Journal of Applied Sciences, Information and Computing

Volume 4, Issue 2, November 2023

ISSN: 1813-3509 https://doi.org/10.59568/JASIC-2023-4-2-03

A REVIEW OF CLUSTER UNDER-SAMPLING IN UNBALANCED DATASET AS A METHODS FOR IMPROVING SOFTWARE DEFECT PREDICTION

Abdulhamid Sani[1], V. S. Manjula[2,] Musa Ahmed Zayyad[3]

[1,]Department of computer science, School of Mathematics and Computing, Kampala International University Uganda; abdulhamidsanihja@gmail.com;

[2]Corresponding Author, Professor, Department of Computer Science, Kampala International University, Kampala, Uganda, East Africa, **manjusunil.vs@gmail.com,**
Orcid ID: (0000-0003-0308-3289)
[3]Department of Information Technology, School of Mathematics and Computing, Kampala International University Uganda; zayyad.musa@kiu.ac.ug

Abstract

In many real-world machine learning applications, including software defect prediction, detecting fraud, detection of network intrusion and penetration, managing risk, and medical dataset, class imbalance is an inherent issue. It happens when there aren't many instances of a certain class mostly the class the procedure is meant to identify because the occurrence the class reflects is rare. The considerable priority placed on correctly classifying the relatively minority instances—which incur a higher cost if incorrectly categorized than the majority instances—is a major driving force for class imbalance learning. Supervised models are often designed to maximize the overall classification accuracy; however, because minority examples are rare in the training data, they typically misclassify minority instances. Training a model is facilitated by balancing the dataset since it keeps the model from becoming biased in favor of one class. Put another way, just because the model has more data, it won't automatically favor the majority class. One method of reducing the issue of class imbalance before training classification models is data sampling; however, the majority of the methods now in use introduce additional issues during the sampling process and frequently overlook other concerns related to the quality of the data. Therefore, the goal of this work is to create an effective sampling algorithm that, by employing a straightforward logical framework, enhances the performance of classification algorithms. By providing a thorough literature on class imbalance while developing and putting into practice a novel Cluster Under Sampling Technique (CUST), this research advances both academia and industry. It has been demonstrated that CUST greatly enhances the performance of popular classification techniques like C 4.5 decision tree and One Rule when learning from imbalance datasets.

Keywords: Algorithms, Decision Tree, CUST, Data sampling, One Rule, Software Defect Prediction, Unbalanced Dataset.

1. INTRODUCTION

In supervised learning tasks, classification is predicting the category or class of unseen data by using labelled categorical data as a source of knowledge (Demidova & Klyueva, 2018). The issue of class imbalance is one of the primary difficulties in supervised categorization. The condition when a dataset has a higher representation of one class than another is referred to as this problem. The negative class (majority class) and the positive class (minority class) are the two main groupings or classes into which the data in this issue domain is divided. There are far more examples in the majority class than in the minority class. In the fields of machine learning, classification is a well-researched technique (Li et al.,, 2014; Challagulla, 2015; Malhotra, 2012).

Numerous real-world applications of machine learning algorithms include the detection of faults (Han & Wang, 2016), the detection of email spam (Zeng & Wang, 2015), the prediction of cancer (Schaefer, 2015 & Reza, 2019), the detection of credit card fraud (Subudhi, 2018; Bauder, 2018; Mohammed, 2018; Melo-Acosta, 2017), intrusion detection (Rodda, & Erothi, 2022), and many more. Unfortunately, the datasets used in these real-world applications are typically unbalanced, which means that most machine learning techniques on these datasets yield unfavorable results. (Ofek, Rokach, Stern, & Shabtai, (2022)); Sowah, Agebure, Godfrey, Koumadi, & Fiawoo, (2020). As a result, several classification algorithms predict the majority class with high accuracy while predicting the minority class with comparatively low accuracy (Gao, Hong, Chen, Harris, & Khalaf, 2021). However, the majority class finds the minority class more appealing because its instances are significantly fewer than those of the majority class (Gao, Hong, Chen, Harris, & Khalaf, 2021). Therefore, having balanced prediction accuracy for both the negative and positive classes' occurrences is of study interest.

