SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM
TMLAI TRANSACTIONS ON MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE
Volume 8 No 2
ISSN 2054-7390

COSM: Controlled Over-Sampling Method. A Methodological Proposal to Overcome the Class Imbalance Problem in Data Mining

Gaetano Zazzaro
CIRA, Italian Aerospace Research Centre, Capua (CE), Italy;
g.zazzaro@cira.it

ABSTRACT

The class imbalance problem is widespread in Data Mining and it can reduce the general performance of a classification model. Many techniques have been proposed in order to overcome it, thanks to which a model able to handling rare events can be trained. The methodology presented in this paper, called Controlled Over-Sampling Method (COSM), includes a controller model able to reject new synthetic elements for which there is no certainty of belonging to the minority class. It combines the common Machine Learning method for holdout with an oversampling algorithm, for example the classic SMOTE algorithm. The proposal explained and designed here represents a guideline for the application of oversampling algorithms, but also a brief overview on techniques for overcoming the problem of the unbalanced class in Data Mining.

Keywords: Class imbalance problem; Data Mining; Holdout Method; Oversampling; Rare Class Mining; Undersampling.

1 Introduction

In many real application fields, the discovery and modeling of rare events is crucial for understanding complex phenomena [1]. For example, rare weather conditions, if not forecasted, can be very dangerous for the population, housing, air traffic, and so on; unauthorized and fraudulent use of a credit card must be detected as soon as possible; an unidentified cyberattack is very dangerous for companies, causing huge economic losses. Sometimes such events are so diluted in the database that the Data Mining algorithms used for training analytical models fail to characterize them: such events are exchanged as noise [2]; if these events constitute a class value (+), the trained model could always give the same answer (-), ignoring the minority class. The main problem with class imbalance states is that standard models are often biased towards the majority class.

In Data Mining this condition is called class imbalance problem and it occurs when one of the two classes (in the binary case) has many more samples than the other class. What "many more" means is not clearly quantifiable and depends on the case. Most of the time, being able to train a model capable of classifying rare events, in conditions of high class imbalance, is an impossible goal, unless ad hoc strategies are first adopted. This problem is one of the main problem that degrades the performance of classification models [3] [4]. Various techniques have been proposed in order to solve the problem of class imbalance, including

DOI: 10.14738/tmlai.82.7925
Publication Date: 02nd May, 2020
URL: http://dx.doi.org/10.14738/tmlai.82.7925

over-sampling of the minority class or under-sampling of the majority one. Another widely used approach is focused on the cost-sensitive learning techniques included in meta-learning approaches [5]. These techniques take the misclassification cost in its account by assigning higher cost of misclassifications to the minority class, penalizing the correct classifications to the majority class, and generating the model with lowest cost.

In this paper, a design of a methodology for oversampling is proposed. By using an oversampling technique, the minority class is oversampled by taking each minority class sample and adding new synthetic records by applying various strategies that are deepened and compared. The method is called Controlled Over-sampling Methodology (COSM) because a classification model – the controller – is created that can check if the new synthetic examples really belong to the minority class. The controller assists the entire sampling procedure, eventually rejecting the misclassified examples.

Various aspects of the proposed method are also considered, including its relationship with the holdout method.

2 Holdout Method

The holdout method [6] is a very common strategy in Data Mining, mainly aimed at providing a useful scheme for datasets split and design, in order to train a model and evaluate its performances.

2.1 Basic Holdout

The whole dataset is randomly partitioned into two disjoint sets, called training and test sets. It is common to hold out one-third of data for testing and use the remaining two-third for training, but other proportions are possible, depending on the amount of data and other factors discussed later in the paper.

This simple subdivision does not take into consideration the distribution of the target class. In spite of the random partitioning, the two subsets could have very different distributions of the target class. In order to overcome this unlikely event, the training and test sets must not only be obtained randomly but they must be also stratified, so that the class distribution of the records in each set is approximately the same as that in the initial dataset.

This subdivision scheme can be enriched by considering a further subset of the training set, called validation set. More in detail, the validation set is used to select the algorithm parameters and then choose the model with the best performance metrics. This step is essential to mitigate the overfitting problem [7] [8]. Also in this case, the subdivision is random, can be stratified, and the subdivision percentages can vary, or be the same as the first splitting.

