Fuzzy Systems and Data Mining VII
A.J. Tallón-Ballesteros (Ed.)
© 2021 The authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA210186

Mass Ratio Variance Majority Undersampling and Minority Oversampling Technique for Class Imbalance

Piboon Polvimoltham[a] and Krung Sinapiromsaran[a,1]
[a] *Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University, Bangkok, Thailand*

Abstract. A sampling method is one of the popular methods to deal with an imbalance problem appearing in machine learning. A dataset having an imbalance problem contains a noticeably different number of instances belonging to different classes. Three sampling techniques are used to solve this problem by balancing class distributions. The first one is an undersampling technique removing noises from a class having a large number of instances, called a majority class. The second one is an over-sampling technique synthesizing instances from a class having a small number of instances, called a minority class, and the third one is the combined technique of both undersampling and oversampling. This research applies the combined technique of both undersampling and oversampling via the mass ratio variance scores of instances from each individual class. For the majority class, instances with high mass ratio variances are removed whereas for the minority class, instances with high mass ratio variances are used in synthesizing minority instances. The results of this proposed sampling technique help improve recall over standard classifiers: a decision tree, a random forest, Linear SVM, MLP on all synthesized datasets; however it may have low precision. So the combined measure of precision and recall is used, F1-score. Recall and F1-scores of synthesized datasets and UCI datasets are significantly better for collections of datasets having small imbalance ratio. Moreover, the Wilcoxon signed-rank test is used to confirm the improvement for datasets having imbalance ratio smaller than or equal to 0.2.

Keywords. Mass ratio variance score, Undersampling, Oversampling, Imbalanced problem, Classification

1. Introduction

A class imbalanced problem[1] is one of the important topics in classification from machine learning. It is a problem of building a classifier in the presence of underrepresented class instances and highly skewed class distributions. This occurs when the number of instances representing an important class is much smaller than those from other classes. In a binary classification, the smaller class is called the minority class or the positive class while another class is called the majority class or the negative class. The main purpose of classification on this problem is to identify the minority instances

[1] Corresponding Author, Krung Sinapiromsaran, Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University, Bangkok, Thailand; E-mail: krung.s@chula.ac.th

as accurately as possible. In real world applications, minority instances are important such as fraud transactions in the fraud detection[2], ailing patients in the medical diagnosis[3], default loans in the credit approval[4]. In addition, a class imbalance problem in the medical diagnosis is to detect and diagnose the patterns of certain diseases within patient electronic healthcare records. It is normal that some life threatening diseases are rare among patients. The misclassification of these cases can lead to the patient's death so the ailing patients that identify as healthy should not be occurred, i.e. the number of false negative patients should be small. Traditionally, a minority instance tends to be misclassified when a standard classifier is applied on an imbalanced dataset due to its tiny portion.

There are three main methodologies to deal with an imbalanced problem. First, a data-level methodology[5, 6, 7, 8] resamples the distribution of class instances to make them balance. Then this new dataset can be used to train with any classifier. Many techniques in this approach are an oversampling technique[9, 10, 11] which synthesizes random instances from the minority group avoiding those from majority groups, or an undersampling technique[12, 13] which discards random instances from the majority group to extend the minority region of instances in the minority class or the mixture of an oversampling and undersampling technique[14]. Second, an algorithmic-level methodology upgrades or reimplements the classification algorithms to be more robust to noise while handling minority instances successfully[15, 16, 17, 18, 19]. Third, the hybrid methodology combines both the data-level approach and the algorithmic-level approach such as Adaboost[20], Boosting[21], Bagging[22], etc.