It is impossible to ignore the problem of class imbalance in real-world datasets and how it affects machine learning and statistical methods. This research consider some outstanding data level methods to imbalance learning as well as new sampling techniques that take other data quality issues like noise or inconsistent instances and outliers into consideration when sampling in order to improve the performance and reliability of statistical and machine learning algorithms when learning from imbalance datasets. This study presents a novel Hybrid Cluster-Based Sampling Technique (HCBST) that could be applied to enhance classification algorithms' performance when they are learning from unbalanced datasets. The efficacy of data level approaches serves as the technique's driving force.

1.1 Objectives of the Study

1. To develop an effective method for under sampling instances of the majority class in datasets that are imbalanced.
2. To put the method into practice using Python
3. To test the method with One R Classification Algorithm and C4.5 Decision Tree
4. To appraise and evaluate the suggested technique's effectiveness by contrasting its results with those of previously established sampling approaches

1.2 Research Questions

1. In imbalanced datasets, how can we develop an effective method for under sampling instances of the dominant class?
2. How can the Python language be used to implement the technique?
3. How can the C4.5 Decision Tree and One R Classification algorithms be used to test the method?
4. How can the effectiveness of the suggested technique be determined by contrasting its results with those of sampling procedures that are currently in use?

2. METHODOLOGY

The methodologies employed in the study are covered in this section, together with the datasets, sampling strategies, tools, system specs, performance measures, and experimental techniques.

2.1 Datasets

The experimental setup would make use of eleven datasets, five of which came from the University of California Irvine Repository (Frank, 2017) and six of which came from the National Aeronautics and Space Administration (NASA) Metric Data Program (MDP) (Gray, Bowes, Davey, Sun, & Christianson, 2011). The datasets used from both repositories are listed in Table 1. The NASA MDP datasets are made up of information from several software projects that NASA has worked on. The original information was retrieved from a backup by (Tantithamthavorn, 2016). All 13 NASA Software defect datasets were subjected to a rigorous data purification process that was fully detailed by Gray (2011) in 2011. As a consequence, each dataset contained between 6 and 90% of its original data. The datasets from Shepperd (2013) have been cleansed and will be utilized as backups in this study. Shepperd (2013) sought to determine the extent to which research analyses that have been published and based on the NASA Software Defect Datasets are relatively informative (Shepperd, 2013).

But Petrić (2016) found further guidelines for eliminating problematic data that Shepperd (2013) had missed. By following these guidelines, it was possible to determine which two of the 13 NASA Software Defects datasets—JM1 and MC2—were the most troublesome (Petrić, 2016). The machine learning community uses the UCI Repository, a collection of databases, domain theories, and data generators, for the empirical investigation of machine learning algorithms. The multi-class datasets from the UCI Repository would be transformed into binary classification issues for this investigation. A summary of the datasets and their class distributions is presented in Table 1. The study will utilize NASA Software Defects and UCI datasets as data samples due to their public availability and widespread usage by fellow academics conducting related research (Petrić, 2016).

Table 1. Summary of Datasets

Dataset		Attributes	Cumulative Instances	Minority Class	Majority class instances	Ratio of Imbalance
NASA Datasets						
	PC1	39	761	63	699	11.54
	PC2	39	1586	18	1568	98.16
	MW1	37	265	28	237	8.78
	MC2	41	128	45	83	1.89
	CM1	39	345	42	302	7.19
	KC3	42	201	36	164	4.56
UCI Datasets	Abalone,	8	730	41	688	15.40
	Abalone.	9	4177	32	4145	129.5
	Ecoli.	7	335	19	315	1.80

	Glass,	11	215	18	198	11.69
	Yeast.	10	265	21	245	12.30

2.2 Methods of Sampling:

In this study, the suggested sample approach will be evaluated for effectiveness against eight (8) other data sampling methods. The techniques include one-sided selection, cluster under sampling technique, adaptive synthetic sampling the Synthetic Minority Oversampling Technique, random under sampling, random oversampling, and under-Sampling based on Clustering. Python would be used for his study to implement the CLU Ster-based hybrid sampling approach, under-sampling based on clustering, and the Cluster Under sampling Technique. Utilizing Scikit-learn (Pedregosa, 2011), the remaining conventional sample methods would be applied**.**

