CUSBoost: Cluster-based Under-sampling with Boosting for Imbalanced Classification

Farshid Rayhan, Sajid Ahmed, Asif Mahbub, Md. Rafsan Jani,
Swakkhar Shatabda, and Dewan Md. Farid
Department of Computer Science & Engineering, United International University, Bangladesh
Email: dewanfarid@cse.uiu.ac.bd

***Abstract*—Class imbalance classification is a demanding research problem in the context of machine learning and its applications, as most of the real-life datasets are often imbalanced in nature. Existing learning algorithms maximise the classification accuracy by correctly classifying the majority class, but misclassify the minority class. The challenge lies in that the minority class instances represent the data of greater interest than the majority class instances in real-life applications. Recently, several techniques based on sampling methods (under-sampling and over-sampling of the majority and minority class respectively), cost-sensitive learning methods, and ensemble learning have been used in the literature for classifying imbalanced datasets. In this paper, we present a new clustering-based under-sampling approach with boosting (AdaBoost) algorithm, called CUSBoost, for effective imbalanced classification. The proposed algorithm provides an alternative to RUSBoost (random under-sampling with AdaBoost) and SMOTEBoost (synthetic minority over-sampling with AdaBoost) algorithms. We evaluated the performance of the CUSBoost algorithm against the state-of-the-art methods based on ensemble learning like AdaBoost, RUSBoost, SMOTEBoost on 13 imbalanced binary and multi-class datasets with various imbalance ratios. The experimental results show that the CUSBoost is a promising and effective approach for dealing with highly imbalanced datasets.**

***Keywords*—Boosting; Class imbalance; Clustering; Ensemble classifier; Sampling; RUSBoost**

I. Introduction

In machine learning (ML) for data mining (DM) applications, supervised learning (or classification) is the method of identifying novel instances employing classifiers (or classification algorithms) based on a set of data with known class labels (training data) [1], [2], [3], [4]. Often real-world data sets are multi-class, high-dimensional and class-imbalanced, which fall-off the classification accuracy of many ML algorithms. So, a number of ensemble classifiers with sampling techniques have been proposed for classifying binary-class low-dimensional imbalanced data in the last decade [5], [6], [7]. Ensemble classifiers use multiple ML algorithms to enhance the performance of individual classifiers and combine multiple hypotheses to form an advance hypothesis [3]. The sampling methods use under-sampling (under-sampling of the majority class instances) and over-sampling (over-sampling the minority class instances) techniques to generate a more balanced class distribution. The random under-sampling technique suffers from the loss of potentially useful training instances. Furthermore, over-sampling with replacement proved to be ineffective in increasing the minority class recognition as it increases the likelihood of overfitting. [8].

In real-world class imbalance data sets, the majority class instances dominate in quantity in comparison to the minority class instances. However, the minority class instances are representing the information of greater concern as compared to the majority class instances [9]. The traditional ML for DM algorithms, such as decision tree (DT) [1], [3], k-nearest neighbors (kNN) [1], and naïve Bayes (NB) classifier [2], build the classification models that maximise the classification rate but ignore the minority class. The most approved practices for resolving class imbalance problems are sampling techniques, ensemble methods, and cost-sensitive learning methods. Sampling techniques (under-sampling and over-sampling) either remove the majority class instances from the imbalanced data or add the minority class instances into the imbalanced data to get the balanced data. Ensemble methods such as Bagging, Boosting and their variations are also widely used for classifying imbalanced data. Basically, the ensemble methods use sampling technique in each iteration. The cost-sensitive learning is also applied for solving the class imbalance problems, which assigns different costs in relation to the misclassification error of classes. Typically high cost is assigned for the minority class instances and vice versa. However, the classification results are not stable in cost-sensitive learning methods as it is very challenging to achieve a precise misclassification. Furthermore, different misclassification costs can potentially lead to different conclusions.