An undersampling algorithm concentrates on removing instances from the majority class, it reduces the total amount of information that the model has to learn from. Currently, there are many undersampling techniques such as DBMUTE 2017, MUTE, 2011, but a random undersampling algorithm (RUS) is the simplest method that removes minority instances randomly without any restriction. There are many intelligent approaches toward undersampling such as Tomek-link[23], it is the method that based on 1-Nearest-Neighbour, groups the borderline minority instance with nearest majority instance then removes those majority instances, this makes borderline unblemish and easy to partition. An oversampling algorithm contrasts this operation by increasing the number of minority instances. The simplest method is the random oversampling algorithm (ROS). It randomly duplicates instances from the minority class, which will not expand the region of the minority class. One of the popular oversampling techniques that expand the region of this class is the Synthetic minority oversampling technique (SMOTE)[5]. It produces artificial minority instances by interpolating between existing minority instances and their nearest minority neighbours. The enhanced SMOTE algorithm has been developed such as Borderline-SMOTE[6] and Safe-Level-SMOTE[7] that deal with some majority instances during the synthetic process.

A misclassified minority instance normally lies further away from other minority instances or abnormal minority instances. An algorithm to help a classifier to recognize them should remove some surrounding majority instances within the overlapping region. In addition, it should also synthesize a small number of minority instances near these minority instances. Hence, the resampling technique is proposed. In addition, the algorithm may be used to identify abnormal instances in the majority class for removal. This can be achieved using the anomaly score, the Mass-ratio variance based outlier factor(MOF)[24]. The algorithm to generate MOF requires no parameter and uses the density to assign high scores to outliers. This makes MOF perfect to detect those abnormal instances for majority and minority classes.

The mass ratio variance majority undersampling and minority oversampling technique (MUOT) is proposed. It uses MOF to detect abnormal instances in both majority and minority classes. An abnormal instance from the majority class is treated as noise which will be removed to clean up the overlapping area between both classes while abnormal instances from the minority class will be packed with synthesized minority instances. To evaluate the performance of the proposed method via precision, recall and F1-score, four standard classifiers will be executed on synthesized datasets and UCI datasets. Finally, the Wilcoxon signed-rank test will be used to demonstrate the effectiveness of the proposed method for unseen instances.

2. Related work and background knowledge

A mass-ratio-variance based outlier factor algorithm[24] is a parameter-free density-based outlier scoring algorithm. It gives scores to all instances from a dataset. The high score is given to an outlier whereas the low score is given to normal instances via the variance of mass-ratio scores. The following definitions are used to define MOF.

Definition 1: Given a dataset $D \subseteq \mathbb{R}^d$ the Euclidean distance of instance $x = (x_1, \ldots, x_d) \in D$to instance $y = (y_1, \ldots, y_d) \in D$ denoted as $d(x, y)$is defined as

$$d(x,y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

Definition 2: Given a dataset $D \subseteq \mathbb{R}^d$, the set of all instances within the neighbourhood of instances $x \in D$with respect to the radius r is defined as the set of points that lies within the ball centred at instance x with the radius r

$$N(x,r) = \{z \in D | d(x,z) \leq r\}$$

Definition 1 is the Euclidean distance definition and definition 2 defines the set of neighbourhoods of instance x with respect to the radius r. The next definition defines the mass-ratio of an instance with respect to another instance. The last definition defines the mass-ratio variance score of an instance.

Definition 3: For instance $y \in D$and instance $x \neq y \in D$, the mass-ratio of instance $y$with respect to instance $x$is defined as

$$mr_x(y) = \frac{|N(y, d(x,y))|}{|N(x, d(x,y))|}$$

For instance x, the definition 3 will assign mass-ratios to other instances in the dataset. If this instance x is outlier, the denominator will contribute a small number so that other instances will have high mass-ratios, except the one that is close to x. If this instance x is among other instances in a dataset, this mass-ratio will be close to 1.

Definition 4: $\bar{mr}_x$is defined as mean of the mass-ratio distribution of other instances, except x and MOF of instance x is defined as the variance of the mass-ratio distribution. These MOFs are used to separate abnormal instances from normal ones. They are used in the proposed method, MUOT, for a class imbalance problem. The criterion to separate abnormal instances from normal instances will explain in detail in the next section.