2.3 Tools Employed

Scikit-learn, a Python machine learning package that completely integrates a variety of machine learning techniques, and the Python Environment running on 64-bit Windows_10 are the tools utilized in this work (Pedregosa, 2011). Compared to equivalent C or C++ programs, Python is a very basic interpretative language that makes it possible to write more understandable and generally much shorter code (Van, Voor, Rossum, 1995). Among the scikit-learn classification techniques used were K-Nearest Neighbors, Support Vector Machines, Random Forest, Multilayer Perceptron, Adaboost, Naïve Bayes, and Quadratic Discriminant Analysis. The default parameters of the classifiers were used in this experiment.

2.4 System Requirements

The following describes the platform's system definition, which served as the foundation for developing the suggested technique:

Table 2. System specifications

Operating System	Windows_10
Architecture of the system	64bits
Random Access Memory	32GB
Central Processing Unit	Core i7
No of Core	4(four)
Threads	8(eight)
Caches	6MB
Frequency	2.8GHz
Turbo	3.8GHz
Storage Capacity	256GB M2 SATA III SSD and 2 TB SATA HDD

The primary performance metrics to be employed in assessing the classifiers' efficacy are the Matthews Correlation Coefficient (MCC), Geometric Mean (G-Mean), and Area Under the Receiver-operating-

characteristic Curve (AUC). The scikit-learn classifiers would have routines for calculating these performance measures.

2.5 Experimentation Approach

Prior to training the classification models, the training data would be sampled using eight different sampling procedures. Using stratification and a random seed, each dataset would first be divided into training and testing to provide an accurate out-of-sample performance evaluation. By using stratification, it would be possible to split the data so that each of the resultant datasets had an equal representation of the original dataset. The results would be easily repeatable thanks to the random seed.

On the training dataset, stratified tenfold cross-validation would also be done. Every time a validation is performed on the training dataset, the held-out data is used to estimate the validation performance. The results are then recorded as validation performance for further examination, and the completed model is employed to evaluate the test dataset and document the testing results. After the tenfold cross-validation is complete, the results would be averaged and documented for both the testing and validation performances. The same process would be carried out ten times, with a different random seed value each time, to further decrease biases that might

have been introduced during the stratification phase in the division of the training and testing data or the cross-validation process. The validation and testing results for each dataset are averaged to determine the overall performance of the classifier under consideration. The testing findings would dictate the overall performance of the model.

3. FINDINGS AND INTERPRETATION

The study's design for the Hybrid Cluster-Base Sampling Technique (HCBST) and the equations employed to meet the predetermined study objectives are included in the results and analysis section that follows.

3.1 Designed of hybrid cluster-based sampling approach

The HCBST method outperforms existing sampling strategies using the k-means algorithm in terms of computational time while also boosting the overall effectiveness of well-known methods for machine learning. The HCBST approach consists of two steps: the oversampling stage, in which synthetic minority class instances are created using SMOTE (Chawla, Bowyer, Hall, & Kegelmeyer, 2022).

The second step, referred to as under sampling, is based on the notion of using clustering to identify outliers in data (Kaufman & Rousseeuw, 1990; Rocke & Woodruff, 1996). The HCBST design allowed for the flexible use of various sample settings for the under sampling and oversampling procedures. Thus, the sampling parameters would be estimated before the sample process. Figure 3.2 provides a summary of the HCBST design. The oversampling procedure, derived from SNOCC, is a method for oversampling minority class instances (Zheng, Cai, & Li,

2015). Unlike SMOTE, where the synthetic samples are placed on the line segment between the seed samples, SNOCC uses a technique to construct synthetic samples inside the region bounded by the line segments between the seed samples. SNOCC calculates the distances to the k-nearest neighbors, their mean mi, and their sigma, which is equal to the average of mi plus the standard deviation from equation (2.2), using seed samples from the minority class as an input.

