JURNAL INFOTEL
Informatics - Telecommunication - Electronics

Website Jurnal : http://ejournal.st3telkom.ac.id/index.php/infotel
ISSN : 2085-3688; e-ISSN : 2460-0997

Optimization of software defects prediction in imbalanced class using a combination of resampling methods with support vector machine and logistic regression

Windyaning Ustyannie[1], Emy Setyaningsih [2]*, Catur Iswahyudi [3]
[1,2] Department of Computer Systems Engineering, Institut Sains & Teknologi AKPRIND
[3] Department of Informatics, Institut Sains & Technology AKPRIND
[1,2,3] Kalisahak Street No. 28, Yogyakarta 55222, Indonesia
*Corresponding email: emysetyaningsih@akprind.ac.id

Received 18 October 2021, Revised 20 November 2021, Accepted 7 December 2021

Abstract — The main problem in producing high accuracy software defect prediction is if the data set has an imbalance class and dichotomous characteristics. The imbalanced class problem can be solved using a data level approach, such as resampling methods. While the problem of software defects predicting if the data set has dichotomous characteristics can be approached using the classification method. This study aimed to analyze the performance of the proposed software defect prediction method to identify the best combination of resampling methods with the appropriate classification method to provide the highest accuracy. The combination of the proposed methods first is the resampling process using oversampling, under-sampling, or hybrid methods. The second process uses the classification method, namely the Support Vector Machine (SVM) algorithm, and the Logistic Regression (LR) algorithm. The proposed, tested model uses five NASA MDP data sets with the same number attributes of 37. Based on the t-test, the $P_{Value} < \alpha = 0.0344 < 0.05$ and the $t_{count} > t_{table} = 3.1524 > 2.7765$, indicates that the combination of the proposed methods is suitable for classifying an imbalanced class. The performance of the classification algorithm has also improved with the use of the resampling process. The average increase in AUC values using the resampling in the SVM algorithm is 17.19%, and the LR algorithm is at 7.26% compared to without the resampling process. Combining the three resampling methods with the SVM algorithm and the LR algorithm shows that the best combining method is the oversampling method with the SVM algorithm to software defects prediction in imbalanced class with an average accuracy value of 84.02% and AUC 91.65%.

Keywords – defect prediction, imbalanced class, logistic regression, resampling, support vector machine.

I. INTRODUCTION

Software is part of a computer system developed by many companies or government agencies. Making software aims to make activities run effectively, quickly, and accurately, but there are obstacles in software development, namely costly costs [1], [2]. Therefore, high-quality software development must be followed, free from module errors [3]. A module error in software is called a software defect. Therefore, we need a program to identify modules prone to defects and predict errors in these modules [4], [5].

According to [6], research on software defects is divided into three focuses: estimation, association rules, and classification of software defects. In addition to focusing on these three topics, software defect research also focuses on clustering and data set analysis [7]. One of the most popular approaches to software flaw prediction is the classification algorithm. The classification algorithm will categorize software based on its attributes towards defects or not [2]. The popular classification method for software defects prediction can use two approaches, namely machine learning and statistics. The machine learning techniques used are the Decision Tree algorithm [8]–[10], Support Vector Machine (SVM) [2], [4], Naïve Bayes (NB) [5], Artificial Neural Network (NN) [11], [12], K-Nearest Network [13], [14], Artificial Neuro-fuzzy [15], [16], and Random Forest [17]. In comparison, the statistical

approach to classifying software defects is to use regression models such as Logistics Regression (LR) [18] and Multiple Linear Regression (MLR) [19].

In general, software defects have a class distribution with fewer defective classes than non-defective ones [15], [20]. Software defects are a minority class, so there will be many defects that are not found. There is currently no standard for what percentage of a class of data sets is said to be unbalanced. Researchers agree that a data set is unequal if one class has a percentage of 2 times that of another class [21].