The methods for resolving class imbalance problems can be split into two groups: (a) external methods and (b) internal methods. The external methods are also known as data balancing methods, which preprocess the imbalanced data to get the balanced data. The internal methods rework existing learning algorithms in order to diminish their susceptibility to the class imbalance when learning from the imbalanced data. In this paper, we present a new clustering-based under-sampling approach with boosting (AdaBoost), called CUSBoost algorithm. We divide the imbalanced dataset into two part: majority class instances and minority class instances. Then, we cluster the majority class instances into several groups using k-means clustering algorithm and select the majority class instances from each cluster to form a balanced dataset. Clustering groups the instances in the dataspace according to their similarity and ensures that each cluster contains instances more analogous to each other than to those of other clusters. So, instead of randomly removing the majority class instances we used clustering technique to select the majority class instances. CUSBoost combines the sampling and boosting methods to form an efficient and effective algorithm for class imbalance learning. We tested the performance of CUSBoost algorithm

978-1-5386-2044-1/17/$31.00 ©2017 IEEE

with AdaBoost, RUSBoost, and SMOTEBoost algorithms on 13 imbalance datasets. Based on the experimental results, we can validate that combining clustering-based under-sampling approach with AdaBoost algorithm is a promising technique for improving the bias of class imbalance problems.

The remainder of this paper is organised as follows. Section II presents related works. Section III describes the data balancing methods, and Section 3 presents the CUSBoost algorithm. Section V provides the experimental results. Finally, we conclude in Section VI.

II. Related work

In the last decade, sampling methods, ensemble methods based on bagging and boosting, and cost-sensitive learning methods have been used to manage imbalanced binary classification problems. [10], [11]. Fig. 1 shows the process of classifying imbalanced data using sampling with boosting approach.

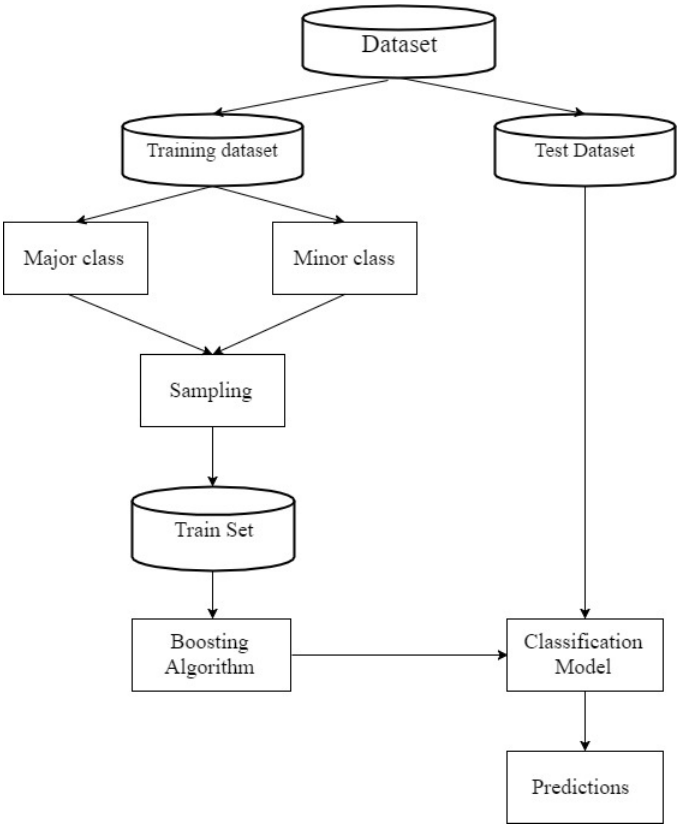


Fig. 1: Sampling with boosting for classifying imbalanced data.

Sun et al. [8] proposed an ensemble technique for dealing with binary-class imbalance problems by transforming an imbalanced binary learning process into multiple balanced learning processes. In this approach, instances of the majority class were divided into several groups/ sub-data sets, where each sub-set has a similar number of minority class instances. Thus, a number of balanced datasets were generated. After that, each balanced data set was utilized in creating a binary classifier. Finally, these binary classifiers were amalgamated in creating an ensemble classifier to categorize the new data.