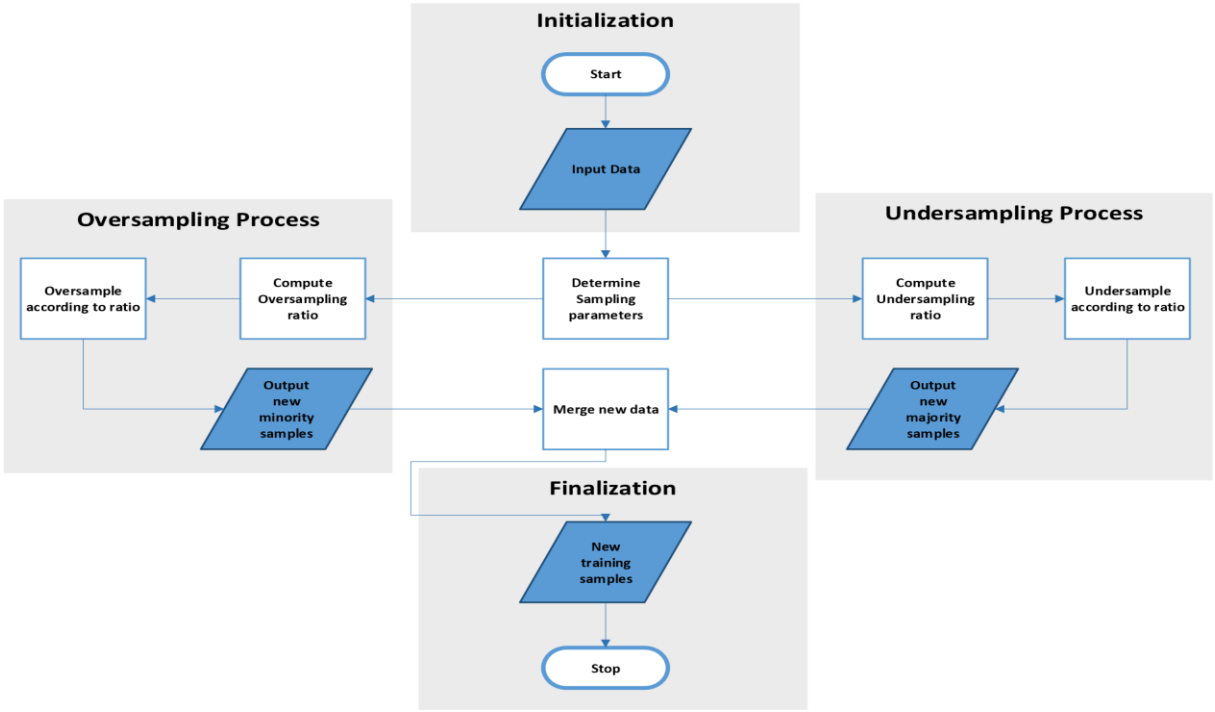

Figure 1. An overview of the HCBST design

Next, a seed sample (s1) and its two nearest neighbors (s2 and s2)—referred to as sigma nearest neighbors—are chosen at random by the algorithm so that their distances are smaller than sigma (Zheng, Cai, & Li, 2015). Then, it creates a vector in three dimensions (b1, b2, b3) so tha $b_1 + b_2 + b_3 = 1$ (eq. 1)

At last, the new synthetic sample is produced by using the equation:

$$S = b_1 s_1 + b_2 s_2 + b_3 s_3 = 1 \quad \text{(eq. 2)}$$

The process would be continued until the required number of minority samples were obtained. In 2015, Zheng, Cai, and Li shown by experiments that this approach yields synthetic samples that, in comparison to SMOTE, are more representative of the actual minority samples. However, the presence of majority samples is not taken into account in the distribution space of the seed samples that were used to produce the synthetic samples. To address this issue, HCBST would use a selection criterion that eliminates synthetic samples that are more likely to overlap with

majority samples. To apply the HCBST method, which oversamples, the researcher needs to provide three parameters: r0, bm, and 0s. The number of sampled minority instances and sampled majority instances should be matched, as shown by Bm and 0s, respectively. to return only synthetic samples or both original and synthetic samples. The number of minority samples needed to produce N0, which is provided by the following equation, would be ascertained using the value r0

$$N_0 = N_m (r_0 - 1) \quad \text{(eq. 3)}$$

where Nm is the initial minority sample number.