3. MUOT

MUOT uses mass-ratio-variance scores for undersampling and oversampling. Originally, the mass-ratio-variance score was designed to identify outliers of a static dataset by giving high MOF scores to outliers and low MOF scores to normal instances. MUOT is the algorithm that resamples imbalanced to balanced dataset. Both undersampling and oversampling steps are performed only on abnormal instances. The undersampling step is to remove abnormal instances from the majority class and oversampling step is to synthesize the minority instances into balls using abnormal instances from the minority class as center. To identify abnormal instances, a threshold for MOF scores must be selected. For a univariate data distribution, $Q_3 + 1.5IQR$ is suggested as the IQR rule to identify outliers. Note $IQR = Q_3 - Q_1$ where Q_3 is the 75th percentile and Q_1 is the 25th percentile. Nevertheless, to find an appropriate threshold for MUOT, 50th, 60th, 70th, 80th and 90th percentiles are investigated to compare with $Q_3 + 1.5IQR$ via the decision tree. After resampling techniques are applied to datasets with different thresholds. The one with the highest F1-score will be selected as the appropriate threshold for MUOT, see Figure 1. From these experiments, the best threshold is 90 percentile. So 10% of majority instances will be removed and 10% of minority instances will be used in the oversampling step to make sure that there are enough minority instances for a classifier to recognize these minority outcasts.

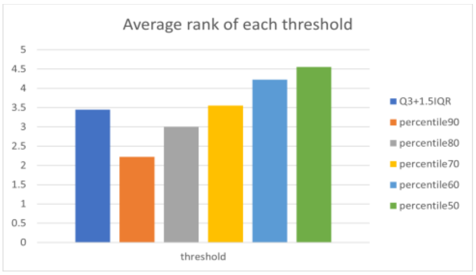

Figure 1: Average rank of each threshold compared

The following pseudocode demonstrates the step-by-step of the MUOT algorithm.

```
Algorithm MUOT(X, y, p)
Input: Array of data X; vector of target y; percentile threshold p (default = 90)
nFeatures = number of features, nPos = number of minority instances, nNeg = number of majority instances
Note: 1. $y_i$ is 0 for a majority instance and 1 for a minority instance
      2. # is used for a line comment
      3. compute_MOF(S) returns MOF scores of each instance in S
      4. percentile(S, p) returns the $p^{th}$ percentile value from S
      5. a group of selected instances is represented in a numpy array as Variable_name[*conditions* or *index*]
      6. ball(x, nFeatures, r, n) returns n synthesized instances within a ball centered at x and radius r.
      7. enumerate(S) returns a sequence of (i, s) where i is the index of s in S
      7. concatenate(S, T) returns the new set that concatenate T to S
Output: the balanced dataset X and y

1. XNeg is the subset of X having target y = 0
2. MOFNeg = compute_MOF(XNeg)
3. NegThreshold = percentile(MOFNeg, p)
4. NegAbnormal is the set of instances from XNeg having MOFNeg > NegThreshold
5. nNegAbnormal = the number of instances from NegAbnormal
6. X = is the subset of X removing NegAbnormal
```

```
7.  nNegNormal = nNeg - nNegAbnormal
8.  nSyn = nNegNormal - nPos  # nSyn will be the number of synthesized instance
9.  **if** nSyn > 0
10.   **then** # Do the oversampling step
11.       XPos is the subset of X having target y = 1
12.       MOFPos = compute_MOF(XPos)
13.       PosThreshold = percentile(MOFPos, p)
14.       PosAbnormal is the set of instances from XPos having MOFPos > PosThreshold
15.       nPosAbnormal = the number of instances from PosAbnormal
16.       nSynPos = nSyn*(MOFPos[PosAbnormal] / Sum(MOFPos[PosAbnormal]))
17.       NegIndex = index of nearest majority instance from instances in PosAbnormal
18.       radius = distance[NegIndex] # radius is set as the distance from the minority instance to its
    nearest majority instance
19.       for i, abnormal in enumerate(PosAbnormal)
20.           SynPos = ball(abnormal, nFeatures, radius[i], nSynPos[i])
21.           X = concatenate(X, SynPos)
22.           Y = concatenate(Y, 1)
23.       endfor
24. endif
25. return X, y
    End MUOT
```