Chawla et al. [12] presented an over-sampling technique called SMOTE (Synthetic Minority Over-sampling TEchnique) algorithm where over-sampling is conducted on the minority class through the creation of synthetic minority class-instances instead of over-sampling with replacement. SMOTE generated synthetic instances by operating in *feature space* rather than *data space* employing k minority class nearest neighbors. Their result showed that the combination of over-sampling with under-sampling performed better in Receiver Operating Characteristic (ROC) space. Santos et al. [13] implemented a cluster-based (k-means) over-sampling approach where SMOTE was adapted to oversample clusters with reduced sizes. This work considered merging the minority class instances from the multiple over-sampled datasets. Blagus and Lusa [14] investigated the behavior of SMOTE on high-dimensional imbalanced data, in which the aggregate of features overshadows the number of training instances. They found that feature selection is necessary for SMOTE with k-nearest neighbors (kNN), as SMOTE strongly biases the classifier towards the minority class.

Seiffert et al. [15] presented a novel hybrid sampling/boosting algorithm, called RUSBoost, which applied random under-sampling (RUS) with AdaBoost algorithm. RUS randomly removes the majority class instances to form a balanced data. RUSBoost was built based on the SMOTEBoost (synthetic minority over-sampling with AdaBoost) algorithm [16]. SMOTEBoost was built upon over-sampling approach with AdaBoost algorithm. Galar et al. [17] presented an ensemble algorithm by evolutionary under-sampling (EUA) approach, called EUSBoost, to classify highly imbalanced datasets. EUS generated several sub-datasets using randomly under-sampling technique and obtained a best under-sampled dataset of the original dataset that cannot be improved further. RUSBoost, SMOTEBoost, and EUSBoost applied data sampling techniques into the AdaBoost.M2 algorithm by considering the minority class instances and the majority class instances. Yen and Lee [18] presented a cluster-based under-sampling approach to cluster all the training instances (majority class instances and minority class instances) into some clusters. This approach selected an appropriate quantity of majority class instances from each cluster based on the ratio of majority class instances to the minority class instances in the respective cluster.

III. Data Balancing Methods

Researchers have previously proposed various ways of dealing with imbalanced datasets. The following section describes a few of those methods.

A. Sampling Techniques

1) Under-sampling: Under-sampling involves the removal of some majority class instances to result in a balanced distribution of all classes [9]. However, this can result in the removal of informative instances from the majority class especially if the number of instances of the minority class is very small (for example majority to minority ratio is 100:1). Thus, there will be a huge loss of information from the majority class

978-1-5386-2044-1/17/$31.00 ©2017 IEEE

instances, leading to a non-optimal classification. Fig. 2 shows the random under-sampling process to select the majority class instances randomly.

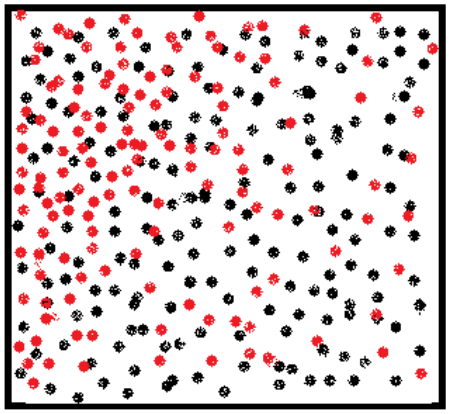


Fig. 2: Random under-sampling (RUS) approach. Here the red dots are selected instances from the majority class, where black and red dots are representing all the majority class instances.

2) Oversampling: In the case of oversampling, we increase the number of minority class instances and this can be done in different ways. The most common of these methods is random oversampling, where minority class instances are reproduced to ensure both class instances are balanced. The issue with this method, however, is that there is a high probability of over-fitting due to the same instances occurring multiple times. SMOTE (Synthetic Minority class Oversampling TechniquE) is another oversampling technique which handles this problem by creating synthetic instances of the minority class instead of duplicating the minority class instances [12]. This is done by interpolation of minority class instances which gather near each other. Nevertheless, this problem can be present to some extent because the newly generated samples could be almost identical to the existing minority class instances. Modified Synthetic Minority Over-sampling TechniquE (MSMOTE) is the variant of Synthetic Minority Over-sampling TechniquE (SMOTE) that additionally considers the distribution of minor class instances. It splits the data belonging to the minority class into three categories (border, safe & latent noise) based on their distance with all training instances.