Data with an imbalanced class can affect the algorithm's performance because the prediction results produce a majority class [22], [23]. Approaches to solving problems that occur in data with an imbalanced class can be made using two methods, namely data level and algorithm level [23]. The data level approach is usually used when there is data with sporadic classes. The algorithm level approach is used to find patterns to fit the data with an imbalanced class. The process at the data level is done by the resampling method. The resampling method is used to improve the class distribution of the data. In general, resampling methods can be divided into oversampling, under-sampling, and hybrid [24]. Oversampling is the simplest method in dealing with minority classes by conducting random classes at the time of sampling by duplicating positive classes and balancing classes randomly. If the oversampling method works by adding data to the minority class, then the under-sampling process reduces the data in the majority class [22]. The hybrid method is a method that uses a combination of oversampling and under-sampling techniques.

Some of the resampling methods proposed include Random Oversampling (ROS), Random Under-Sampling (RUS), Synthetic Minority Oversampling Technique (SMOTE) [6], Fractal Synthetic Minority Oversampling Technique (FSMOTE) [25], and Multi-label Synthetic Minority Oversampling Technique (MLSMOTE) [26]. The simplest oversampling method is the random oversampling method. The random oversampling process works by randomly selecting minority data and duplicating it until balanced class distribution [21]. However, several studies have shown that the under-sampling method is less efficient than the oversampling method because this method eliminates the majority of data, resulting in the loss of important information from the data set, especially in small data sets [26].

Referring to the research that has been done, the contribution of this research is the proposed software defect prediction method to identify the best combination of resampling methods with the appropriate classification method to provide the highest accuracy. The first process of the proposed methods is a resampling process using oversampling, under-sampling, and hybrid approaches. The second process is the software defect prediction process using the classification method. The SVM algorithm represents the machine learning-based algorithm, and the LR algorithm represents the statistical approach. The SVM method proposed in this study is due to the advantages of SVM, which has hyperplane capabilities. The concept of hyperplane is the ability to separate data from its class, especially for data with large margins. SVM also can generate adaptive boundaries, which is very important in dealing with the problem of imbalanced class [23]. The LR algorithm proposed in this study, due to the LR algorithm, is one of the most widely used regression algorithms, easy to implement and interpret [7].

Since 2000, 64.79% of software defect prediction studies have an imbalanced class using data sets on the National Aeronautics and Space Administration (NASA) Metrics Data Program (MDP) repository [7]. Therefore, to test this study's proposed software defect prediction model using a data set sourced from the NASA MDP repository [2], [27]. The software defect prediction method with an imbalanced class is the best measured using the value of Area Under Curve (AUC) and accuracy [28].

The remainder of this paper is organized as follows. Section 2 covers the research flow of the proposed software defect prediction model using a combination of resampling methods with SVM and LR classification methods. Then, it describes the comparison of the measurement results of the proposed software defect prediction model in section 3 and section 4. Finally, section 5 concludes the article.

II. RESEARCH METHODS

This paper proposed a software defect prediction model using a combination of the best resampling method approach with the appropriate classification method to provide the highest accuracy, as shown in Fig. 1.

Based on Fig. 1, this research process consists of three main steps; in the first step, we carry out the resampling process using oversampling, under-sampling, and hybrid methods to balance the data before the classification process is carried out. After the data is balanced, the second step is followed by the classification process using the SVM and LR algorithms. Our last step is testing to determine the combination of resampling and classification algorithms that produce the highest accuracy by measuring accuracy and AUC values.