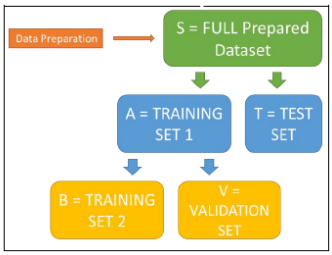

Figure 1. The general holdout method schema

Figure. 1 shows the framework of the complete holdout method. To recap:

- S is the dataset with all the records; it can be subpart of a much larger database; this set has undergone all the preparation steps: for example, possible selection, cleaning and normalization of data, selection and extraction of features if useful, and any other operations on the data.
- A is the intermediate Training set, starting from which the final training is obtained.
- B is the final Training set; a model, for example a classifier, is built on this set by applying mainly a statistical or a Machine Learning algorithm.
- V is the Validation set; it is useful to tune and select the parameters (or hyper-parameters) of the algorithm chosen for training the model. In other words, V is used to compare the performances of all the trained models and decide which one to take.
- T is the Test set; the model is then tested on this set; T is used to obtain the performance metrics, such as accuracy, sensitivity, specificity, AUC, F-measure, and so on; moreover, T is useful to understand if the model is overfitted. Generally, $T \cup A = S$.

The holdout method is not recommended when working with small datasets. In these cases, some variations may be applied to avoid that the dataset subdivisions can further reduce the number of data for the training and test phases.

2.2 Some Variations on the Theme: k-Fold Cross-Validation

The holdout method is the simplest kind of cross-validation method; the latter represents a more general method. In this approach, also called k-fold cross-validation, the original dataset is randomly partitioned into k (generally $k=10$) equal sized subsamples. For each of k experiments, $k-1$ folds are used for the training phase and the remaining one for testing phase. This procedure is repeated k times. The error estimates are averaged to yield an overall error estimate, as well as the other performance metrics. k-fold cross-validation seems to give better approximations of generalization than the holdout method [6] [8].

In some uses of the method described, such as when the multi-division of holdout reduces the number of records in the training set too much, a mixed approach between holdout and k-fold cross-validation can be applied. In few words, the training set is not further subdivided into validation (V), but the model is trained on Training set (A) with the k-fold cross-validation method: we can say that the model is cross-validated. Finally, as usual, it is tested on the test set (T) in order to calculate the performances of model.

3 Class Imbalance Problem

The class imbalance problem consists in a skewed distribution of instances that belong to different classes; because class distribution plays a key role in Data Mining and Machine Learning classification task, this problem can compromise the training phase and the performance of the classification model.

3.1 Examples in Real Datasets

In many real world applications, datasets suffer the problem of the class imbalance. In these situations, discovering instances of rare class is “akin to finding a needle in a haystack”. Furthermore, a model able to describe the minority class tends to be highly specialized, and this condition is not desired because a good model is a model that is able to generalize, otherwise the model goes into overfitting. However, most Data Mining algorithms do not work very well with imbalanced datasets [8].

Briefly, a dataset is affected by the unbalanced class problem when one of the classes has many more samples than the other ones. The most of machine learning algorithms is more focusing on classification of samples belonging to the majority class while ignoring or misclassifying samples of the minority one.

For example, if the target class has only two values (binary class case) “0” and “1”, and if the distribution is 99.9% of “1” and 0.01% of ”0“, a classifier that always says ”1“ is a very accurate model, because it never exhibits a false ”0“ and has a very low percentage (precisely the 0.01%) of false “1”.

There exist many case studies that do not have a balanced data set. Some examples are:

- Discovery of fake news;
- Distinction among earthquakes, nuclear and non-nuclear explosions;
- Document selection and filtering;
- Forecast of extreme weather conditions;
- Recognition of fraudulent telephone calls.

3.2 Strategies to Handle Imbalanced Datasets

As mentioned above, imbalance datasets can degrade the performance of a model that has been trained by applying a data driven technique; the Machine Learning algorithms lead to misclassifying the minority class records or treated them as noise. Even if the evaluation metric is changed, it is hard for the model to be accurate on the minority class, or that the chosen metric has a good result.

Many techniques have been proposed in order to overcome the problem of learning models on an unbalanced class. They can be categorized into three main categories: re-sampling [8], cost-sensitive learning [9] [10], and ensemble-based methods [11]. Some of them are summarized in Table 1. Nevertheless, the choice of the strategy to be followed strictly depends on the data and on the learning algorithm, and there is no absolute advantageous technique.

