D26	pc4	37	1,280	178	7.2
D27	Yeast (1 v. 7)	7	429	30	14.3
D28	Yeast (6 v. others)	8	1,449	35	41.4
D29	Abalone (19 v. others)	8	4,142	32	129.4
D30	Wine Quality – Red (3 v. 5)	8	1,890	26	72.7
D31	Abalone (20 v. (8, 9, 10))	11	681	10	68.1

The experimental procedure was repeated 50 times in order to precisely estimate these evaluation metric values. The classifiers were trained on the training set and assessed using the test set after a dataset was randomly divided into 70% for training and 30% for testing. The macro average of the 50 trials served as the overall metric scores.

4.1.2. Information Sources

Publicly accessible datasets with a broad variety of dataset attributes were used for this assessment. The datasets are summarized in Table 1. The UCI Machine Learning Repository served as the source for D1–D10 [9], the

Kaggle Dataset provided D11–D20, the OpenML dataset provided D21–D26 [36], and the KEEL repository provided D27–D31 [1]. Few category attributes are present in these datasets, and those that are are dictionary-encoded to numeric attributes.b Two of the classes in the datasets were chosen to be used for the binary classification job, which is shown in the dataset column in brackets. Other datasets were for the multi-class classification work. The datasets varied in terms of the imbalance ratio (IR), the number of majority and minority cases (#major, #minor), and dimensionality (#dim.).

4.1.3. Starting points

The following simple baselines, synthetic oversampling techniques, and undersampling-based techniques were used to compare MUEnsemble.

- SMT: AdaBoost Classifier [30] plus SMOTE [8]
- PWS: AdaBoost Classifier + ProWSyn [3]
- PFS: AdaBoost Classifier + SMOTE with Polynomial Fitting [12]
- RUS: AdaBoost Classifier plus random undersampling with a sampling ratio of one RUSBoost [31] is the RBS.
- EasyEnsemble (EE) [22]

Synthetic oversampling techniques include SMT, PWS, and PFS; SMT is the most widely used technique, but PRW and PFS have demonstrated better results in [18]. A baseline method for undersampling-based classifiers is RUS. With a sampling ratio of 1.0, it is a straightforward undersampling. RUSBoost is a boosting ensemble approach, while EasyEnsemble is an undersampling-based ensemble method that is closer to MUEnsemble except for the inclusion of different sampling ratios. The classifiers of the simulated oversampling techniques and RUS were assigned to the AdaBoost classifier [30], the foundational classifier of EasyEnsemble and MUEnsemble, in order to compare the impacts of re-sampling. SMT, PWS, and PFS implementations were found in [19], whereas RUS, EasyEnsemble, and RUSBoost implementations were found in the imbalanced-learn package [21]. Scikit-learn (version 0.20.3) was used to learn classifiers.C

4.1.4. Characteristics

The following were the MUEnsemble parameters. nP = nN = 10. The Gaussian function's μ and σ2 parameter spaces were defined as follows: $\mu \in \{0.2, 0.6, 1.0, 1.4, 1.8, 2.0\}$, and $\sigma 2 \in \{0.2, 0.4, 0.6,..., 3.0\}$. The baselines' parameters were set to their default values.