4. Experiments and Results

This section reports the experimental results of the MUOT algorithm via four classifiers. The experimental datasets are grouped into three collections having the imbalance ratio (IR) of 0.1, 0.2 and 0.3. Note that the imbalance ratio is defined as the ratio between number of minority instances and number of all instances. IR values represent the ratio of imbalance data, a low IR value means there is a small number of minority instances (highly imbalanced) while a high IR value means the data is more balanced. In each collection, the subcollection is defined based on the number of clusters of 2, 3 and 4. Each collection will contain 3 subcollections, each subcollection is defined in 3D and 5D with 100 or 300 instances. Then the datasets are generated randomly 30 times and the average performance will be reported from these 30 datasets in each subcollection and each collection. The reported results are the average performance measures of each collection by each subcollection based on different classifiers and measures.

$$Imbalanced\ Ratio(IR) = \frac{|minority\ instances|}{|all\ instances|} \tag{1}$$

The performance measures are computed from the confusion matrix shown below. TP is the count of true positive instances: the actual class is positive and the predicted class is also positive. TN is the count of true negative instances: the actual class is negative and the predicted class is negative. FP is the count of false positive instances: the actual class is negative but the predicted class is positive. FN is the count of false negative instances: the actual class is positive but the predicted class is negative.

	Actual Positive	Actual Negative
Predicted Positive	TP: True Positive	FP: False Positive
Predicted Negative	FN: False Negative	TN: True Negative

Figure 2: Confusion matrix

In a class imbalance problem, three measures are more crucials than others which are TP, FN and FP. Note that FP is the number of negative instances that are predicted as positive and FN is the number of positive instances that are predicted as negative which should be very low for a class imbalance problem. So recall, see Equation 3, will be more emphasized than precision, see Equation 2. Nevertheless, to incorporate both measures, F1-score is used as the harmonic mean of precision and recall, see Equation 4.

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$F1-score = 2 \times \frac{Precision \times Recall}{Precision+Recall} \quad (4)$$

4.1 Synthesized data

There are 36 settings in the experiment which are grouped by IR and the number of clusters. Collection 1 has IR = 0.1, Collection 2 has IR = 0.2 and Collection 3 has IR = 0.3. Each collection also has three subcollections grouped by the number of clusters = 2, 3 and 4. In each subcollection, there are 4 settings varying by the number of features = 3 and 5 and the number of instances = 100 and 300 as shown in Table 1. In each setting, 30 datasets are randomly generated based on the provided setting.

Table 1. Information of synthesized datasets used in the experiment

Synthesized data				
Collection (IR)	**Subcollection**	**#clusters**	**#features**	**#instances**
1 (IR = 0.1)	1.1	2	3, 5	100, 300
	1.2	3	3, 5	100, 300
	1.2	4	3, 5	100, 300
2 (IR = 0.2)	2.1	2	3, 5	100, 300
	2.2	3	3, 5	100, 300
	2.3	4	3, 5	100, 300
3 (IR = 0.3)	3.1	2	3, 5	100, 300
	3.2	3	3, 5	100, 300
	3.3	4	3, 5	100, 300

The average performances of precision, recall and F1-score are reported in Figure 3 over four classifiers: a decision tree, a random forest, linear SVM, MLP comparing between the use of the original dataset and the dataset from the MUOT algorithm.

The results of the experiments are shown in Table 2 and Figure 3. Each cell in Table 2 reports mean±sd from each setting, where mean is the average performance and sd is the standard deviation. To easily see the increase and decrease of performance, Figure 3 shows the barplot of each measurement compared between the original and the MUOT datasets. It has a 3 by 3 barplots by the collections and the subcollections. But in this paper are shows a 2 by 2 barplots which the collection having IR= 0.1 and 0.3 and the subcollection having the number of clusters= 2 and 4. In case of full Figure 3 can be find at this following links[https://bit.ly/3k2SMvo].