Strategies for overcoming the problem of the unbalanced class can be natively incorporated into the learning algorithm for training a classifier.

Table1. Techniques for Imbalanced Problem

Category	Technique	Algorithm	Reference
Re-Sampling	Under-sampling	Random Undersampling	[8]
		Clustering-based	[12]
	Over-sampling	SMOTE	[13]
		ADASYN	[14]
		Clustering-based	[15]
Cost Sensitive Learning	Direct	ICET	[16]
		CSDTree	[17]
	Meta-Learning	MetaCost	[18]
		CSC	[19]
		CSnaiveBayes	[20]
		Empirical Thresholding	[21]
Ensemble-based	Bagging-based	SMOTEBaggings	[22]
		Asymetric Bagging	[23]
		Ensemble Variation	[24]
	Boosting-based	SMOTEBoost	[25]
		RUSBoost	[26]
	Hybrid	EasyEnsemble	[27]
		BalanceCascade	[27]

3.3 SMOTE and ADASYN

Synthetic Minority Over-sampling Technique (SMOTE) [13] is an oversampling method able to create new artificial examples of minority class based on the similarity among the existing elements. SMOTE is the most used algorithm for oversampling, and there are numerous variants of it [28] [29] [30].

Let x_i be a record belonging to the minority class, $\hat{x_i}$ one of the k-nearest neighbors of x_i, and δ_i a random number belonging to $[0,1]$. A new synthetic example of the minority class is calculated as: $x_{new}=x_i+(\hat{x_i}-x_i)\cdot\delta_i$

The new x belongs to the line between x_i and $\hat{x_i}$.

The main shortcoming of SMOTE is the problem of overgeneralization. SMOTE's algorithm does not regard to the majority class and, in the case of highly skewed class distributions, a harmful mixture of the classes is obtained.

However, SMOTE yields among the best results as far as re-sampling and modifying the probabilistic estimate techniques go [31].

Another very common oversampling algorithm is Adaptive Synthetic (ADASYN) sampling procedure [14]. Its key idea is, in few words, to automatically find the number of synthetic observations to be generated for each observation belonging to the rare class by using a density distribution function. The number of synthetic samples, generated for each observation of the minority class, is determined by the percentage of samples belonging to the majority class in its neighborhood. The steps of the ADASYN are:

- Calculate the ratio of minority to majority examples using $d=m_s m_l$, where m_s and m_l are the number of minority and majority class examples respectively. d is the Degree of Imbalance.
- Calculate the total number of synthetic minority data to generate, by using $G=\beta\cdot(m_l-m_s)$; G is the total number of minority class data to generate. β is the ratio of minority: majority data desired after ADASYN. $\beta=1$ means a perfectly balance between two classes after ADASYN.
- For each x_i of the minority samples, find its k-nearest neighbors and calculate the ratio $r_i=\Delta_i k$, where Δ_i is the number of majority class examples.
- Normalize the r_i values: $\hat{r_i}=r_i \Sigma r_i$, and $\Sigma\hat{r_i}=1$.
- Calculate the amount of new synthetic examples to generate in each neighborhood: $G_i=G\hat{r_i}$.
- Generate G_i new data for each neighborhood, taking x_i. Select in random manner another minority example x_{zi} within the same neighborhood. The new synthetic example can be calculated by using:

$$x_{new}=x_i+(x_{zi}-x_i)\cdot\delta$$

where δ is a random number belonging to $[0,1]$, x_i and x_{zi} are two minority examples within a same neighborhood.

4 COSM Framework

One of the main disadvantages of the methods for oversampling is that the synthetic examples may not have the minority label or that they could never occur in the real world.

COSM is a general framework for the application of oversampling algorithms. Its main strength is the controller model $\mathcal{C}$, trained using an undersampling technique $\mathcal{M}$ and a machine learning algorithm; furthermore, $\mathcal{C}$ is tested on an independent set, which has the same class distribution as the original dataset, and which is also used to test the final classifier $\mathcal{F}$.

4.1 General Description

Figure 2 shows the framework of COSM. COSM can be entirely employed, for example, by using the operators, filters and algorithms of the WEKA open source software [32] [33].