4.2. Findings

Table 2. *Precision* Comparison

	Oversampling			Undersampling			MUEnsemble	
data	SMT	PWS	PFS	RUS	RBS	EE	Uni	Gau
D1	0.279	0.275	**0.402**	0.136	0.200	0.158	0.118	0.149
D2	0.998	0.998	0.998	0.999	0.999	0.999	0.999	**1.000**
D3	0.142	0.246	**0.572**	0.135	0.139	0.140	0.137	0.140
D4	0.626	0.631	**0.668**	0.455	0.451	0.468	0.440	0.648
D5	0.749	0.730	**0.900**	0.690	0.491	0.735	0.698	0.728
D6	0.611	0.621	**0.666**	0.533	0.472	0.536	0.519	0.545
D7	0.619	0.633	**0.849**	0.451	0.482	0.498	0.421	0.481
D8	0.918	0.918	**0.927**	0.903	0.903	0.914	0.888	0.910
D9	0.977	0.974	0.966	0.977	0.969	0.986	0.986	**0.990**
D10	0.984	0.983	0.985	0.992	0.982	0.991	0.994	**1.000**
D11	0.635	0.616	**0.694**	0.457	0.456	0.457	0.429	0.625
D12	0.083	0.109	**0.340**	0.030	0.040	0.044	0.034	0.045
D13	0.120	0.141	**0.841**	0.078	0.097	0.093	0.080	0.093
D14	0.271	0.277	**0.412**	0.150	0.165	0.171	0.145	0.171
D15	0.295	0.285	**0.464**	0.295	0.296	0.297	0.286	0.432
D16	0.923	0.893	0.859	0.935	0.929	0.944	0.955	**0.991**
D17	0.979	0.972	0.963	0.988	0.982	0.991	0.994	**0.999**
D18	0.564	0.604	**0.694**	0.516	0.518	0.519	0.504	0.663
D19	0.734	0.756	**0.854**	0.634	0.621	0.646	0.610	0.805
D20	0.047	0.050	**0.954**	0.036	0.038	0.037	0.036	0.037
D21	0.211	**0.245**	0.227	0.164	0.181	0.177	0.153	0.184
D22	0.338	0.343	**0.439**	0.191	0.199	0.189	0.168	0.193
D23	0.265	0.272	**0.290**	0.156	0.165	0.163	0.147	0.158

D24	**0.319**	0.262	0.305	0.177	0.163	0.194	0.155	0.200
D25	0.310	0.285	**0.350**	0.228	0.217	0.241	0.205	0.232
D26	0.529	0.527	**0.616**	0.410	0.435	0.424	0.379	0.440
D27	0.241	0.178	**0.262**	0.123	0.163	0.155	0.125	0.176
D28	0.336	0.285	**0.493**	0.115	0.213	0.142	0.122	0.141
D29	0.028	**0.029**	0.005	0.017	0.022	0.020	0.016	0.021
D30	0.237	0.223	**0.400**	0.053	0.128	0.062	0.049	0.061
D31	**0.064**	0.031	0.025	0.031	0.042	0.039	0.036	0.034

4.2.1. Q1: Does MUEnsemble perform better than standard techniques?— Indeed, MUEnsemble performed the best in terms of AUC, F2, gmean, and recall.

Performance comparisons in terms of precision, recall, gmean, F2, and AUC are displayed in Tables 2, 3, 4, 5, and 6, respectively. MUEnsemble performed better than the other approaches in terms of recall, gmean, F2, and AUC metrics. The oversampling-based methods, particularly SMOTE with polynomial fitting (PFS), fared best for the precision metric. Conversely, MUEnsemble and other undersampling-based techniques outperformed the oversampling-based techniques in terms of recall scores. Higher recall is preferred in many real-world applications, including the detection of arrhythmias and credit card theft. According to this viewpoint, methods based on undersampling are superior to those based on oversampling.

