GUM: A Guided Undersampling Method to Preprocess Imbalanced Datasets for Classification

Kisuk Sung[1], W. Eric Brown[2], Erick Moreno-Centeno[3], and Yu Ding[3]

***Abstract*— In *imbalanced* datasets, where the *majority class* has significantly more instances than the *minority class*, conventional classification methods exhibit poor minority-class detection performance because they tend to classify most instances as majority instances. To address this problem, this paper presents a general-purpose imbalanced-data preprocessing method that combines two instance-selecting techniques to obtain a clean and balanced set of training instances. The first technique, *ensemble outlier filtering*, removes outlier instances from both majority and minority classes. The second technique, *normalized-cut sampling*, samples the majority class aiming to preserve its distribution across the majority region. Our proposed data preprocessing method uses these two techniques and can be combined with any general classification methodology on the sub-sampled data to construct a classification model. Computational results show the proposed method outperforms several widely used imbalanced-data classification methods.**

I. Introduction

In an *imbalanced dataset*, the *majority class* has significantly more instances than the *minority class*. On such datasets, conventional classification methods tend to perform poorly because they are typically designed to minimize the number of misclassified instances over all the training data. In other words, conventional methods may have relatively low misclassification rates when applied to imbalanced data, even if they correctly classify very few of the minority instances. This paper proposes a guided undersampling method (GUM) for preprocessing imbalanced data to boost the performance of subsequent classification. GUM first uses a modified ensemble-based outlier-filtering technique based on [1], which removes outliers from *both* minority and majority classes, and then applies a new normalized-cut sampling method, which uses normalized-cut clustering [2] to form a majority-class sample aiming to preserve the distribution across the majority region.

GUM is a new sampling method that outperforms alternative methodologies. Specifically, this paper makes the following contributions: (1) Our ensemble outlier-filtering technique in GUM is specifically designed for imbalanced data and thus achieves strong outlier detection performance on *both* majority and minority classes. In particular, the minority-class outlier identification and removal strategy in GUM is a noteworthy contribution because few existing approaches attempt to remove minority outliers. Moreover, our methodology utilizes the power of ensemble methods to identify outliers; while other techniques identify minority outliers using simple rulesets such as those incorporating Tomek Links [3]. Despite the scarcity of approaches to remove minority outliers, as Section III-A argues and Section V demonstrates, removing minority outliers (in addition to majority outliers) is important to obtain a high-performing classification model. (2) The normalized-cut sampling in GUM aims to obtain a subset of the majority instances that preserves the distribution of majority instances across the majority class region. To the best of our knowledge, no previous undersampling-based classification method has attempted to obtain such a "distribution-preserving" subset.

The remainder of the paper is organized as follows. Section II presents a literature review of imbalanced-data classification and widely used imbalanced-data preprocessing approaches. Section III details the two instance-selecting techniques that compose GUM. Section IV describes the datasets and computational setup under which we test GUM. Section V describes the numerical experiments and outcomes. Finally, Section VI summarizes our undertaking.

II. Literature Review

Data preprocessing approaches to address data imbalance can be categorized as sampling methods, synthetic-data generation methods, and outlier-removal methods.

Sampling methods balance the number of instances between the classes by oversampling minority instances and/or undersampling majority instances. Oversampling increases the number of minority instances by resampling the minority instances with replication [4], [5]. Oversampling approaches fail to address the fundamental issue (lack of minority instances) because they simply append replicated instances to the original dataset. Thus, the constructed classifier risks over-fitting the repeated instances [6]. Undersampling decreases the number of majority instances by removing instances from the majority class. For example, [6] randomly undersampled the majority instances without replacement; while [7] used weighted no-replacement undersampling with greater weight on the majority class near the minority class. Due to loss of majority class, undersampling techniques might not construct an accurate classification model [4], [8], [9]. Some researchers have used oversampling and undersampling simultaneously [4], [5], [10]. To address the aforesaid drawback of undersampling methods, some studies undersample the majority class using clustering techniques as a guide. For example, [11] aimed to obtain a majority class sample consisting of more majority instances in the regions where majority instances dominate minority instances. In