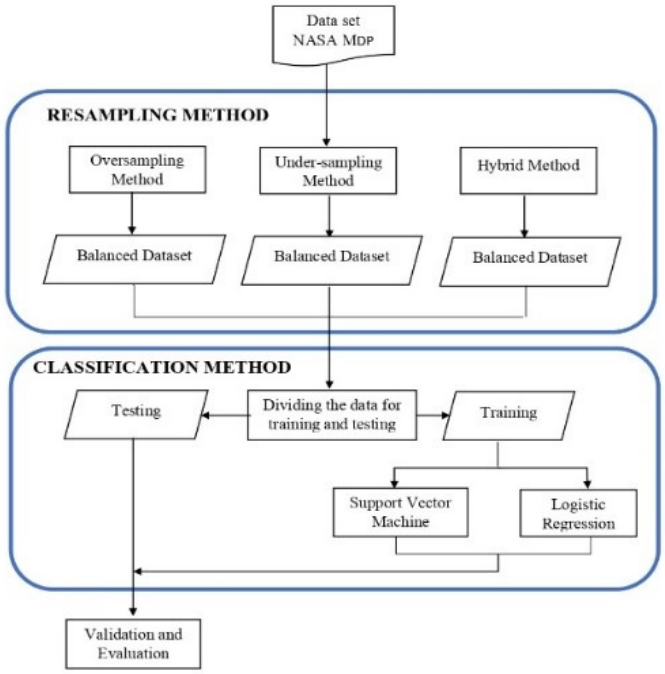

Fig. 1. The proposed method for optimization of software defect prediction

A. Collect Dataset

In this study, the data set used for software defects prediction is the NASA MDP repository data set. The NASA MDP data sets are a software matrix which is a data set that researchers in Software Engineering research commonly use, especially for research topics of software defects and software failures [2], [27]. The NASA MDP data sets consist of thirteen datasets which can be downloaded at http://promise.site.uottawa.ca/SERepository/datasets-page.html. Still, in this study, only five NASA MDP data sets with the 37 attributes were used to test for the proposed software flaw prediction model, as shown in Table 1.2

Table 1. Description of nasa MDP dataset

Dataset	Description	Sum of module	Sum of module defect	Precentage of defect
CM1	Spaceship instrument	327	42	12.84%
MW1	Storage management for basic data	253	27	10.67%
PC1	Zero gravity experiments are related to combustion	705	61	8.65%
PC3	Flight software from earth satellite orbits	1077	134	12.44%
PC4	Flight software from earth satellite orbits	1287	177	13.75%

B. Resampling Method

The resampling method is the first step of the proposed software defect prediction method, which aims to improve the distribution of data classes. In this step, the test data set will be resampled using three methods. The first is the oversampling method, which works by adding data to the minority class. This method is done by duplicating the minority class and balancing the classes randomly. Increasing the minority class is expected to improve the ability of the classification algorithm to be better because it can recognize the minority class sample from the majority sample.

The second method is the under-sampling method, which reduces most of the data in the majority class by removing noise and excess constraints from the test data set. The trick is to calculate the difference between the majority and minority classes. During the iteration process, the majority class is deleted so that the same number is obtained as the minority class. Furthermore, it is repeated as much as the difference between the majority and minority classes. This step is considered safe because removing excess noise data will not add any information about the majority class. Likewise, if the noise data is in the majority class, it will not significantly affect the information set [29].

The third method is a hybrid method that combines oversampling and under-sampling techniques. This method balances the data set by not removing important information from the data set or overfitting [30].

C. Support Vector Machine (SVM)

The SVM method separates data from its class by constructing a hyperplane. A good hyperplane is not

only a hyperplane that can be used to separate data but has the most significant margin. The simple idea of the SVM method is to maximize margins.

Each training data is represented by (x_i, y_i) where $i = 1,2,\cdots,n$, and x_i is a feature set for training data to-i. The class label is declared as $y_i \in \{-1,+1\}$. The SVM linear classification hyperplane is denoted in (1).

$$w.x_i + b = 0 \quad (1)$$

w and b are model parameters and $w.x_i$ is the inner product between w and x_i. The x_i data that enters the y_i class is the data that satisfies equation (2).

$$y_i = \begin{cases} -1 & if\ w.x_i + b \le -1 \\ +1 & if\ w.x_i + b \ge +1 \end{cases} \quad (2)$$

x_i is the training data set, w is the support vector weight value that is perpendicular to the hyperplane, b is bias value, y_i is label class from x_i, and $1 = 1,2,\cdots,n$. To calculate the value of b is used (3).

$$b = -\frac{1}{2}(w.x^+ + w.w^-) \quad (3)$$

$w.x^+$ is the weight value for the positive class, and $w.x^-$ is the weight value for the negative class. The value of w is obtained from (4).