Under-sampling method generally works better than over-sampling methods as long as the imbalance ratio of the dataset is not very high [9].

B. Cost-sensitive learning

This method tackles the class imbalance problem by assigning more misclassification cost to minority class instances for the underlying classifier. The misclassification cost can be used in the cost function and the cost function can be optimised by the classifier. The classifier will now treat both majority and minority classes equally due to their underlying costs. But in this case, the greatest challenge is in assigning the costs as they are difficult to be derived from datasets.

C. Ensemble Learning

Ensemble learning methods combine multiple base learners which may be the same or different [19], [20]. This usually increases the predictive capability of the individual base classifiers, thus making it adaptive to different datasets. Ensemble methods find their usage in a wide variety of problems.

1) Bagging: Bagging creates multiple sub-sets of the original dataset through sampling with replacement or without replacement. These sub-sets are used by the base learners while considering each instance with equal weight. The output of the individual base learners is considered as votes to determine the final prediction.

2) Boosting: Boosting is similar to bagging in that it combines multiple base learners to obtain a result based on voting technique. However, it's differences lie in that boosting assigns weight to instances according to how hard they are to classify, thus setting a high weight of instances that are difficult to classify. One base learner contributes to the weight used by the next base learner. Weights are assigned to the base learners as well according to their predictive accuracy and this is taken into consideration when classifying a new test instance. Although boosting was not created specifically for imbalanced dataset problems, it's characteristic of assigning weights to examples that are relatively harder to classify makes it ideal for handling class imbalance problems. It can be somewhat referred to as a cost-sensitive method.

IV. CUSBoost Algorithm

CUSBoost is based on the combination of cluster-based sampling and Adaboost algorithm. It is similar to RUSBoost and SMOTEBoost with the critical difference occurring in the sampling technique. SMOTEBoost uses SMOTE method to oversample the minority class instances, while RUSBoost uses random under-sampling on the majority class. In comparison, our proposed CUSBoost uses cluster-based sampling from the majority class. CUSBoost separates the majority and minority class instances from the original dataset and clusters the majority class instances into k clusters using k-means clustering algorithm. Here, the parameter k is determined by hyper-parameter optimisation. After that, random under-sampling is conducted on each of the created clusters by randomly selecting 50% of the instances (but it can be tuned according to the domain problem or dataset) and removing the rest. As clustering is used before sampling, theoretically this algorithm will perform best when the dataset is highly cluster-able. These representative samples are joined with the unchanged minority class instances to achieve balanced datasets.Our algorithm's strength lies in the fact that it considers examples from all subspaces of the majority class since k-means clustering puts each instance in some cluster. Other similar methods often fail in obtaining proper representatives of the majority class. Fig. 3 shows the proposed cluster-based under-sampling technique to select the majority class instances.

CUSBoost combines each vote of an ensemble of decision trees using C4.5 algorithm [21] to classify new instances. All instances are loaded with identical weights of $\frac{1}{d}$, where d represents the complete number of training instances. These weights are calibrated according to their classification result: weights are decreased when correctly classified and increased

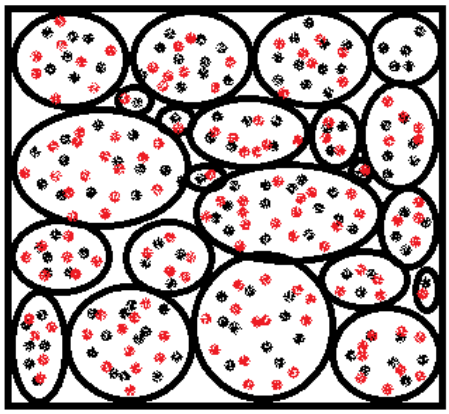


Fig. 3: Cluster-based under-sampling (CUS) approach. Here the red dots are selected instances from the majority class, where black and red dots are representing all the majority class instances.

when misclassified. Hence the weight of an instance denotes its difficulty to be classified correctly. The *error rate* of model M_i, is found by aggregating the weights of misclassified instances in D_i as shown in Eq. 1. Here, $err(x_i)$ is a binary variable where it is true when instance x_i is misclassified and false when x_i is correctly classified. CUSBoost algorithm is summarised in Algorithm 1.

$$error(M_i) = \sum_{i=1}^{d} w_i * err(x_i) \tag{1}$$

V. Experimental Results

In this section, we present the experimental analysis to examine the performance of our proposed CUSBoost algorithm. We have used datasets from KEEL-dataset repository [22] with different imbalance ratio. Table I show the datasets details .