[1]Kisuk Sung* is with Samsung Life Insurance, Seoul, 06620, Korea

[2]W. Eric Brown* is with the Rawls College of Business, Texas Tech University

[3]Erick Moreno-Centeno* and Yu Ding are with the Wm Michael Barnes'64 Department of Industrial and Systems Engineering, Texas A&M University, College Station, Texas, USA, emc@tamu.edu and yuding@tamu.edu

contrast, [12] designed an undersampling method where majority instances near the class boundary are more likely to be selected as sampled data. Despite the improved performance, these cluster-based undersampling methods risk an inaccurate classification model if the selected instances are not sufficiently spread across the majority class region and thus form a poor representative sample of the majority class.

Synthetic-data generation methods generate artificial minority instances (i.e., minority instances different from those in the original dataset). The synthetic minority oversampling technique (SMOTE) generates a synthetic data instance for each minority instance between the selected minority instance and one of its minority-class neighbors [13]. Borderline SMOTE (BSMOTE) identifies minority instances located near the class boundary and, for each such instance, generates synthetic data instances using the SMOTE data-generating process [14]. The absent data generator (ADG) [15] relies on two criteria for creating synthetic minority instances: (i) new data should be close to the boundary between the majority and minority classes, and (ii) new data should be sufficiently close to existing minority points.

Outlier-removal methods are also used in imbalanced-data preprocessing. While the importance of outlier removal is well known, to the best of our knowledge, there are very few imbalanced-data-classification studies that identify/remove outliers from both the majority and minority classes. Early outlier-filtering techniques used a single classifier to identify outlier instances. For example, in a seminal paper, training instances that were not correctly classified by a k-NN classifier were deemed as outliers [16]. Similarly, [17] used a C4.5 classifier to identify and remove instances whose class label differed from the class predicted on each tree leaf. That said, some extreme methods remove all minority data and resort to a one-class classification approach [18]. The drawback of single-classifier methods is that they implicitly assume that the classifier being used is the most appropriate for the data. Because ensemble outlier-filtering methods remove this assumption, they are more robust than single-classifier methods [19]. Ensemble-based outlier-filtering techniques use either different base classification algorithms [19] or different subsets of the training set to train multiple classifiers [1]. We note that general purpose of anomaly detection methods (e.g., [20], [21]) can be used for identifying the outliers. But the main purpose here differs from anomaly detection.

One additional set of approaches that merit mentioning are **cost-sensitive** methods. These techniques do not preprocess the data but instead address the imbalance directly by assigning asymmetric costs to misclassified majority and minority instances. Specifically, the misclassification cost of the minority class, C_{minor} is higher than that of the majority class, C_{major}. A frequently used parameter within cost-sensitive techniques is the so-called ***imbalance ratio*** $r := C_{minor}/C_{major}$. For example, [22]–[24] used the cost-sensitive approach within decision-tree classifiers and [25]–[27] used it within support vector machines (SVM). The main drawback of cost-sensitive techniques is that their classification performance is sensitive to the cost ratio used and finding the optimal cost ratio is challenging [5], [15].

III. Proposed Guided Undersampling Method

GUM entails two algorithmic elements, detailed in the subsequent subsections.

A. Ensemble Outlier-filtering Technique

As ensemble outlier-filtering techniques were not specifically designed for imbalanced datasets, the base classifiers in the ensemble perform poorly on minority instances. Thus they misclassify most minority instances as majority instances and mark them as outliers. The approach in [1] is a step forward in addressing the data-imbalance problem. Still, it has the shortcoming that there may be insufficient minority instances in each subset when building the partition.

Our ensemble filtering addresses the aforesaid shortcoming of the method in [1] with the following two simple but critical modifications, which allow each training subset in the ensemble to be balanced. First, our method sets the number of classifiers in the ensemble to be (approximately) equal to the imbalance ratio r and partitions the majority class in r disjoint subsets. Second, each classifier in the ensemble is trained using all minority instances and one of the r majority-class partitions. Our modifications have these advantages: they are computationally inexpensive, they dampen the impact of class imbalance, and they leverage the power of ensemble methods. Most importantly, our improved technique demonstrates a strong outlier-detection performance.