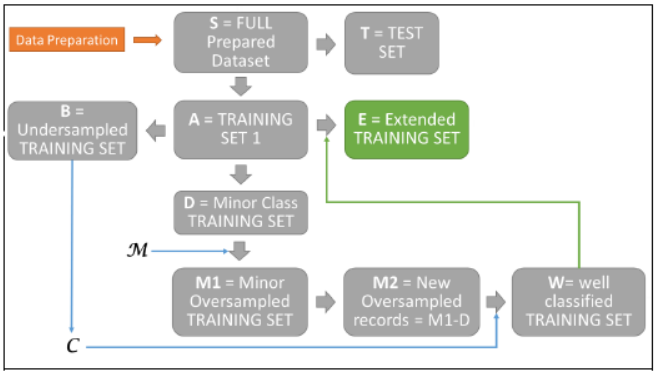

Figure 2. The complete schema of the subdivision of the dataset in COSM.

The Data Preparation step of CRISP-DM methodology [34] for the knowledge discovery in large database process covers all activities needed to build the final dataset from the raw data. After this phase, the full prepared and imbalanced dataset S is splitted into test set (T) and training set 1 (A), in accordance with the holdout method. T has the same distribution of the class target of S by applying a stratified filter, and T is, for example, the 34% of S.

The set B is obtained by randomly undersampling the set A. B has the same number of records tagged with (+) and (-)

Since the undersampling technique can lead to a loss of information, in a more advanced way, the random removal of minority class records (+) can be replaced by applying a "bootstrap" ("bagging") approach [35]. In a nutshell, the set A is subdivided into N subsets, in which the elements of the minority class (+) are all fixed, while the records of the majority class (-) are randomly sampled with replacement in a number equal to records tagged with minority class (+). In this way, each of the N subsets is balanced, the elements tagged with "+" do not vary, and may have records tagged with "-" in common. In a formal way:

$B_i = D \cup R_i \ (i=1,\ldots,\mathrm{N})$
where $D=\{x \in A : x \ has \ "+" \ class\}$ and $|D|=m$
$R_i=\{x \in A : x \ has \ "-" \ class\}$, $|R_i|=m, \forall i=1,\ldots,\mathrm{N}$
$B=\bigcup B_{ii=1,\ldots,N}$

The model C (Figure 3), called controller, is an ensemble [36] of N classifiers C_i $(i=1,\ldots,N)$. Each C_i is a cross-validated classifier trained on a different balanced set B_i. Moreover, each C_i is trained by a different learning algorithm. Finally, the class can be obtained by taking a majority vote on the individual predictions of the C_i base classifier

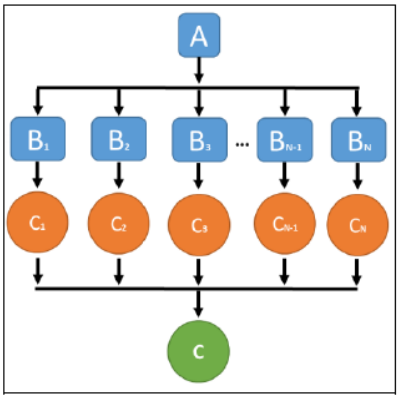

Figure 3. Combine classifiers in the ensemble schema of COSM

Additionally, C is tested on the set T, by calculating the metrics of the Table 2, based on confusion matrix [8].

The COSM procedure proceeds by considering the subset D of A consisting of only the m elements of the minority class ($D=\{x \in A : x \ has \ "+" \ class\}$), and by applying an oversampling technique $\mathcal{M}$ to D in order to create a set ($M2$) of new synthetic minority class examples.

$M1$ is the set of all the minority class examples, including the new synthetic ones ($M2=M1-D$). The number of elements of $M2$ depends on $\mathcal{M}$ and its parameters.

The controller model C is applied on the new set $M2$ in order to reject the examples that are misclassified by C: these elements are false positives according to classifier C.