TABLE I: Imbalanced datasets description.

No.	Datasets	Instances	Features	Class Values	Imbalance Ratio
1	pima	768	8	2	1.87
2	dermatology	366	34	6	5.55
3	segment0	2308	19	2	6.02
4	led7digit	443	7	2	10.97
5	abalone9-18	731	8	2	16.4
6	yeast	1484	8	10	23.15
7	$poker - 9_v s_7$	244	10	2	29.5
8	kddcup-guess passwd_vs_satan	1642	41	2	29.98
9	yeast5	1484	8	2	38.73
10	ecoli	336	7	8	71.5
11	abalone19	4174	8	2	129.44
12	Page Blocks	548	10	5	164
13	Statlog (Shuttle)	2175	9	7	853

Algorithm 1 CUSBoost Algorithm

Input: Imbalanced data, D, No. of iterations, k, & C4.5 decision tree induction algorithm.
Output: An ensemble classifier.
Method:

1: assign initial weight to each instance, $x_i \in D$ to $\frac{1}{d}$;
2: **for** i = 1 to k **do**
3: create balanced dataset D_i with distribution D using cluster-based under-sampling;
4: derive a tree, M_i from D_i employing C4.5 algorithm;
5: compute the error rate of M_i, $error(M_i)$;
6: **if** $error(M_i) \geq 0.5$ **then**
7: return to step 3;
8: **end if**
9: **for** each $x_i \in D_i$ that correctly classified **do**
10: multiply the weight of x_i by $(\frac{error(M_i)}{1-error(M_i)})$; // update weights
11: **end for**
12: normalise the weight of each instances, x_i;
13: **end for**

To use the ensemble to classify instance, x_{New}:

1: initialise each class weight to zero;
2: **for** i = 1 to k **do**
3: $w_i = log\frac{1-error(M_i)}{error(M_i)}$;
4: $c = M_i(x_{New})$;
5: add w_i to weight for class c;
6: **end for**
7: return the maximum weighted class;

A. Evaluation Methods

ROC curve is a graph that represents the true positive rate (tpr) along the y-axis and false positive rate (fpr) along the x-axis. The perfect classifier would have an area under the curve (auroc) of 1, where tpr = 1 and fpr = 0. At different cut - off points the curve shows the tpr and fpr of the classifier at different thresholds.

$$TPRate = \frac{TP}{TP + FN} \tag{2}$$

$$FPRate = \frac{FP}{FP + TN} \tag{3}$$

In the case of imbalanced classification problems, emphasis is placed particularly on the predictive accuracy of minority/positive class while retaining accuracy for the majority/negative class. This would correspond to high tpr and low fpr, and thus reflected by a high auroc. Hence we are using auroc to compare our proposed method with other state-of-the-art techniques.

B. Results

In this experiment, we have compared the proposed CUSBoost method with AdaBoost, RUSboost, and SMOTEBoost methods. Each dataset was validated using Area Under the ROC Curve (AUC). As for the base learner, we used C4.5 decision tree induction in boosting. Keel-dataset repository's

978-1-5386-2044-1/17/$31.00 ©2017 IEEE

implementation was used for the AdaBoost, RUSBoost, and SMOTEBoost algorithms [22]. Each of these experiments are done 5 times with 10 fold cross validation and their mean scores are shown in Table II.

TABLE II: Average performance of the AdaBoost, RUSboost, SMOTEBoost, and CUSBoost methods on 13 imbalanced datasets.