Given training data $D_{train} = D_{maj} \cup D_{min}$, with imbalance ratio $r = |D_{maj}|/|D_{min}|$, our ensemble outlier-filtering technique for imbalanced data works as follows (steps 3 and 4 are unchanged from the original method in [1]):

1) Partition D_{maj} into r equally-sized subsets and build r distinct training subsets; each composed of D_{min} and one of the partitions of D_{maj}. Note that each training subset is balanced.
2) Train r classifiers, each on a different training subset.
3) Predict the class of each instance in D_{train} using the majority voting scheme.
4) Remove the *outliers* (instances whose predicted class differs from their true class) from the training set.

In Step (2), one may choose any general-purpose classifier. The objective of the outlier-filtering process is to construct a classifier without bias from outliers. Note that our filtering process removes outliers from both majority and minority classes. Removing minority instances seems counter-intuitive due to their scarcity. However, removing minority outliers is critical to constructing an accurate classifier—perhaps even more important than removing majority outliers, whose adverse effects are dampened by the presence of all the other numerous non-outlier majority instances.

We illustrate the impact of adequately removing minority instances via the example in Fig. 1. Figure 1(a) plots the original imbalanced data and the class boundary constructed by a cost-sensitive SVM, whereas Fig. 1(b) shows the

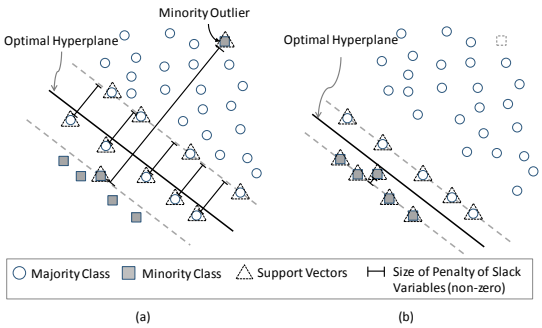

Fig. 1. Illustration of ensemble outlier-filtering procedure

minority-outlier filtered data and the similarly constructed class boundary. Note that, due to the higher penalty used on the misclassified minority instance, the single outlier biases the boundary causing five majority instances to be misclassified. In contrast, Figure 1(b) shows that when removing the minority outlier, the decision boundary correctly separates the majority and minority classes.

B. Normalized-cut Sampling

As Section II argued, traditional undersampling methods risk an inaccurate classification model due to the loss of majority instances. Specifically, if the sampled majority instances are not spread out over the majority class region, the decision boundary could be incorrect near areas without sampled instances. To address this problem, we propose a new cluster-based undersampling method, *normalized-cut sampling*, which undersamples the majority instances so that the chosen instances aim to preserve the majority-class distribution. Normalized-cut sampling first runs recursively normalized-cut clustering [2] to group the majority instances into a pre-specified number of approximately balanced clusters. Then it forms the majority-class sample by including the medoid (the instance with the smallest average distance to all instances in the given cluster) of each cluster.

The n-dimensional data points are represented with an undirected complete graph $G = (V, E)$, where V is the set of instances and E contains an edge between each pair of instances. The edge weight between nodes i and j, w_{ij}, is a measure of the similarity between instances i and j. Shi and Malick [2] defined w_{ij} as:

$$w_{ij} = e^{-\|x_i - x_j\|^2}, \tag{1}$$

where x_i and x_j are n-dimensional data of instance i and j, respectively. The clustering problem is formulated as a graph partitioning problem that minimizes the following function:

$$Ncut(S, \bar{S}) = \frac{cut(S, \bar{S})}{assoc(S, V)} + \frac{cut(S, \bar{S})}{assoc(\bar{S}, V)}, \tag{2}$$

where $cut(S, \bar{S})$ is the sum of edge-weights between the two partitions S and $\bar{S}$ and $assoc(S, V)$ is the sum of edge-weights from nodes in S to all nodes in the graph G; $assoc(\bar{S}, V)$ is similarly defined.