If r0 is set to 1, oversampling won't happen, and if r0 is set to 2, sampled minority cases will almost quadruple the number of original minority samples. In line with SNOCC, HCBST generates a synthetic sample after determining the required quantity of minority samples. The distance to the closest majority sample is calculated for each of the sigma neighbors, s2 and s2, and the average of these distances is determined as s□. Lastly, it finds the distance, dn, between the new synthetic sample and the closest majority sample. If dn's, a new seed sample, S1, would be used to resume the process after discarding the new synthetic sample. The process would be repeated until the number of acceptable synthetic samples was equal to zero.

During the second stage of the sampling process, a CUST approach would be used to under sample instances of the majority class. CUST groups the remaining samples into k-clusters after removing inconsistent samples using a method based on Tomek connections. It removes duplicates by selecting majority samples for each cluster according to a preset ratio. Like SNOCC, CUST does not consider the local closeness of the nearby class instances. Therefore, it may be decided to select majority cases that coincide with minority situations. To overcome this issue, HCBST uses a technique similar to the oversampling process to filter out majority instances that are particularly likely to overlap with examples of minority classes. An overview of the oversampling process is shown in Figure 3.

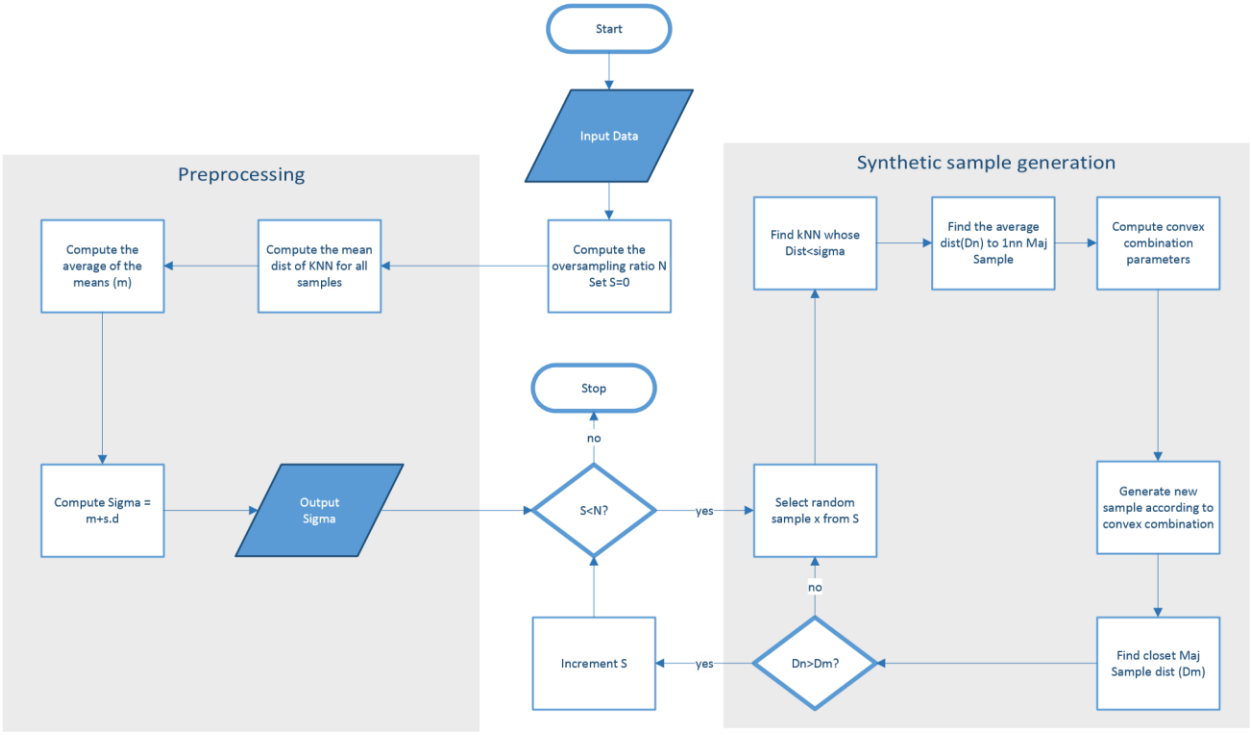

Figure 2. Oversampling Process

In order for the HCBST method to carry out the under sampling process, the experimenter needs to provide the parameters ru, pm, km, and km. The ratio of majority samples to original minority samples after under sampling is known as Ru. The number of clusters to use from the majority samples is indicated by km, the number of minority neighbors to search for is indicated by km, and the fraction of majority samples to discard is indicated by pm.