Table 3. *Recall* Comparison

	Oversampling			Undersampling			MUEnsemble	
data	SMT	PWS	PFS	RUS	RBS	EE	Uni	Gau
D1	0.588	0.511	0.328	0.658	0.463	0.717	0.803	**0.931**
D2	0.886	0.863	0.865	0.947	0.914	0.961	0.968	**0.971**
D3	0.872	0.521	0.133	0.896	0.887	0.905	0.921	**0.992**
D4	0.411	0.406	0.357	0.616	0.621	0.613	0.649	**0.977**
D5	0.902	0.909	0.852	0.912	0.752	0.914	0.919	**0.960**
D6	0.677	0.655	0.568	0.812	0.733	0.818	0.836	**0.957**
D7	0.853	0.828	0.732	0.901	0.811	0.917	0.934	**0.976**
D8	0.926	0.926	0.915	0.932	0.923	0.939	0.956	**0.976**
D9	0.511	0.447	0.242	0.624	0.377	0.764	0.832	**0.968**
D10	0.283	0.283	0.350	0.720	0.273	0.683	0.783	**1.000**
D11	0.553	0.538	0.457	0.733	0.723	0.734	0.769	**0.947**
D12	0.897	0.878	0.827	0.917	0.875	0.915	0.922	**0.959**
D13	0.876	0.834	0.456	0.914	0.868	0.910	0.925	**0.971**
D14	0.593	0.456	0.259	0.826	0.540	0.878	0.921	**0.983**
D15	0.181	0.121	0.015	0.631	0.628	0.634	0.705	**0.993**
D16	0.728	0.593	0.360	0.782	0.756	0.816	0.862	**0.984**
D17	0.675	0.533	0.329	0.828	0.727	0.869	0.915	**0.994**
D18	0.664	0.606	0.490	0.762	0.760	0.763	0.784	**0.951**
D19	0.708	0.703	0.666	0.817	0.813	0.814	0.832	**0.951**
D20	0.944	0.934	0.483	0.983	0.981	0.984	0.985	**0.996**
D21	0.347	0.393	0.153	0.648	0.388	0.716	0.892	**0.960**
D22	0.423	0.423	0.331	0.697	0.358	0.772	0.858	**0.962**
D23	0.320	0.390	0.260	0.686	0.308	0.708	0.784	**0.970**
D24	0.550	0.446	0.242	0.738	0.352	0.839	0.934	**0.971**
D25	0.483	0.417	0.300	0.746	0.512	0.798	0.887	**0.952**
D26	0.704	0.730	0.509	0.880	0.695	0.925	0.949	**0.980**
D27	0.467	0.467	0.344	0.640	0.398	0.756	0.809	**0.967**
D28	0.555	0.609	0.482	0.838	0.651	0.847	0.820	**0.964**
D29	0.450	0.400	0.004	0.698	0.340	0.830	0.930	**0.960**
D30	0.713	0.637	0.325	0.792	0.552	0.853	0.833	**0.963**
D31	0.167	0.067	0.067	0.613	0.193	0.693	0.740	**0.900**

Table 4. *Gmean* Comparison

	Oversampling			Undersampling			MUEnsemble	
data	SMT	PWS	PFS	RUS	RBS	EE	Uni	Gau
D1	0.723	0.675	0.552	0.686	0.625	**0.734**	0.703	**0.734**
D2	0.940	0.927	0.928	0.947	0.952	0.967	**0.972**	0.966
D3	0.847	0.702	0.364	0.851	0.851	0.858	**0.862**	0.860
D4	0.618	0.615	0.582	0.698	0.688	0.701	**0.704**	**0.704**