Our normalized-cut sampling method is built on top of Shi and Malick's normalized cut [2]. Given training data $D_{train} = D_{maj} \cup D_{min}$, our normalized-cut sampling undersamples the majority instances so that the number of sampled majority instances is the same as the number of minority instances, as follows:

1) Construct $G_1 = (V, E)$: Assign all majority instances to the node-set V and all node-pairs in V to the edge set E; assign edge-weights with Eq. (1).
2) For $k = 1, \ldots, |D_{min}| - 1$ do
   a) Bipartition the graph G_k using normalized-cut clustering [2].
   b) Construct a new undirected complete graph G_{k+1} including only the instances in the cluster with the largest cardinality (among the $k + 1$ clusters formed so far).
3) Form a subset comprising the medoid of each cluster and return this subset as the new and balanced training set.

Our normalized-cut sampling has two desirable properties. First, the new training set is balanced. Second, the sampled majority set tends to preserve the density distribution of the majority class. The first property follows since the sampled data comprises the medoid from each cluster and the number of clusters equals the number of minority instances. The second property follows since high-density regions will have more clusters than low-density regions *and* clusters have approximately the same cardinality. The reason behind why high-density regions have more clusters than low-density regions is because instances within a cluster are similar, i.e., close to each other, whereas instances in distinct clusters are different, i.e., far from each other.

C. Implementation of GUM

While GUM can work with any general-purpose classifier, here we use SVM as the base classifier. SVM has several properties that make it a competitive choice as base classifier. Specifically, SVM is a strong-performing method in its own right, can be straightforwardly modified to include cost sensitivity, and has paired well with other data preprocessing techniques in the past. The procedure outlined below describes how GUM is implemented and paired with SVM.

1) Given training data $D_{train} = D_{maj} \cup D_{min}$ with imbalance ratio $r = \frac{|D_{maj}|}{|D_{min}|}$, use our ensemble outlier-filtering technique (with SVM as the classifier in our filtering technique's second step) to obtain the outlier-filtered training data, $D'_{train} = D'_{maj} \cup D'_{min}$.
2) Apply to D'_{train} normalized-cut sampling, returning D''_{maj}, which is the spread-out sample of D'_{maj}.
3) Apply the SVM algorithm on the balanced data $D''_{maj} \cup D'_{min}$ to construct the classification model.

IV. Experimental Design

This section describes the experimental setup under which we analyze the performance of GUM on 11 datasets and compare it against five commonly used imbalanced-data classification approaches. To make a fair comparison and because our purpose is to compare the data preprocessing

techniques rather than the base classifier, we used SVM (using a radial basis function as the kernel function) as the base classifier for all methods. For each dataset and classification method, we used cross-validation with stratified sampling to determine the kernel-width parameter, the cost ratio between the margin penalty, and the total sum of the slack variables. All methods, including ours, were implemented in the LIBSVM tool package in MATLAB [28].

A. Methods in Comparison

GUM is compared against the following five classification methods: the raw-data base classifier, cost-sensitivity, random-undersampling, SMOTE, and BSMOTE. The imbalance ratio of a given original dataset is denoted as r hereafter.

Raw-data SVM (Raw) is the traditional SVM [29] applied directly to the raw data. Cost-sensitive (CS) SVM is a traditional SVM but setting its cost ratio, C_{minor}/C_{major}, equal to r. Random-undersampling (Rand) randomly undersamples the majority instances to obtain a balanced dataset and then applies a traditional SVM on the sampled data. SMOTE and BSMOTE were explained in the literature review section. For both SMOTE and BSMOTE, sufficient synthesized data is generated to achieve a balanced data set; this new balanced data set is then used to train a traditional SVM classifier.

B. Performance Measure

We use the area under the curve (AUC) measure of the receiver operating characteristic (ROC) plot [30]. AUC is a well-established metric to evaluate models in the imbalanced-data classification literature. Indeed, according to some researchers, AUC (and related techniques) are more appropriate than error rates to measure model success in the context of imbalanced data and disparate error costs (see e.g., [13], [14], [23], [30]).

The ROC curve is created by plotting each $(False\ alarm,\ Detection\ power)$ point for the test data at various threshold settings, where false alarm is the false positive rate and the detection power is the true positive rate. After plotting the ROC curve, the AUC is computed by measuring the area under the curve. A higher AUC implies better performance.