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \quad (4)$$

α_i is data weight value to-i, y_i is class data to-i, and x_i is data to-i.

Hyperplane class support $+$ has to function $w.x_i + b = +1$ symbolized to be H_1. The formula for calculating the margin value is declared in (5).

$$Margin = |dH_1 - dH_2| = \frac{2}{\|w\|} \quad (5)$$

dH_1 is range hyperplane class support $+1$, dH_2 is range hyperplane class support -1, and $\|w\|$ is the weight vector w. The optimal margin calculated by maximizing distance between hyperplane and nearest data by minimizing the equation inverse (5) or minimizing $\frac{1}{2}\|w\|^2$.

D. Logistic Regression (LR)

This LR model is used to see the probability of an event and compare the risk of an event occurring by considering the factors that influence it [31]. LR is part of the regression analysis used when the dependent variable (response) is dichotomous. The dichotomous variable in this study consists of two values that represent software defects or not software defects [32]. This variable can also be illustrated as binary data, and the value is represented by 0 (no software defect) and 1 (software defect). The general form of LR can be seen in (6).

$$Y = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_p x_{pi} \quad (6)$$

Y is non-biner metric, β_0 is constant, and β_j is coefficient of parameter with $j = 1,2,\cdots,j$. The coefficients for the independent variables are estimated using the logit value as the dependent measure. Each predicted value can be converted into a probability between 0 and 1, as shown in (7).

$$Logit_i = ln\left(\frac{\pi(x_i)}{i-\pi(x_i)}\right) \quad (7)$$

E. Performance Measures

The next step is to measure the performance of the proposed method through the validation and evaluation stages. Validation and evaluation of experimental results is a measuring tool to determine how well the proposed method is compared to the methods studied previously. In addition, the results of validation and evaluation can be used to determine whether there is a significant difference between the proposed method and another method. The performance of the proposed method is analyzed and evaluated through various measures generated from the confusion matrix for a binary class (shown in Table 2.).

Table 2. Confusion matrix for a binary class

Class	Predicted as Positive	Predicted as Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

Diagnostic testing of the classification performance in this study used the Area Under Curve (AUC). AUC is a curve that describes probability with sensitivity and specificity variables with a limit value between 0 to 1. AUC is presented in (8) [30].

$$AUC = \frac{Sensitivity + Specificity}{2} \quad (8)$$

The calculation of the sensitivity value uses (9), while the measure of the specificity value uses (10).

$$Sensitivity = TP_{Rate} = \frac{TP}{TP+FN} \quad (9)$$

$$Specificity = TN_{Rate} = \frac{TN}{FP+TN} \quad (10)$$

Accuracy is defined as the level of closeness between the predicted value and the actual value. The accuracy value is calculated by taking the correct predictive percentage of the overall data. The calculation of the accuracy value uses (11) [2].

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (11)$$

Accuracy classification of diagnostic tests using AUC can be described in Table 3. [33]

Table 3. Accuracy classification of diagnostic tests using AUC

AUC value	Classification
0.90 - 1.00	Excellent classification
0.80 - 0.90	Good classification
0.70 - 0.80	Fair classification
0.60 - 0.70	Poor classification
< 0.60	Failure

The performance of the proposed model can be measured using the t-test. The t-test is widely used to determine whether there is an influence between the dependent variable and the independent variable [33]. The t-test was conducted to determine whether the

combination of resampling methods with different classification methods in an imbalanced class gave significant results. The decision t-test can be with two events: comparing P_{Value} with α or t_{Value} with t_{table}. The hypothesis in the t-test of two independent samples is the null hypothesis (H_0) and the alternative hypothesis (H_1). The purpose of the t-test is to determine which of the two assumptions below are true:

- H_0 : experimental value gave no significant results between the combination method of resampling and the SVM algorithm compared to the combination method of resampling and LR algorithms in an imbalanced class.
- H_1 : experimental value gave significant results between the combination method of resampling and the SVM algorithm compared to the combination method of resampling and LR algorithms in an imbalanced class.