From Table 2, all experiments show the decrease in precision and the increase in recall. For F1-scores, all experiments show improvement of this performance measure. Notice that for the number of clusters = 2, the improvement of recalls and F1-scores are small with respect to the larger number of clusters for all collections. The reason for this behavior is that by removing majority abnormal instances will expand the minority region so a classifier would be able to classify more minority instances which increases recall. However, this behavior will cause precision to decrease because the more minority prediction, the more chance that some majority instances will be predicted as minority instances which will decrease precision. Therefore, F1-score should be used to determine the performance of MUOT. This demonstrates that the MUOT algorithm can help classifiers to give a better improvement for the original datasets having IR less than or equal to 0.2 and the more number of clusters the better improvement from MUOT can be obtained.

Table 2. Mean±sd of precision, recall and F1-score of each collection

Collection	Subcollection		Mean±sd		
IR	#cluster	Models	precision	recall	F1-score
IR = 0.1	2	Original	0.6030 ± 0.1670	0.3974 ± 0.1077	0.4474 ± 0.1118
		MUOT	0.5035 ± 0.1200	0.5110 ± 0.1175	0.4767 ± 0.1063
	3	Original	0.4403 ± 0.1972	0.2399 ± 0.1054	0.2810 ± 0.1144
		MUOT	0.3774 ± 0.1463	0.3904 ± 0.1330	0.3499 ± 0.1268
	4	Original	0.4591 ± 0.0398	0.3446 ± 0.1746	0.3571 ± 0.1740
		MUOT	0.4368 ± 0.1296	0.5068 ± 0.1740	0.4344 ± 0.1367
IR = 0.2	2	Original	0.7699 ± 0.0949	0.6147 ± 0.0877	0.6608 ± 0.0871
		MUOT	0.6656 ± 0.0917	0.7139 ± 0.0700	0.6675 ± 0.0734
	3	Original	0.6577 ± 0.1086	0.4536 ± 0.0979	0.5046 ± 0.0948
		MUOT	0.5653 ± 0.0975	0.6003 ± 0.0655	0.5562 ± 0.0728
	4	Original	0.6780 ± 0.1435	0.5069 ± 0.1356	0.5485 ± 0.1371
		MUOT	0.5685 ± 0.1200	0.6265 ± 0.1224	0.5743 ± 0.1187
IR = 0.3	2	Original	0.8023 ± 0.0693	0.7254 ± 0.0721	0.7461 ± 0.0670
		MUOT	0.7194 ± 0.0671	0.7942 ± 0.0447	0.7450 ± 0.0556
	3	Original	0.7441 ± 0.0889	0.5938 ± 0.1072	0.6401 ± 0.1024
		MUOT	0.6537 ± 0.0806	0.6963 ± 0.0734	0.6614 ± 0.0761
	4	Original	0.7366 ± 0.0995	0.6277 ± 0.1322	0.6606 ± 0.1191
		MUOT	0.6618 ± 0.0806	0.7052 ± 0.1014	0.6730 ± 0.0905
UCI collection		Original	0.5750 ± 0.2381	0.4774 ± 0.2453	0.4843 ± 0.2559
		MUOT	0.5646 ± 0.1903	0.6087 ± 0.1939	0.5551 ± 0.2054