C. Datasets

All of the base datasets used in this study have the following important characteristics: (1) they are open to the public (all are from the UCI Repository [31]); (2) they are widely used in previous imbalanced-data classification studies; and (3) the AUC obtained from the raw-data SVM on the dataset with an imbalance ratio of 5 was less than 0.95. This third characteristic is important because it is very difficult to accurately discriminate the methods' performances on datasets that are relatively easy even for the naive raw-data SVM. Table I lists the 11 base datasets used in this study. Since some of the base datasets are not particularly imbalanced and all have different imbalance ratios, we generated our test datasets as described next:

TABLE I
DATASET DESCRIPTION

Dataset	Dim	# of Inst.	# of Maj.	# of Min.
Australian	14	690	383	307
CMC	24	1473	1140	333
Ecoli	9	336	301	35
German	24	1000	700	300
Glass	9	214	197	17
Haberman	3	306	225	81
Heart	13	270	150	120
Liver	6	345	200	145
Pima	8	768	500	268
Vehicle	18	846	634	212
Yeast	9	1484	1055	429

1) In order to be able to measure the methods' classification performances on datasets with comparable imbalance ratios, we created four imbalanced datasets (with imbalance ratios of 5, 10, 20, and 30, resp.) from each base dataset. This gives a total of 44 datasets.
2) In addition, to minimize any effect from the random sampling used to generate the 44 datasets, we repeated the above procedure 10 times for each dataset-ratio pair. Hereafter, these random datasets are referred to as *repetitions*. Altogether we generated 440 total repetitions (10 repetitions of each of the 44 datasets).

V. EXPERIMENTAL RESULTS

A. Performance Analysis of GUM

Table II lists the AUC obtained by each classification method on each dataset. The AUC values reported in Table II are the average AUC over the 10 repetitions, each obtained via fivefold cross-validation with stratified sampling. Values in bold-face denote the highest AUC for each dataset. Notably, GUM attains the highest AUC on 30 out of the 44 datasets. Moreover, excluding the Liver and Glass base datasets, GUM provides either the highest AUC or an AUC close to the highest value.

To assess the statistical significance of the competitiveness of different methods, we used the Friedman test, as revised in [32] to compare multiple classification methods. The Freidman test is a non-parametric statistical test for detecting differences between rank data. The null hypothesis is that the ranks of all classification methods are equivalent. If the null hypothesis is rejected, we used the post-hoc test to determine which classification method(s) have consistently a higher or lower rank relative to others.

For each dataset (i.e., each row in Table II), we ranked the classification methods in terms of the average of their ranks (over the 44 datasets), hereafter referred to as the *average AUC rank*; the lower value, the higher rank. Notably, GUM has the lowest average AUC rank value (1.89) among all classification methods. Figure 2 shows the Friedman test for statistical significance with the Nemenyi post-hoc analysis. The average AUC rank of each method is denoted with a triangle. The critical difference range obtained by the Nemenyi post-hoc analysis is depicted with the horizontal line segments. In this analysis, two methods are deemed significantly different when their critical difference ranges

TABLE II

AUC OF GUM AND ALTERNATIVE METHODS ON EACH DATASETS.