In the first stage of the under sampling process, the algorithm would select the mode of under sampling by selecting or rejecting samples from the majority samples according to the criteria set by the researcher. When the parameter pm is set, the algorithm is told to under sample by rejecting majority occurrences. In all other circumstances, the option ru will be used to select samples from the majority of cases. If pm was set to 0.1, 10% of the majority samples would be removed, and the parameter ru would be ignored. The majority samples would be clustered into km clusters by the program using the means technique when the under sampling mode was identified. In the event that pm is not specified for every cluster, the algorithm will proceed using CUST. To determine the required number of samples to be selected from each cluster, utilize equation (3.6).

$$MI\ Maj^{i} = r_u \times \frac{}{MA \neq 0} \times MC^{i};\ 1 \leq I \leq k_m, \quad \text{(eq. 4)}$$

Where MCi is the number of majority class samples in the ith cluster, MI is the total a few of sporadic instances, MA is the total number of majority instances, and Maji is the number of majority instances to select from cluster I. Subsequently, a random instance would be selected from the cluster, with copies of previously selected instances being rejected.

Unlike CUST, HCBST takes into account the local proximity of minority circumstances. A minority instance, Sci, is removed from the cluster if any of its kn nearest neighbors are minority instances. This process is done after choosing an instance, Sci, from the cluster. If not, it would add to the selected samples. The procedure would be repeated until the required number of instances from each cluster are acquired. The selected samples from each cluster and the minority samples would be combined to generate the new training set. On the other hand, if pm is set, the number of majority samples that need to be removed from each cluster is decided by;

$Maj^i = Pm \times MA; \quad 1 \leq I \leq k_m$

(eq. 5)

A random instance named Sci would be selected from the cluster, and its kn nearest neighbors would be looked for. Sci would be rejected if any of the kn -1 neighbors were minority cases. The procedure would be continued until the required number of instances were gathered to be removed from each cluster. To build the new training set, each cluster's surviving examples and minority samples would be combined. But keep in mind that during the k-means clustering stage, distance caching would be used to expedite the under sampling computation.