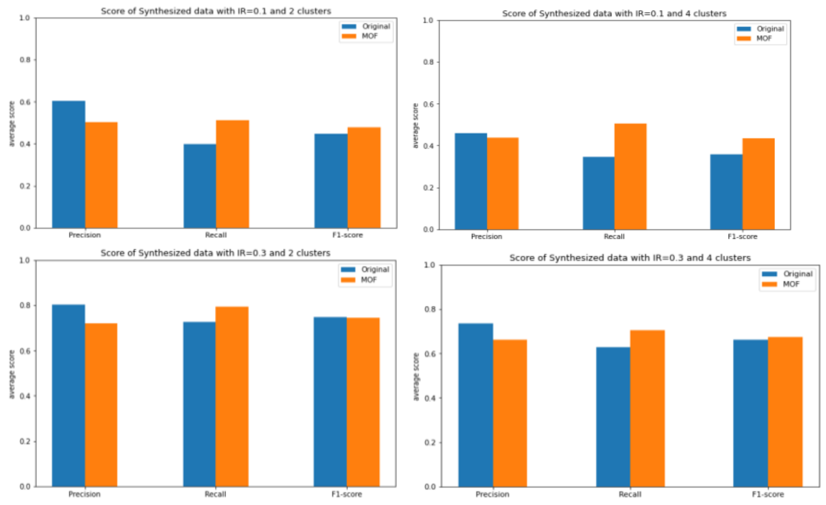

Figure 3: Average precision, recall and F1-score of 4 collection

4.2 *Real world datasets*

Five UCI datasets are used in the experiment. In contrast with the synthesized data, the minority class must be selected in the experiment. The brief description of five UCI datasets and their characteristics are shown in Table 3. In real world datasets, consisting of both binary and multiclass datasets. The multiclass datasets are converted to the binary datasets by selecting one class as the minority class and the rest as the majority class. In Table 3, the target class was chosen to be the minority class as in the column of "minority target" and the rest were set as the majority class. The overall IR values of these five datasets is 0.2467. The average performances from the UCI datasets are shown in Figure 4.

Table 3. Information of UCI datasets used in the experiment

Datasets	#instances	#features	minority target	#minority	IR
Wine	178	13	"3"	48	0.2697
Parkinsons	195	22	"0"	48	0.2461
Haberman	306	3	"2"	81	0.2647
E Coli	336	7	"imU"	35	0.1041
Pima	768	8	"1"	268	0.3489
Average					0.2467

From the previous observation, the MUOT algorithm can help classifiers to gain better recall and F1-score for IR less than 0.3 so it is expected to see the improved performance for these five UCI datasets having the average IR as 0.2467. From Figure 4, the improved performance of recall and F1-score are obtained. This can be concluded that the MUOT algorithm can help classifiers improve recall and F1-score.

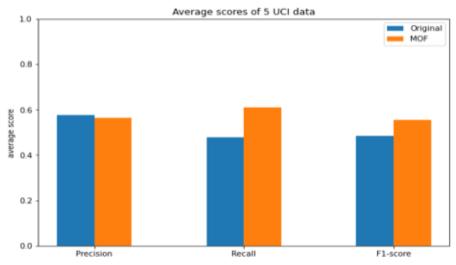

Figure 4: Average precision, recall and F1-score of 5 UCI dataset

4.3 Results

The MUOT algorithm generates a new dataset that increases recall for a classifier since more positive instances can be easily recognized, but it will decrease precision due to the enlarged minority regions so F1-score is the preferred measure which incorporates recall and precision together. All results show the increases of recall and F1-score and the decrease of precision. From three collections varying by IR values, the ranges of recall and F1-score increase corresponding to IR, the more balanced the datasets are, the less improvement MUOT will be. For the result of different numbers of clusters, MUOT exhibits the highest F1-score when the number of clusters is equal to 4. This may come from the spread of minority instances among all clusters. So MOF can be used to help detect abnormal instances very well when more minority instances are spread across the dataset. For UCI datasets, MUOT improves recall and F1-score and decreases precision. To validate these findings, the non-parametric Wilcoxon signed-rank tests are used to find the statistical significance of F1-score between the original datasets and the datasets from the MUOT algorithm with respect to four classifiers.

4.4 The Wilcoxon signed-rank test

The Wilcoxon signed-rank tests are used to evaluate the statistical significant improvement of datasets from the MUOT algorithm against the original datasets. In each test, the original datasets will be evaluated by 4 standard classifiers compared with the datasets generated from the MUOT algorithm based on the same classifier. Table 4 showed the p-values from the Wilcoxon signed-rank test if the p-value is less than 0.05, then it is considered to be significantly different.