Dataset							
Base DataSet	Imbalance Ratio	Raw	CS	Rand	SMOTE	BSMOTE	GUM
Australian	5	0.911	0.904	0.908	0.880	0.886	**0.915**
	10	0.889	0.893	0.898	0.848	0.870	**0.915**
	20	0.735	0.839	0.885	0.813	0.838	**0.909**
	30	0.682	0.815	0.889	0.808	0.809	**0.901**
CMC	5	0.630	0.700	0.701	0.631	0.642	**0.711**
	10	0.577	0.670	0.690	0.589	0.604	**0.699**
	20	0.570	0.622	0.660	0.594	0.585	**0.691**
	30	0.568	0.595	0.662	0.605	0.592	**0.673**
Ecoli	5	0.937	**0.947**	0.931	0.889	0.908	0.937
	10	0.924	**0.943**	0.938	0.871	0.896	0.939
	20	0.785	0.930	0.934	0.849	0.879	**0.939**
	30	0.700	0.915	0.918	0.792	0.873	**0.929**
German	5	0.734	0.696	0.764	0.670	0.678	**0.770**
	10	0.695	0.669	0.751	0.666	0.689	**0.757**
	20	0.679	0.626	0.700	0.672	0.685	**0.731**
	30	0.633	0.587	0.656	0.659	0.661	**0.716**
Glass	5	0.780	**0.820**	0.677	0.809	0.779	0.712
	10	0.720	0.815	0.676	**0.819**	0.765	0.704
	20	0.578	0.710	0.650	**0.713**	0.676	0.650
	30	0.572	0.682	0.603	**0.705**	0.633	0.616
Haberman	5	0.602	0.668	0.645	0.630	0.624	**0.684**
	10	0.588	0.628	0.603	0.592	0.611	**0.640**
	20	**0.620**	0.591	0.589	0.565	0.604	0.619
	30	0.572	0.586	0.573	0.553	**0.600**	0.576
Heart	5	0.883	0.815	0.893	0.852	0.855	**0.900**
	10	0.843	0.800	0.882	0.852	0.862	**0.898**
	20	0.813	0.777	0.859	0.824	0.831	**0.880**
	30	0.734	0.750	0.856	0.835	0.829	**0.884**
Liver	5	**0.674**	0.655	0.611	0.663	0.651	0.634
	10	0.641	0.623	0.561	0.623	**0.643**	0.599
	20	0.588	0.589	0.568	**0.616**	0.611	0.557
	30	0.542	0.571	0.560	**0.600**	0.591	0.552
Pima	5	0.786	0.786	**0.828**	0.724	0.729	**0.828**
	10	0.682	0.758	0.798	0.709	0.720	**0.812**
	20	0.660	0.726	0.797	0.705	0.714	**0.809**
	30	0.636	0.675	0.784	0.682	0.695	**0.799**
Vehicle	5	**0.858**	0.808	0.796	0.776	0.778	0.836
	10	**0.791**	0.781	0.754	0.747	0.758	0.788
	20	0.653	0.699	0.697	0.718	0.713	**0.742**
	30	0.641	0.656	0.666	0.720	0.717	**0.728**
Yeast	5	0.706	0.756	0.782	0.671	0.679	**0.785**
	10	0.641	0.738	0.770	0.660	0.669	**0.779**
	20	0.621	0.702	0.759	0.642	0.660	**0.772**
	30	0.597	0.674	0.733	0.625	0.651	**0.768**

do not overlap. The Friedman statistic χ_F^2 is 57.68, and thus the null hypothesis is rejected with p-value 3.67×10^{-11}. Moreover, the post-hoc Nemenyi analysis (see Fig. 2) asserts that the average AUC rank of GUM is significantly higher than that of all tested methods (lower value / higher rank). This analysis suggests that GUM substantially outperforms the other five imbalanced-data methods when using SVM as the base classifier.

B. Minority Overlap Index

From Table II, it is evident that GUM failed to outperform most methods on two datasets: the Liver and Glass base datasets. While it is not surprising that no method consistently outperforms all others, it is helpful to understand the circumstances under which a method works well. For this reason, this section introduces the concept of *minority overlap index*, which quantifies the proportion of the minority-class region that falls within the majority-class region. We further argue that GUM's performance lags in datasets where the minority overlap index is exceptionally high.

To measure the minority overlap index empirically, we use the following procedure: (1) Identify the class boundary (and thus the region) of each class using a one-class SVM classifier. In this study, we used the one-class classifier in [33] with parameter $\nu = 0.5$. (2) Calculate the empirical minority overlap index by counting the number of minority instances inside the intersection of the majority and minority classification regions and dividing this number by the number of minority instances inside the minority class region. Effectively, this calculation empirically evaluates the ratio of the volumes of the area where the regions overlap and the minority region.