D5	0.935	0.937	0.919	0.935	0.806	0.940	0.939	**0.941**
D6	0.789	0.779	0.733	0.840	0.774	0.844	0.847	**0.850**
D7	0.905	0.893	0.851	0.909	0.870	**0.924**	0.919	**0.924**
D8	0.936	0.936	0.934	0.933	0.929	0.940	0.938	**0.941**
D9	0.661	0.604	0.452	0.638	0.537	**0.711**	0.622	0.710
D10	0.478	0.464	0.537	0.713	0.440	0.732	0.760	**0.807**
D11	0.712	0.701	0.658	0.754	0.748	0.755	0.753	**0.758**
D12	0.939	0.931	0.908	0.932	0.917	**0.940**	0.938	0.939
D13	0.909	0.892	0.674	0.909	0.897	**0.915**	**0.915**	**0.915**
D14	0.745	0.658	0.502	0.821	0.686	0.856	0.852	**0.860**
D15	0.398	0.332	0.120	0.606	0.606	**0.609**	0.601	**0.609**
D16	0.736	0.680	0.569	0.747	0.742	0.762	**0.764**	0.762
D17	0.751	0.685	0.556	0.803	0.763	0.829	0.822	**0.836**
D18	0.752	0.732	0.678	0.777	0.777	0.780	0.780	**0.781**
D19	0.804	0.806	0.800	0.829	0.817	0.832	0.826	**0.834**
D20	0.960	0.955	0.695	0.974	0.974	**0.976**	0.975	**0.976**
D21	0.532	0.575	0.345	0.634	0.540	0.667	0.629	**0.695**
D22	0.610	0.611	0.554	0.687	0.533	0.708	0.686	**0.722**
D23	0.538	0.556	0.481	0.668	0.489	0.682	0.675	**0.684**
D24	0.703	0.631	0.470	0.731	0.537	0.781	0.749	**0.789**
D25	0.650	0.603	0.528	0.727	0.630	**0.753**	0.733	0.740
D26	0.800	0.813	0.696	0.849	0.777	0.872	0.860	**0.879**
D27	0.635	0.616	0.559	0.645	0.561	0.726	0.690	**0.759**
D28	0.723	0.756	0.681	0.829	0.768	**0.856**	0.831	0.844
D29	0.624	0.584	0.013	0.680	0.505	0.744	0.705	**0.758**
D30	0.828	0.782	0.558	0.785	0.709	0.830	0.797	**0.840**
D31	0.249	0.114	0.113	0.613	0.302	0.692	**0.696**	0.653

Table 5. F_2 Comparison

data	Oversampling			Undersampling			MUEnsemble	
	SMT	PWS	PFS	RUS	RBS	EE	Uni	Gau
D1	**0.478**	0.431	0.334	0.366	0.354	0.418	0.371	0.406
D2	0.905	0.886	0.887	0.956	0.929	0.968	0.974	**0.977**
D3	0.430	0.426	0.158	0.422	0.428	0.432	0.430	**0.433**
D4	0.441	0.437	0.394	0.575	0.574	0.577	0.592	**0.631**
D5	0.867	0.866	0.862	0.857	0.652	**0.871**	0.864	**0.871**
D6	0.662	0.648	0.585	0.735	0.650	0.740	0.745	**0.750**
D7	**0.792**	0.779	0.752	0.750	0.711	0.784	0.751	0.780
D8	0.925	0.924	0.917	0.926	0.919	0.934	0.941	**0.942**
D9	0.563	0.497	0.281	0.670	0.425	0.798	0.857	**0.972**
D10	0.321	0.323	0.393	0.753	0.310	0.720	0.813	**1.000**
D11	0.567	0.552	0.490	0.654	0.645	0.654	0.664	**0.670**
D12	0.303	0.364	**0.642**	0.131	0.168	0.185	0.146	0.188
D13	0.387	0.421	**0.501**	0.292	0.335	0.329	0.298	0.330
D14	0.479	0.401	0.278	0.433	0.363	**0.481**	0.444	0.478
D15	0.196	0.136	0.018	0.514	0.513	0.517	0.546	**0.593**
D16	0.760	0.636	0.407	0.809	0.785	0.839	0.880	**0.986**
D17	0.720	0.586	0.378	0.856	0.765	0.891	0.930	**0.995**
D18	0.641	0.605	0.520	0.696	0.695	0.698	0.706	**0.723**
D19	0.713	0.713	0.697	0.772	0.763	0.774	0.775	**0.777**
D20	0.197	0.207	**0.536**	0.158	0.164	0.163	0.156	0.160
D21	0.305	0.346	0.178	0.405	0.316	0.444	0.452	**0.486**
D22	0.398	0.401	0.344	0.452	0.300	0.476	0.469	**0.496**
D23	0.303	0.395	0.263	0.406	0.256	0.420	0.417	**0.421**
D24	0.479	0.387	0.248	0.448	0.276	0.503	0.464	**0.513**
D25	0.434	0.380	0.308	0.512	0.399	**0.545**	0.533	0.529
D26	0.659	0.677	0.526	0.715	0.619	0.747	0.729	**0.758**
D27	0.382	0.351	0.320	0.343	0.295	0.422	0.383	**0.461**
D28	0.478	**0.486**	0.478	0.365	0.441	0.422	0.379	0.408
D29	0.112	**0.113**	0.097	0.079	0.094	0.092	0.075	0.096
D30	**0.500**	0.461	0.334	0.206	0.320	0.238	0.198	0.236
D31	**0.301**	0.270	0.251	0.135	0.205	0.165	0.156	0.156