Table 4. Wilcoxon signed-rank test, p-values of each dataset

Collection	Subcollection	P-value		
IR	#cluster	precision	recall	F1-score
IR = 0.1	2	0.004181	3.05E-05	0.015503
	3	3.35E-02	6.10E-05	0.000214
	4	4.04E-01	3.05E-05	0.000305
IR = 0.2	2	6.10E-05	3.05E-05	4.04E-01
	3	4.27E-04	3.05E-05	9.16E-05
	4	6.10E-05	3.05E-05	6.29E-03
IR = 0.3	2	3.05E-05	3.05E-05	0.175354
	3	9.16E-05	3.05E-05	0.028992
	4	3.05E-05	3.05E-05	0.433197
UCI collection		4.75E-01	3.62E-05	7.30E-03

For collection 3 with IR = 0.3 and the number of clusters = 2 and 4, it shows no significantly different performances of F1-score since their p-values are higher than 0.05. While all other collections, the p-values are smaller than 0.05 so the significant improvement can be obtained using MUOT. It can be concluded that MUOT is more effective on imbalance datasets having IR value less than or equal 0.2 from the datasets.

5 Conclusion

This paper proposed the undersampling and oversampling techniques using MOF for a class imbalance problem, called MUOT (Mass ratio variance majority undersampling and minority oversampling technique). It selects abnormal instances using MOF from the parameter-free outlier scoring method with the 90 percentile threshold. The undersampling step performs on majority instances having MOF score above 90 percentile and the oversampling step performs on abnormal minority instances using the same threshold setting inside the ball that does not include any majority instance according to MOF scores. This enlarges the minority regions and aids classifiers to recognize more minority instances.

The experiments with different IR values shows the improved performance of MUOT for recall and F1-score. In addition, datasets having IR less than 0.3 will be improved using MUOT. So that MUOT can handle imbalance datasets very effectively when a dataset is imbalanced. More characteristics of datasets should be investigated further via MUOT and the threshold values for undersampling and oversampling should be investigated.

References

[1] Ali A, Ralescu A, Shamsuddin SM. Classification with class imbalance problem: A review. Int. J. Advance Soft Compu. 2015 Jan. vol. 5, no. 3, p.176-204.