The decision is reached based on experimental evidence which rejects or accepts H_0. Hypothesis testing in this paper uses a significant level value of $\alpha = 0.05$. The hypothesis used is as follows:

- Based on the significance value P_{Value}, if the value is $P_{Value} < \alpha$, then reject H_0, or if $P_{Value} > \alpha$, then accept H_0.
- Based on the value of t_{count} and t_{table}, if the value of $t_{count} > t_{table}$, then reject H_0, or if the value of $t_{count} < t_{tables}$, then accept H_0.

III. RESULTS

This paper discusses the prediction of software defects when an imbalanced class problem is found in the data set. The test in this paper uses five NASA MPD data sets that have the same number attributes of 37 (CM1, MW1, PC1, PC3, PC4). The method tested is the model's performance combining resampling methods: oversampling, undersampling, and Hybrid, followed by the SVM and LR classification methods.

In the first step, a t-test will be conducted to determine whether the data processing result using the combination resampling method and the SVM algorithm compared to the combination resampling method and the LR algorithm in an imbalanced class gives significant results. The second step tests the validation of the proposed model for software defects prediction based on the value of accuracy and AUC. Finally, the third step compares the performance of each proposed model to get the best combination of software defect prediction models. This section will also compare the performance of the best variety of resampling and classification methods with the models proposed in other studies.

A. T-test Results

The evaluation process of the proposed model is carried out using a t-test, which aims to determine whether data processing using a combination of resampling methods with SVM and LR algorithms gives significant results. The results of the t-test calculations are presented in Table 4.

Table 4. Results calculations t-test

Paired t-test	
df	4
t_{count}	3.1254
t_{table}	2.7765
P_{Value}	0.0344
α	0.05

Based on Table 4, the t-test results show that the $P_{Value} < \alpha = 0.0344 < 0.05$ and the $t_{count} > t_{table} = 3.1254 > 2.7765$. Based on a valid hypothesis, the conclusion that H_0 is rejected, which means the experimental value gave significant results between the combination method of resampling and the SVM algorithm compared to the combination method of resampling and LR algorithms in an imbalanced class.

B. Analysis Combination Method of Resampling and SVM Algorithm

The results of the accuracy calculation using a combination of oversampling, under-sampling, or Hybrid methods with the SVM algorithm are shown in Table 5.

Table 5. Comparison of accuracy value for combination method resampling and SVM algorithm

Data Set	Oversampling	Under-sampling	Hybrid
CM1	0.8403	0.5294	0.7246
MW1	0.8261	0.5000	0.7925
PC1	0.8535	0.8000	0.8497
PC3	0.8140	0.7091	0.7965
PC4	0.8669	0.8841	0.8541
Average	**0.8402**	**0.6845**	**0.8035**

Table 5 shows that of the five data sets, four data sets have high accuracy when using a combination of oversampling and SVM methods. The average accuracy results are shown in Table 5.; the combination method of oversampling and SVM algorithm has the highest accuracy of 84.02% because the average accuracy > 80% proves that the combination of the proposed methods is high accuracy for software defects prediction in imbalanced class.

The results of the AUC calculation using a combination of oversampling, under-sampling, or Hybrid methods with the SVM algorithm are shown in Table 6.