Table 2. Classification Performance Evaluation Metrics

Name	Formula	Description
Accuracy	Acc=(TP+TN)/ (TP+TN+FP+FN)	Fraction of correct predictions on the total number of predictions
True Positive Rate / Sensitivity	TPR=TP/(TP+FN)	Fraction of positive examples predicted correctly by the classifier
True Negative Rate / Specificity	TNR=TN/(TN+FP)	Fraction of negative examples predicted correctly by the classifier
False Positive Rate	FPR=FP/(TN+FP)	Fraction of negative examples predicted as a positive class
False Negative Rate	FNR=FN/(TP+FN)	Fraction of positive examples predicted as a negative class
Precision	Prc=TP/(TP+FP)	Fraction of examples classified as positive that are really positive
AUC		Area Under the ROC Curve

Finally, W is the subset of $M2$ well classified by C: it is made up of non-rejected elements. And $E=A \cup W$ is the new extended training set, which is the training set for the final classification model $\mathcal{F}$ and which is tested on the test set T. The performances of $\mathcal{F}$ on T can be compared with the performances of C on T.

5 Conclusion

Overcoming the problem of the unbalanced class depends on numerous elements. It depends on complexity of the data, severity of class imbalance, size of data and classifier involved. The framework designed here can be applied independently of the Machine Learning algorithm or the selected oversampling technique.

COSM needs to be tried and tested, especially to define a strategy for selecting the following its parameters:

- the percentage of S to get the test set T;
- the algorithm to obtain the controller model C trained on set B;
- the oversampling technique $\mathcal{M}$;
- the algorithm to obtain the final classifier trained on the set E.

As mentioned above, COSM can be entirely employed, for example, by using WEKA software. The paper describes the design of the methodology, while its implementation with all the experimental tests will be addressed in a future work.

ACKNOWLEDGMENT

The author would mention the Big Data Facility and COND-IWT projects, both funded by the Italian PRORA.

REFERENCES

[1] A. Ali, S. M. Shamsuddin, and A. L. Ralescu, *Classification with class imbalance problem: A review*, Int. J. Advance Soft Compu. Appl, vol. 7, no. 3, November 2015, pp. 176-204, ISSN: 2074-8523.

[2] S. M. A. Elrahman and A. Abraham, *A Review of Class Imbalance Problem*, Journal of Network and Innovative Computing, vol. 1, 2013, pp. 332-340, ISSN: 2160-2174.

[3] N. V. Chawla, N. Japkowicz, and A. Kotcz, *Editorial: special issue on learning from imbalanced data sets*, SIGKDD Explorations, vol. 6, no. 1, 2004, pp. 1-6.

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

[5] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta, *Metalearning: applications to data mining*, Springer-Verlag Berlin Heidelberg, 2009, doi: 10.1007/978-3-540-73263-1.

[6] S. Raschka, *Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning*, ArXiv abs/1811.12808, 2018.

[7] D. M. Hawkins, *The Problem of Overfitting*, Journal of Chemical Information and Computer Sciences 44(1):1-12, May 2004, doi: 10.1021/ci0342472.

[8] P. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, Pearson Addison Wesley, 2005.

[9] N. Thai-Nghe, Z. Gantner, and L. Schmidt-Thieme, C*ost-sensitive learning methods for imbalanced data*, in The 2010 International Joint Conference on Neural Networks (IJCNN), Barcelona, 2010, pp. 1-8, doi: 10.1109/IJCNN.2010.5596486.

[10] C. Elkan, *The Foundations of Cost-Sensitive Learning*, in Proc. Of the 17th Intl. Joint Conf. on Artificial Intelligence, August 2001, pp. 973-978.

[11] M. Galar, A. Fernandez, E. Barrenechea, and H. Bustince, *A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches*, in IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 42, no. 4, pp. 463-484, 2012, doi: 10.1109/TSMCC.2011.2161285.

[12] W. C. Lin, C. F. Tsai, Y. H. Hu, and J. S. Jhang, *Clustering-based undersampling in class-imbalanced data*, Information Sciences, voll 409–410, October 2017, pp. 17-26, doi: doi.org/10.1016/j.ins.2017.05.008.

[13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, *SMOTE: Synthetic Minority Over-sampling Technique*, in Journal of Artificial Intelligence Research 16, 2002, pp. 321-357.

[14] H. He, Y. Bai, E. A. Garcia, and S. Li, *ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning*, 2008 International Joint Conference on Neural Networks (IJCNN 2008), Honk Hong, 2008, pp. 1322-1328, doi: 10.1109/IJCNN.2008.4633969.

[15] A. Sánchez, E. Morales, and J. Gonzalez, *Synthetic Oversampling of Instances Using Clustering*, International Journal of Artificial Intelligence Tools, 22, 2013, doi: 10.1142/S0218213013500085.