[2] Aweyemi JO, Adetunmbi AO, Oluwadare SA. Credit card fraud detection using machine learning techniques: A comparative analysis. International Conference on Computing Networking and Informatics (ICCNI); 2017 Oct 29-31. p. 1-9.
[3] Kononenko I. Machine learning for medical diagnosis: history, state of the art and perspective. Artificial Intelligence in Medicine. 2001 Aug. vol. 23, no. 1, p. 89-109.
[4] Mozina M, Zabkar J, Bratko I. Argument based machine learning. Artificial Intelligence. 2007 Jul-Oct. vol. 171, no. 10–15, p. 922-937.
[5] Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research. 2002. vol. 16, p. 321–357.
[6] Han H, Wang WY, Mao BH. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In: Huang DS, Zhang XP, Huang GB, editors. Advances in Intelligent Computing. ICIC; 2005. Lecture Notes in Computer Science; Berlin, Heidelberg, Springer. vol. 3644, p. 878-887.
[7] Bunkhumpornpat C, Sinapiromsaran K, Lursinsap C. Safe-Level-SMOTE: Safe-Level-Synthetic Minority Over-Sampling TEchnique for Handling the Class Imbalanced Problem. In Theeramunkong T, Kijsirikul B, Cercone N, Ho TB, editors. Advances in Knowledge Discovery and Data Mining. PAKDD; 2009. Lecture Notes in Computer Science; Berlin, Heidelberg, Springer. vol. 5476, p. 475-482.
[8] Rout N, Mishra D, Mallick MK. Handling Imbalanced Data: A Survey. In: Reddy M, Viswanath K, K.M. S, editors. International Proceedings on Advances in Soft Computing, Intelligent Systems and Applications; 2017 Dec 28. Advances in Intelligent Systems and Computing; Singapore, Springer. vol 628, p. 431-443.
[9] Gosain A, Sardana S. Handling class imbalance problem using oversampling techniques: A review, International Conference on Advances in Computing, Communications and Informatics; 2017 Sep 13-16. p. 79-85.
[10] Bunkhumpornpat C, Sinapiromsaran K, Lursinsap C. DBSMOTE: Density-Based Synthetic Minority Over-sampling TEchnique. Appl Intell 36. 2012 Apr. p. 664–684.
[11] Chiamanusorn C, Sinapiromsaran K. Extreme Anomalous Oversampling Technique for Class Imbalance. In Proceedings of the 2017 International Conference on Information Technology (ICIT 2017); 2017 Dec 27. Association for Computing Machinery, New York, NY, USA. p. 341–345.
[12] Liu X, Wu J, Zhou Z. Exploratory Undersampling for Class-Imbalance Learning. In IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics); 2009 Apr. vol. 39, no. 2, p. 539-550.
[13] Bunkhumpornpat C, Sinapiromsaran K. DBMUTE: density-based majority under-sampling technique. Knowl Inf Syst 50. 2017 Mar. p. 827–850.
[14] Junsomboon N, Phienthrakul T. Combining Over-Sampling and Under-Sampling Techniques for Imbalance Dataset. In Proceedings of the 9th International Conference on Machine Learning and Computing; 2017 Feb 24. USA, New York, Association for Computing Machinery. p. 243–247.
[15] Sahin Y, Bulkan S, Duman E. A cost-sensitive decision tree approach for fraud detection. Expert Systems with Applications. 2013 Nov 1. vol. 40, no. 15, p. 5916-5923.
[16] Boonchuay K, Sinapiromsaran K, Lursinsap C. Decision tree induction based on minority entropy for the class imbalance problem. Pattern Anal Applic 20. 2017 Aug. p. 769–782.
[17] Sagoolmuang A, Sinapiromsaran K. Oblique Decision Tree Algorithm with Minority Condensation for Class Imbalanced Problem. Engineering Journal. 2020 Feb 8. vol. 24, no. 1, p. 221-237.
[18] Sagoolmuang A, Sinapiromsaran K. Decision Tree Algorithm with Class Overlapping-Balancing Entropy for Class Imbalanced Problem. International Journal of Machine Learning and Computing. 2020 May 3. vol. 10, no. 3, p. 444-451.
[19] Suebkul K, Sinapiromsaran K. Recursive Tube-Partitioning Algorithm for a Class Imbalance Problem. Thai Journal of Mathematics. 2020. vol. 18, no. 4, p.2041-2051.
[20] Thanathamathee P, Lursinsap C. Handling imbalanced data sets with synthetic boundary data generation using bootstrap re-sampling and AdaBoost techniques. Pattern Recognition Letters. 2013 Sep 1. vol. 34, no. 12, p. 1339-1347.
[21] Sun Y, Kamel MS, Wong AKC, Wang Y. Cost-sensitive boosting for classification of imbalanced data. Pattern Recognition. 2007 Dec. vol. 40, no. 12, p. 3358-3378.
[22] Breiman L. Bagging Predictors. Machine Learning 24. 1996 Aug. p. 123-140.
[23] Devi D, Biswas S, Purkayastha B. Redundancy-driven modified Tomek-link based undersampling: A solution to class imbalance. Pattern Recognition Letters. 2017 Jul 1. vol. 93, p. 3-12.
[24] Changsakul P, Boonsiri S, Sinapiromsaran K, Mass-ratio-variance based Outlier Factor. 18th International Joint Conference on Computer Science and Software Engineering (JCSSE); 2021 Jul, p. 1-5.