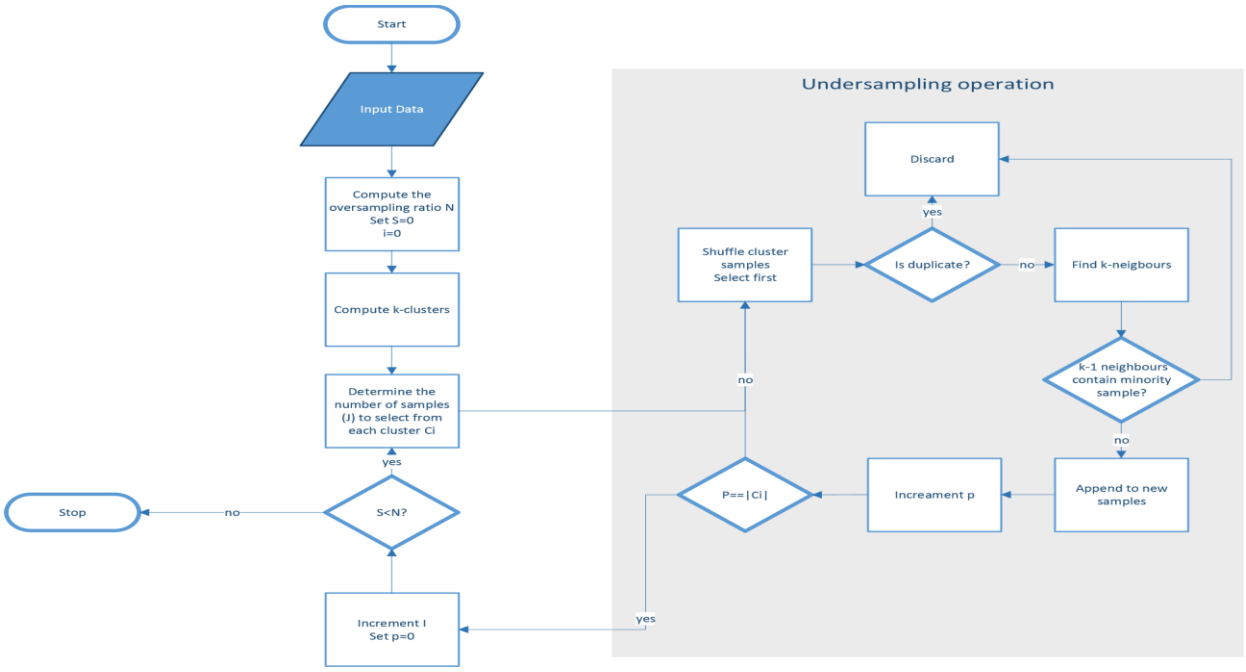

Figure 3. Under sampling Process

4. CONCLUSION

The hybrid cluster-based sampling methodology was developed to improve the overall performance of machine learning algorithms while cutting down on computing time, in contrast to other sampling strategies that employ the k-means algorithm. Two steps of the HCBST method are the oversampling stage, when SMOTE is used to create synthetic minority class instances (Chawla, Bowyer, Hall, & Kegelmeyer, 2022). The idea of employing clustering to find outliers in data is the foundation of the second stage, known as under sampling (Kaufman & Rousseeuw, 1990; Rocke & Woodruff, 1996). Different sample parameters for the under sampling and oversampling processes could be used with flexibility thanks to the design of HCBST.

5. REFERENCES

[1]Acuña E. & Rodríguez, C. (2015) "An empirical study of the effect of outliers on the misclassification error rate," *Transactions on Knowledge and Data Engineering*.

[2]Altman, N. S. (2019) "An introduction to kernel and nearest-neighbor nonparametric regression," *Am. Stat.*, doi: 10.1080/00031305. 10475879.

[3]Bauder, R. A., & Khoshgoftaar, T. M. (2018) "Medicare fraud detection using random forest with class imbalanced big data," in *Proceedings - 2018 IEEE 19th International Conference on Information Reuse and Integration for Data Science, IRI 2018*, 2018, doi: 10.1109/IRI.2018.00019.

.

[4]Chakkrit Tantithamthavorn, Ahmed E Hassan, and Kenichi Matsumoto. The impact of class rebalancing techniques on the performance and interpretation of defect prediction models. IEEE Transactions on Software Engineering (TSE), 2018.

[5]Challagulla, F. B., Bastani, I. L. & Paul, R. A. (2015) "Empirical Assessment of Machine Learning based Software Defect Prediction Techniques," *Proc. of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'05).*

[6]Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. (2022)"SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, doi: 10.1613/jair.953.

[7]Chawla, N. V., Hall, L. O., Bowyer, K W. & Kegelmeyer, W. P. (2022) "SMOTE: Synthetic minority oversampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321-357.

[8]Cover, T. M. & Hart, P. E. (1967) "Nearest Neighbor Pattern Classification," *IEEE Trans. Inf. Theory*, doi: 10.1109/TIT.1053964.

[9]Demidova, L. & Klyueva, I. (2018) "Data classification based on the hybrid versions of the particle swarm optimization algorithm," in *7th Mediterranean Conference on Embedded Computing, MECO 2018 - Including ECYPS 2018, Proceedings*, 2018, doi: 10.1109/MECO.2018.8406069.

[10]Drown, D. J., Khoshgoftaar, T. M. & Seliya, N. (2019) "Evolutionary Sampling and Software Quality Modeling of High-Assurance Systems," *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, Vol. 39, No. 5.

[11]Drummond, C. & Holte, R. C. (2013) "C4.5, class imbalance, and cost sensitivity: why under sampling beats over-sampling," *Work. Learn. from Imbalanced Datasets II*, doi: 10.1.1.68.6858.