[16] P. D. Turney, *Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm*, in Jouirnal of Artificial Intelligence Research 2, 1995, pp. 23-26.

[17] C. X. Ling, Q. Yang, J. Wang, and S. Zhang, *Decision Trees with Minimal Costs*, in Proceedings of the 21st International Conference on Machine Learning (ICML), Banff, Canada, 2004.

[18] P. Domingos, *MetaCost: A General Method for Making Classifiers Cost-Sensitive*, in Proceedings of the fifth international conference on knowledge discovery and data mining, SIGKDD, San Diego, ACM, New York, 1999, pp. 155-164.

[19] I. H. Witten and E. Frank, *Data Mining. Practical Machine Learning Tools and Techniques with Java implementations*, Morgan Kaufmann, San Francisco, 2005.

[20] X. Chai, L. Deng, Q. Yang, and C. X. Ling, *Test-Cost Sensitive Naive Bayes Classification*, in Proceedings of the fourth IEEE International Conference on Data Mining, ICDM, 2004, doi: 10.1109/ICDM.2004.10092.

[21] V. S. Sheng and C. X. Ling, *Thresholding for Making Classifiers Cost-sensitive*, in Proceeedings of the 21st National Conference on Artificial Intelligence, 2006, pp. 476-481.

[22] S. Wang and X. Yao, *Diversity analysis on imbalanced data sets by using ensemble models*, in IEEE Symp. Comput. Intell. Data Mining, pp. 324-331, 2009.

[23] D. Tao, X. Tang, X. Li, X. Wu, *Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval*, IEEE Trans. Pattern Anal. Mach. Intell., vol. 28, no. 7, pp. 1088-1099, Jul. 2006.

[24] C. Li, *Classifying imbalanced data using a bagging ensemble variation (BEV)*, in Proc. 45th An. Southeast Reg. Conf., NY:ACM, pp. 203-208, 2007.

[25] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, *SMOTEBoost: Improving prediction of the minority class in boosting*, Proc. Knowl. Discov. DB, pp. 107-119, 2003.

[26] C. Seiffert, T. Khoshgoftaar, J. Van Hulse, and A. Napolitano, *Rusboost: A hybrid approach to alleviating class imbalance*, IEEE Trans. Syst. Man Cybern. A Syst. Humans, vol. 40, no. 1, pp. 185-197, Jan. 2010.

[27] X. Y. Liu, J. Wu, and Z. H. Zhou, *Exploratory undersampling for class-imbalance learning*, IEEE Trans. Syst. Man Cyber. Appl. Rev, v. 39, n. 2, pp. 539-550, 2009.

[28] T. Maciejewski and J. Stefanowski, *Local Neighbourhood Extension of SMOTE for Mining Imbalanced Data*, in Proceeedings of 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), Paris, pp. 104-111, 2011, doi: 10.1109/CIDM.2011.5949434.

[29] B. Santoso, H. Wijayanto, K. A. Notodiputro, and B. Sartono, *Synthetic Over Sampling Methods for Handling Class Imbalanced Problemas: A Review*, in IOP Conf. Series: Earth and Environmental Science 58, 2017, doi: 10.1088/1755-1315/58/1/012031.

[30] S. Hu, Y. Liang, L. Ma, and Y. He, *MSMOTE: Improving classification performance when training data is imbalanced*, in Proc. 2nd Int. Workshop Comput. Sci. Eng., vol. 2, 2009, pp. 13-17.

[31] N. V. Chawla, *C4.5 and Imbalanced Data sets: Investigating the Effect of Sampling Method*, Probabilistic Estimate, and Decision Tree Structure, in ICML Workshop on Learning from Imbalanced Data sets, Washington, DC, 2003.

[32] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten (2009), *The WEKA Data Mining Software: An Update*, SIGKDD Explorations, Volume 11, Issue 1.

[33] I. H. Witten and E. Frank, *Data Mining. Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, 2005.

[34] P. Chapman et al., *CRISP-DM 1.0. Data Mining guide*, 2000.

[35] L. Breiman, *Bagging predictors*, Machine Learning, vol. 24, no. 2, pp. 123–140, 1996, doi:10.1007/BF00058655.

[36] L. I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, John Wiley and Sons, Inc., 2004.