Datasets	AdaBoost	RUSBoost	SOMTEBoost	CUSBoost
pima	0.6223	0.6376	0.6597	**0.6679**
dermatology	**0.9342**	0.43	0.632	0.54
segment0	**0.996**	0.957	0.951	0.943
led7digit	0.8937	0.8279	0.9373	**0.9412**
abalone9-18	0.6934	0.7051	0.7195	**0.7243**
yeast	0.7589	0.7382	0.741	**0.7603**
$poker - 9_v s_7$	0.642	0.589	0.6708	**0.967**
kddcup-guess passwd_vs_satan	0.8324	0.8681	0.8513	**0.8745**
yeast5	0.9231	0.887	**0.9253**	0.9
ecoli	0.6354	0.517	**0.6597**	0.6589
abalone19	0.5723	0.5923	0.5583	**0.609**
Page Blocks	0.7605	0.5975	0.8123	**0.8981**
Statlog (Shuttle)	0.8331	0.727	0.8445	**0.8847**

From the results we can see that the CUSBoost algorithm performs best most of the time when the imbalance ratio is within a specific range. As the imbalance ratio gets high CUSBoost starts to outperform all the other methods significantly. This happens because our method does not focus on making the ratio of majority and minority class examples $1 : 1$. So for this reason the sub-sampled training dataset holds a better representation of the majority class while remaining imbalance itself. RUSBoost shows a performance with high variance especially with highly imbalanced datasets, thus it shows a poor performance if the mean results are chosen. But, if the best results are chosen from 10 experiments then RUSBoost outperforms the other methods including the proposed method on the majority of the datasets as shown in Table III. The best result of proposed method is usually quite close to the average result thus proving low variance in its performance .

TABLE III: The best result using AdaBoost, RUSboost, SMOTEBoost, and CUSBoost methods in each dataset is stressed in bold-face.

Datasets	AdaBoost	RUSBoost	SOMTEBoost	CUSBoost
pima	0.6653	**0.6796**	0.667	0.6679
dermatology	**0.9**	0.6935	0.632	0.54
segment0	**0.998**	0.9805	0.9642	0.9523
led7digit	0.9157	0.886	0.9485	**0.9544**
abalone9-18	0.707	**0.805**	0.7029	0.7434
yeast	0.7839	**0.801**	0.7653	0.7895
$poker - 9_v s_7$	0.657	0.7395	0.69	**0.9685**
kddcup-guess passwd_vs_satan	0.8578	**0.9123**	0.8602	0.8849
yeast5	0.9324	**0.9515**	0.9391	0.9126
ecoli	0.6455	**0.677**	0.6753	0.6656
abalone19	0.5796	**0.6498**	0.5876	0.6176
Page Blocks	0.78	**0.937**	0.8592	0.9007
Statlog (Shuttle)	0.8567	**0.8989**	0.8516	0.8967

VI. Conclusion

Existing classification algorithms generally focus on majority class instances and ignore the minority class instances. So, it is a challenging task to construct an effective classifier that can correctly classify the instances of the minority class. This is ever so pertinent as class imbalance problems affect a vast range of domains. Recently, computational intelligence researchers have proposed several hybrid techniques by combining sampling with ensemble classifiers for dealing with class imbalance problems. This paper sought to present a new algorithm called CUSBoost, or Cluster-based Under-sampling with Boosting, in order to alleviate the problem of class imbalance. We have compared the performance of CUSBoost algorithm with the most effective boosting techniques like AdaBoost, RUSBoost, and SMOTEBoost algorithms. From our experimental results, we have found that CUSBoost performed favourably when compared to these popular techniques on datasets having high class imbalance ratio.

Rather than simply randomly choosing samples from the dataset, CUSBoost first clusters the majority class instances and then performs random under-sampling so that the boosting algorithm (Adaboost) can use examples from all regions of the dataset. Thus an advantage it holds over RUSBoost is that the variance of the results is low, which leads to stable performance. Additionally, the performance of this algorithm has been shown to be relatively proportional to the imbalance ratio of the dataset, where higher imbalanced ratio produces a greater performance. It has shown to outperform the other algorithms in a certain range of imbalanced ratio. In this paper, we have compared the performance of CUSBooost with RUSBoost, SMOTEBoost and Adaboost. In future work, we intend to conduct further experiments to continue investigating the performance of CUSBoost with other established sampling methods and ensemble methods.