Table 6. Comparison of AUC value for combination method resampling and SVM algorithm

Data Set	Oversampling	Under-sampling	Hybrid
CM1	0.9226	0.6250	0.8488

Data Set	Oversampling	Under-sampling	Hybrid
MW1	0.9001	0.5000	0.8960
PC1	0.9322	0.7692	0.9333
PC3	0.9041	0.7831	0.8812
PC4	0.9233	0.9167	0.9398
Average	**0.9165**	**0.7188**	**0.8998**

Table 6 shows that the AUC value of the oversampling and SVM combination of the five data sets, three data sets have the highest AUC value. The average result of the highest AUC value using the oversampling and SVM methods is 91.65%. Based on the average AUC value of the combination of oversampling and SVM methods > 90%, it is proven that the proposed combination of the model has excellent performance for software defects prediction in an imbalanced class.

C. Analysis Combination Method of Resampling and LR Algorithm

The results of the accuracy calculation using a combination of oversampling, under-sampling, or Hybrid methods with the LR algorithm are shown in Table 7.

Table 7. Comparison of accuracy value for combination method resampling and LR algorithm

Data Set	Oversampling	Under-sampling	Hybrid
CM1	0.8571	0.4118	0.7826
MW1	0.7717	0.5833	0.6604
PC1	0.8498	0.6400	0.8431
PC3	0.7829	0.7091	0.8009
PC4	0.8254	0.8551	0.8541
Average	**0.8174**	**0.6399**	**0.7882**

Table 7 shows that of the five data sets, three data sets have high accuracy when using a combination of oversampling and LR methods. The average accuracy results are shown in Table 7; the combination method of oversampling and the LR algorithm have the highest accuracy of 81.74%. The average accuracy>80% proves that the combination of the proposed methods is high accuracy for software defects prediction in imbalanced class.

The results of the AUC calculation using a combination of oversampling, under-sampling, or Hybrid methods with the LR algorithm are shown in Table 8.

Table 8. Comparison of AUC Value for Combination Method Resampling and LR Algorithm

Data Set	Oversampling	Under-sampling	Hybrid
CM1	0.8886	0.4028	0.7965
MW1	0.8400	0.6667	0.6624
PC1	0.9084	0.6346	0.9109
PC3	0.8637	0.7910	0.8412
PC4	0.9230	0.9461	0.9364
Average	**0.8847**	**0.6882**	**0.8295**

Table 8 shows that the AUC value of the oversampling and LR combination of the five data sets, three data sets have the highest AUC value. The average result of the highest AUC value using the oversampling and LR methods is 88.47%. Based on the average AUC value of the combination of oversampling and SVM methods > 80%, it is proven that the proposed combination of the model has good performance for software defects prediction with an imbalanced class.

DISCUSSION

The results of testing the AUC values in Table 7 and Table 8 show the proposed best software defect prediction in an imbalanced class method using a combination of oversampling methods for the resampling process. In addition, using the resampling method before classification proses using the SVM and LR algorithms, on average, increases classification performance, as shown in Table 9.

Table 9. Improved Classification Performance using The Resampling Process

Data Set	Increase in AUC (%)	
	LR	SVM
CM1	11.08	35.43
MW1	5.18	16.75
PC1	12.32	16.05
PC3	4.75	12.33
PC4	3.50	5.43
Average	**7.36**	**17.19**

The average increase in AUC values in Table 9 shows that using the resampling method can improve the classification performance for software defect prediction. The most significant increase in the AUC value in the SVM algorithm is 17.19% compared to the LR algorithm at 7.26, which proves the resampling process works very well to improve the performance of the SVM algorithm to software defects prediction in imbalanced class.

Table 5 and Table 7 show that combining the oversampling method with both classification algorithms dominates as the best resampling method in dealing with imbalanced class. The results of comparing the accuracy of the oversampling process combined with the SVM and LR algorithms are presented in Fig. 2.