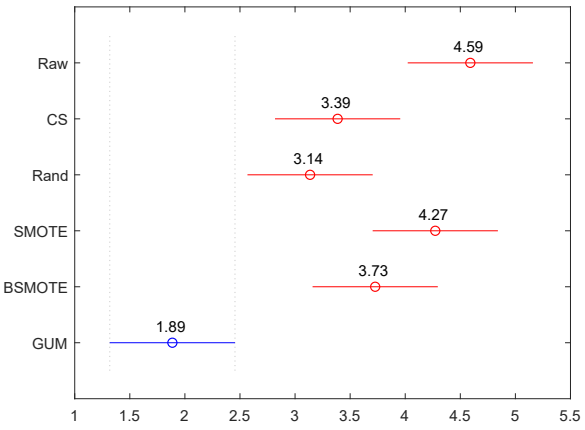

Fig. 2. Post-hoc analysis on the ranks of GUM and alternative methods.

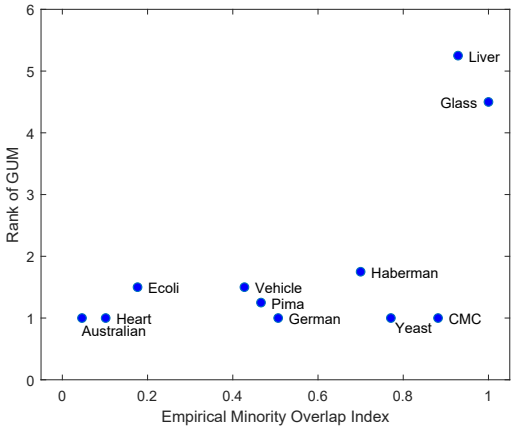

Fig. 3. Average AUC ranks of GUM vs. the empirical minority overlap index for each of the 11 base datasets. GUM shows limited performance in the datasets where the minority overlap index is above a certain threshold.

For each dataset, Figure 3 graphs the rank of GUM (averaged over the 4 imbalance ratios) versus the empirical minority overlap index. Notably, GUM is either the best or the second-best method in all but the liver and glass datasets. These two datasets have extremely high minority overlap indices (0.93 and 1, respectively), while all other datasets have indices less than 0.9. This analysis suggests that GUM performs poorly only when the empirical minority overlap index is very high ($\geq$ 0.9); in which case GUM should be applied with caution. Otherwise, when the empirical minority overlap index is below 0.9, GUM performs highly competitively.

VI. Conclusions

This work presents GUM, a new data-preprocessing method for imbalanced-data classification. The performance of GUM highlights the importance of two findings regarding imbalanced-data classification. First, outlier detection and removal from *both* classes are crucial for handling imbalanced data. In particular, the strong performance of GUM suggests that the relative novelty of our ensemble outlier-filtering method removing minority outliers is noteworthy. Second, researchers understand the importance of selecting representative subsets of data when undersampling the majority class. However, the issue of how best to attain this goal remains under debate. Our normalized-cut sampling method, which aims to select a majority class subsample that preserves the majority distribution, is intuitive and performs strongly.

There is a particular setting where GUM does not perform as well as its competitors. Specifically, GUM's performance lags when a substantial portion of the minority instances fall within the majority class region (i.e., the herein defined minority overlap index is high). Using this index, practitioners can apply GUM when its strength can be leveraged to maximum effect. We plan to investigate this performance lag in future research. Regardless, given the strong performance of GUM across the datasets herein discussed, this method should be strongly considered when addressing binary imbalanced-data classification problems in the future.

VII. Disclaimer

This paper is based partially on the Ph.D. dissertation of the first author at Texas A&M University, entitled "New Data Mining Techniques for Social and Healthcare Sciences."

References

[1] S. Verbaeten and A. Van Assche, "Ensemble methods for noise elimination in classification problems," in *Multiple Classifier Systems*. Springer Berlin Heidelberg, 2003, vol. 2709, pp. 317–325.

[2] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[3] I. Tomek, "Two modifications of CNN," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 7(2), pp. 679–772, 1976.

[4] A. Estabrooks, T. Jo, and N. Japkowicz, "A multiple resampling method for learning from imbalanced data sets," *Computational Intelligence*, vol. 20, no. 1, pp. 18–36, 2004.