References

[1] D. M. Farid, M. A. Al-Mamun, B. Manderick, and A. Nowe, "An adaptive rule-based classifier for mining big biological data," *Expert Systems with Applications*, vol. 64, pp. 305–316, December 2016.

[2] D. M. Farid, L. Zhang, C. M. Rahman, M. Hossain, and R. Strachan, "Hybrid decision tree and naïve bayes classifiers for multi-class classification tasks," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1937–1946, March 2014.

[3] D. M. Farid, L. Zhang, A. Hossain, C. M. Rahman, R. Strachan, G. Sexton, and K. Dahal, "An adaptive ensemble classifier for mining concept drifting data streams," *Expert Systems with Applications*, vol. 40, no. 15, pp. 5895–5906, November 2013.

[4] D. M. Farid and C. M. Rahman, "Assigning weights to training instances increases classification accuracy," *International Journal of Data Mining & Knowledge Management Process*, vol. 3, no. 1, pp. 13–25, January 2013.

[5] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, June 2009.

[6] Y. Sun, A. K. C. Wong, and M. S. Kamel, "Classification of imbalance data: A review," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 4, pp. 687–719, June 2009.

[7] M. Wasikowski and X. wen Chen, "Combating the small sample class imbalance problem using feature selection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1388–1400, October 2009.

[8] Z. Sun, Q. Song, X. Zhu, H. Sun, B. Xu, and Y. Zhou, "A novel ensemble method for classifying imbalanced data," *Pattern Recognition*, vol. 48, no. 5, pp. 1623–1637, May 2015.

[9] D. M. Farid, A. Nowé, and B. Manderick, "A new data balancing method for classifying multi-class imbalanced genomic data," *25th Belgian-Dutch Conference on Machine Learning (Benelearn)*, pp. 1–2, 12-13 September 2016.

978-1-5386-2044-1/17/$31.00 ©2017 IEEE

[10] C. Beyan and R. Fisher, "Classifying imbalanced data sets using similarity based hierarchical decomposition," *Pattern Recognition*, vol. 48, no. 5, pp. 1653–1672, May 2015.

[11] A. D. Pozzolo, O. Caelen, and G. Bontempi, "When is undersampling effective in unbalanced classification tasks?" in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2015, pp. 200–215.

[12] 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, June 2002.

[13] M. S. Santos, P. H. Abreu, P. J. García-Laencina, A. Simão, and A. Carvalho, "A new cluster-based oversampling method for improving survival prediction of hepatocellular carcinoma patients," *Journal of Biomedical Informatics*, vol. 58, pp. 49–59, September 2015.

[14] R. Blagus and L. Lusa, "SMOTE for high-dimensional class-imbalanced data," *BMC Bioinformatics*, vol. 14, no. 106, pp. 1–16, March 2013.

[15] C. Seiffert, T. M. Khoshgoftaar, J. V. Hulse, and A. Napolitano, "Rusboost: A hybrid approach to alleviating class imbalance," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 40, no. 1, pp. 185–197, January 2010.

[16] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: Improving prediction of the minority class in boosting," *7th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pp. 107–109, 22-26 September 2003.

[17] M. Galar, A. Fernández, E. Barrenechea, and F. Herrera, "Eusboost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling," *Pattern Recognition*, vol. 46, no. 12, pp. 3460–3471, December 2013.

[18] S.-J. Yen and Y.-S. Lee, "Cluster-based under-sampling approaches for imbalanced data distributions," *Expert Systems with Applications*, vol. 36, no. 3 (Part 1), pp. 5718–5727, April 2009.

[19] J. R. Quinlan *et al.*, "Bagging, boosting, and c4. 5," in *AAAI/IAAI, Vol. 1*, 1996, pp. 725–730.

[20] R. Maclin and D. Opitz, "An empirical evaluation of bagging and boosting," *AAAI/IAAI*, vol. 1997, pp. 546–551, 1997.

[21] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[22] J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 17, no. 2-3, pp. 255–287, 2011.

978-1-5386-2044-1/17/$31.00 ©2017 IEEE