Table 6. *AUC* Comparison

data	Oversampling			Undersampling			MUEnsemble	
	SMT	PWS	PFS	RUS	RBS	EE	Uni	Gau
D1	0.814	0.805	0.798	0.767	0.743	0.810	0.816	**0.820**
D2	0.994	0.993	0.995	0.982	**0.998**	0.997	0.997	0.996
D3	0.927	0.901	0.930	0.928	0.928	0.934	**0.936**	**0.936**
D4	0.758	0.759	0.770	0.769	0.768	**0.775**	0.774	**0.775**
D5	0.976	0.977	0.976	0.976	0.879	0.977	0.977	**0.980**
D6	0.912	0.911	0.917	0.913	0.861	0.917	**0.918**	**0.918**
D7	0.975	0.968	0.972	0.967	0.946	0.976	0.975	**0.978**
D8	0.979	0.980	0.979	0.978	0.977	0.981	**0.982**	**0.982**
D9	0.734	0.654	0.635	0.685	0.614	0.769	0.772	**0.784**
D10	0.791	0.722	0.740	0.802	0.711	0.846	0.883	**0.894**
D11	0.842	0.836	**0.843**	0.835	0.830	0.836	0.835	0.838
D12	0.976	0.971	0.960	0.973	0.961	0.978	0.980	**0.981**
D13	0.956	0.948	0.965	0.964	0.955	0.969	**0.972**	0.971
D14	0.902	0.908	0.900	0.902	0.840	0.932	0.933	**0.940**
D15	0.601	0.597	0.647	0.648	0.649	**0.652**	**0.652**	**0.652**
D16	0.819	0.789	0.790	0.826	0.825	0.841	**0.842**	0.839
D17	0.860	0.842	0.840	0.886	0.861	0.905	0.905	**0.913**
D18	0.850	0.846	0.853	0.862	0.862	0.864	0.865	**0.866**
D19	0.900	0.902	0.910	0.912	0.907	0.912	0.912	**0.913**
D20	0.993	0.993	**0.997**	0.996	0.996	**0.997**	**0.997**	**0.997**
D21	0.735	0.749	0.695	0.688	0.629	0.737	0.762	**0.785**
D22	0.763	0.741	0.728	0.745	0.618	0.805	**0.822**	0.812
D23	0.734	0.763	0.727	0.743	0.668	**0.802**	0.795	0.775
D24	0.839	0.830	0.819	0.791	0.652	0.851	0.855	**0.861**
D25	0.805	0.808	0.795	0.792	0.706	0.816	**0.826**	0.808
D26	**0.929**	0.922	0.907	0.904	0.884	0.921	0.928	0.923
D27	0.750	0.760	0.766	0.715	0.692	0.807	0.807	**0.812**
D28	0.861	0.895	0.867	0.909	0.902	0.937	0.920	**0.943**
D29	0.767	0.776	0.753	0.748	0.708	0.815	**0.823**	0.816
D30	0.882	0.854	0.845	0.856	0.877	0.923	0.911	**0.929**
D31	0.768	0.675	0.702	0.690	0.654	0.805	**0.834**	0.731

4.2.2. Q2: How well can ensemble classifiers perform when multi-ratio undersampling is taken into account?—The fact that MUEnsemble fared better than EasyEnsemble in the majority of cases and similarly in the others indicates how effective it is.