[5] E. Byon, A. K. Shrivastava, and Y. Ding, "A classification procedure for highly imbalanced class sizes," *IIE Transactions*, vol. 42, no. 4, pp. 288–303, 2010.

[6] D. Mease, A. J. Wyner, and A. Buja, "Boosted classification trees and class probability/quantile estimation," *Journal of Machine Learning Research*, vol. 8, pp. 409–439, 2007.

[7] N. Japkowicz, "The class imbalance problem: Significance and strategies," in *Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI)*, 2000, pp. 111–117.

[8] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: One-sided selection," in *Proceedings of the Fourteenth International Conference on Machine Learning*, vol. 97, 1997, pp. 179–186.

[9] X. Peng, X. Jin, S. Duan, and C. Sankavaram, "Active learning assisted semi-supervised learning for fault detection and diagnostics with imbalanced dataset," *IISE Transactions*, online published, 2022.

[10] G. Weiss and F. Provost, "The effect of class distribution on classifier learning: An empirical study," ML-TR-44, Department of Computer Science, Rutgers University, New Jersey, Tech. Rep., 2001.

[11] S.-J. Yen and Y.-S. Lee, "Cluster-based under-sampling approaches for imbalanced data distributions," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5718–5727, 2009.

[12] D. Wang and M. Shi, "Density weighted region growing method for imbalanced data SVM classification in under-sampling approaches," *Journal of Information and Computational Science*, vol. 11, pp. 6673–6680, 2014.

[13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[14] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," in *Advances in Intelligent Computing*. Springer, 2005, pp. 878–887.

[15] A. Pourhabib, B. K. Mallick, and Y. Ding, "Absent data generating classifier for imbalanced class sizes," *Journal of Machine Learning Research*, vol. 16, pp. 2695–2724, 2015.

[16] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-2, no. 3, pp. 408–421, 1972.

[17] G. H. John, "Robust decision trees: Removing outliers from databases," in *Knowledge Discovery and Data Mining*. AAAI Press, 1995, pp. 174–179.

[18] C. Park, J. Z. Huang, and Y. Ding, "A computable plug-in estimator of minimum volume sets for novelty detection," *Operations Research*, vol. 58, pp. 1469–1480, 2010.

[19] C. E. Brodley and M. A. Friedl, "Identifying mislabeled training data," *Journal of Artificial Intelligence Research*, vol. 11, pp. 131–167, 1999.

[20] I. Ahmed, X. B. Hu, M. P. Acharya, and Y. Ding, "Neighborhood structure assisted non-negative matrix factorization and its application in unsupervised point-wise anomaly detection," *Journal of Machine Learning Research*, vol. 22(34), pp. 1–32, 2021.

[21] I. Ahmed, T. Galoppo, X. Hu, and Y. Ding, "Graph regularized autoencoder and its application in unsupervised anomaly detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, online published, 2021.

[22] P. Domingos, "Metacost: A general method for making classifiers cost-sensitive," in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1999, pp. 155–164.

[23] M. A. Maloof, "Learning when data sets are imbalanced and when costs are unequal and unknown," in *International Conference on Machine Learning (ICML)-2003 Workshop on Learning from Imbalanced Data Sets II*, vol. 2, 2003.

[24] C. X. Ling, Q. Yang, J. Wang, and S. Zhang, "Decision trees with minimal costs," in *Proceedings of the Twenty-first International Conference on Machine learning*. ACM, 2004, p. 69.

[25] Y. Lin, Y. Lee, and G. Wahba, "Support vector machines for classification in nonstandard situations," *Machine Learning*, vol. 46, no. 1-3, pp. 191–202, 2002.

[26] F. R. Bach, D. Heckerman, and E. Horvitz, "Considering cost asymmetry in learning classifiers," *Journal of Machine Learning Research*, vol. 7, pp. 1713–1741, 2006.

[27] H. Masnadi-Shirazi and N. Vasconcelos, "Risk minimization, probability elicitation, and cost-sensitive SVMs." in *International Conference on Machine Learning (ICML)*, 2010, pp. 759–766.

[28] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[29] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[30] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[31] M. Lichman, "UCI Machine Learning Repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[32] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[33] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.