EasyEnsemble (EE) was the most similar technique to MUEnsemble among the undersampling-based methods since it combines several weak classifiers with a predetermined undersampling ratio. The impact of using several sample ratios is demonstrated by the variations in recall, gmean, and F2 scores between MUEnsemble and EasyEnsemble.

4.2.3. Question 3: Could the Gaussian weighting function be superior to the uniform layering function?—Indeed, there was a definite advantage to the Gaussian weighting function over the uniform one.

When it came to all metrics, MUEnsemble with the Gaussian weighting function was generally better than that with the uniform weighting function. This suggests that MUEnsemble does better than simply voting from the base classifiers due to certain factors.

The impacts of the Gaussian weighting function's parameters are compared to those of the uniform weighting function in Figures 4 and 5. Since D4 and D20 illustrate the effects' usual tendency, they are included in these figures to prevent duplication in displaying these effects across all datasets. The dashed horizontal lines in these graphs represent MUEnsemble's performance changes with the uniform weighting function, whereas the solid lines represent changes with the Gaussian weighting function. MUEnsemble was sensitive to the D4 settings, as seen in Figure 4. For example, MUEnsemble's recall increased with the lowest $\mu = 0.2$ (blue line in the picture); yet, this resulted in low gmean scores, which implies that low μ leads to low true negative ratios. On the other hand, Figure 5 demonstrates that MUEnsemble was not particularly sensitive to the D20 settings. According to both figures, MUEnsemble's performances are more closely aligned with the Gaussian and uniform weighting functions the bigger the $\sigma 2$. This is a natural phenomenon since the Gaussian function's distribution becomes flatter as $\sigma 2$ increases.

4.3. Knowledge Acquired

• Precision is preferred by oversampling, whereas recall is preferred by undersampling. This is noteworthy, but not all that surprising. By increasing the density of the data points that represent a minority class, oversampling techniques help classifiers build a boundary around those examples. These methods create examples of a minority class based on the distribution of observed examples. This suggests that these classifiers can attain high precision by confidently classifying unseen samples within the minority class border. Undersampling techniques, on the other hand, randomly alter instances of the majority class while leaving the minority class examples unchanged.

Classifiers can learn a more lenient boundary between minority and majority classes based on the skewed majority examples. This suggests that these classifiers enthusiastically assign the minority class to cases that are similar to the observed minority examples. Consequently, methods based on undersampling improve recall.

• Undersampling with varying sampling ratios has varying class preferences. A common strategy for improving performance on classification tasks is ensemble learning, which favors combining classifiers with different classification criteria. To achieve divergence, EasyEnsemble combines classifiers trained on several subsets of instances. On the other hand, by employing various sample ratios, the suggested approach, MUEnsemble, increases diversity. Class preferences vary among base classifiers trained on examples with varying sampling ratios. This paper's experiment demonstrated MUEnsemble's superiority over EasyEnsemble. Therefore, a better ensemble classifier can be achieved by increasing divergence by adding numerous sampling ratios.
•Simple voting examples can be enhanced with a biased ensemble weighting method. Combining different sampling ratios, as was previously said, broadens the variety of classifiers; yet, classifiers with excessively low or high sampling ratios may be overly biased toward the minority class (or majority class). Overly biased classifiers have a high degree of confidence in incorrectly classifying examples. Ensemble classifier performance may suffer as a result. Weights should be slanted toward the "balanced" sample ratios, according to the experimental results, which showed that MUEnsemble with the Gaussian weighting function performed better than that with the uniform weighting function. "Balanced" in this context refers to the sampling ratio being balanced for both the true positive and true negative rates, not to the sampling ratio equaling 1.0.

5. Conclusion

This research proposed MUEnsemble, an ensemble classification framework that applies multiple undersamplings with varying sampling ratios to the class imbalance problem. The flexibility of the Gaussian balancing function was demonstrated in an experiment. Promising future directions would be examining causes, such as the potential role of small disjuncts [16] and the overlap of example distributions of the majority and minority classes [24], and then combining solutions to the reasons for the degradation into MUEnsemble, since there were still datasets (like D15) for which none of the approaches could perform well. Another shortcoming of MUEnsemble and undersampling-based approaches was identified by this evaluation: they tended to have lower precision ratings than oversampling-based approaches. The class imbalance problem still involves the precision-recall trade-off.

References

1. J. Alcala´-Fdez, A. Ferna´ndez, J. Luengo, J. Derrac, and S. Garc´ıa. 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. H. Bao and M. Sugiyama. Calibrated Surrogate Maximization of Linear-fractional Utility in Binary Classification. *CoRR*, abs/1905.12511, 2019.
3. 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.
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 Algo- rithm 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. D. Dua and C. Graff. UCI Machine Learning Repository, 2019.
10. C. Elkan. The Foundations of Cost-Sensitive Learning. In *IJCAI 2001*, pages 973–978, 2001.
11. M. Galar, A. Ferna´ndez, 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.
12. 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.
13. S. E. Go´mez, L. Herna´ndez-Callejo, B. C. Mart´ınez, and A. J. Sa´nchez-Esguevillas. Exploratory study on Class Imbalance and solutions for Network Traffic Classification. *Neurocomputing*, 343:100–119, 2019.
14. 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.
15. 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.
16. T. Jo and N. Japkowicz. Class Imbalances versus Small Disjuncts. *SIGKDD Explorations*, 6(1):40– 49, 2004.
17. P. Kang and S. Cho. EUS SVMs: Ensemble of Under-Sampled SVMs for Data Imbalance Problems. In *ICONIP 2006*, pages 837–846, 2006.
18. G. Kova´cs. 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. G. Kov´acs. smote-variants: a Python Implementation of 85 Minority Oversampling Techniques. *Neurocomputing*, 366:352–354, 2019. (IF-2019=4.07).
20. B. Krawczyk, M. Wozniak, and G. Schaefer. Cost-sensitive decision tree ensembles for effective imbalanced classification. *Appl. Soft Comput.*, 14:554–562, 2014.
21. G. Lemaˆıtre, F. Nogueira, and C. K. Aridas. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.
22. 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.
23. 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.
24. J. Luengo, A. Ferna´ndez, S. Garc´ıa, and F. Herrera. Addressing data complexity for imbalanced data sets: analysis of SMOTE-based oversampling and evolutionary undersampling. *Soft Comput.*, 15(10):1909–1936, 2011.
25. 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.
26. H. M. Nguyen, E. W. Cooper, and K. Kamei. Borderline over-sampling for imbalanced data classification. *IJKESDP*, 3(1):4–21, 2011.
27. 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.
28. A. D. Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi. Calibrating Probability with Under-sampling for Unbalanced Classification. In *SSCI 2015*, pages 159–166, 2015.
29. D. Rodr´ıguez, 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.
30. R. E. Schapire. A Brief Introduction to Boosting. In *IJCAI 1999*, pages 1401–1406, 1999.
31. 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.
32. 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.
33. S. Sharma, C. Bellinger, B. Krawczyk, O. R. Za¨ıane, and N. Japkowicz. Synthetic Oversampling with the Majority Class: A New Perspective on Handling Extreme Imbalance. In *ICDM 2018*, pages 447–456, 2018.
34. 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.
35. I. Tomek. Two Modifications of CNN. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(11):769–772, 1976.
36. J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo. OpenML: networked science in machine learning. *SIGKDD Explor.*, 15(2):49–60, 2013.
37. 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.

D. L. Wilson. Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. *IEEE Trans. Systems, Man, and Cybernetics*, 2(3):408–421, 1972.