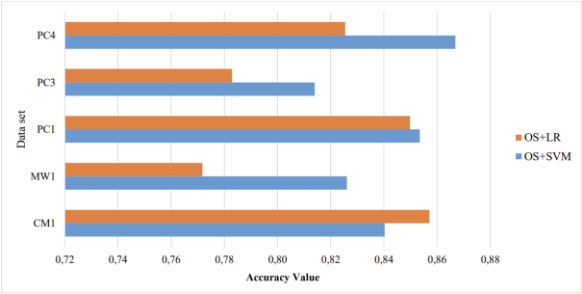

Fig.2. Comparison of accuracy value chart the combination oversampling (OS) with SVM and LR algorithms

Fig.2 shows the majority of accuracy value for the best software defects predicted using a combination of oversampling and SVM algorithms. The average accuracy of the oversampling combination and SVM methods is 2.28% higher than the oversampling and LR methods.

Table 6 and Table 8 show that combining the oversampling method with both classification algorithms dominates as the best resampling method in dealing with imbalanced classes. Fig.3 shows the comparison of the AUC value of the combination method oversampling with the SVM and LR algorithms. Fig.3. shows all of the AUC values for the best software defects prediction using a combination of oversampling and SVM algorithms. The average AUC of the combination of oversampling and SVM methods is 3.17% higher than the oversampling and LR methods.

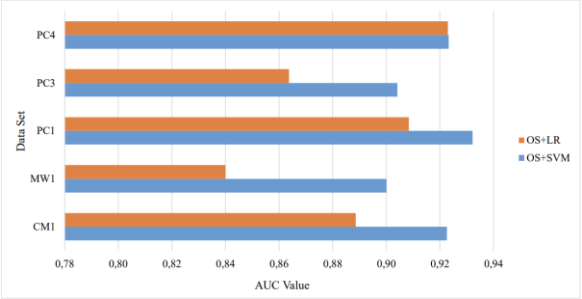

Fig. 3. Comparison of AUC value chart the combination Oversampling (OS) using SVM and LR algorithms

The best-combined performance of the proposed method is a combination of oversampling with SVM for software defects prediction with an imbalanced class. Table 9. shows a comparison of the performance of the best model proposed in this paper with the model proposed by [11], [15] to predict software defects. Research conducted by [11] proposed a combination method of traditional Artificial Neural Network (ANN) and a new Artificial Bee Colony (ABC) algorithm. In addition, an artificial Neuro-Fuzzy Inference system as a basic classifier using SMOTE technique to balance the dataset was proposed by [15].

Table 10. Comparison of the AUC values of the proposed method with other studies

Data Set	Proposed Method	[11]	[15]
CM1	0.9226	0.7700	0.7400
MW1	0.9001	-	-
PC1	0.9322	0.8200	0.7300
PC3	0.9041	-	-
PC4	0.9233	-	-
Average	**0.9165**	**0.7950**	**0.7350**

Table 10 shows the results of the AUC test using the same data set, indicating that the proposed method is on average 7.52% higher than the proposed method [11], [15]. This analysis proves that the combination of the oversampling method with SVM has a better classification performance than the proposed method [11], [15] for software defects prediction in imbalanced class. The overall performance of the combined oversampling method with SVM provides high accuracy and is excellent at software defects prediction in an imbalanced class

CONCLUSION

This study proposed a combination resampling method with a classification algorithm to software defects prediction in an imbalanced class. Testing the proposed model uses five data sets from 13 data sets from the NASA MDP data set, with the same attributes and have dichotomous characteristics. The performance evaluation of the proposed method uses a t-test, the AUC measure, and the Accuracy value. The t-test shows the $P_{Value} < \alpha = 0.0344 < 0.05$ and the $t_{count} > t_{table} = 3.1524 > 2.7765$, which proves experimental value gave significant results between the combination method resampling and the SVM algorithm compared to the combination method of resampling and LR algorithms in an imbalanced class. The performance of the classification algorithm has also improved with the use of the resampling process, the average increase in AUC values in the SVM algorithm is 17.19%, and the LR algorithm is at 7.26% compared to without the resampling process. Combining the oversampling method with the SVM algorithm is more accurate for software defects prediction with an average accuracy value of 84.02%, which is 2.3% higher than the LR classification algorithm. The average AUC value for combining the oversampling method with the SVM algorithm is > 90%, which is 91.65%, indicating that the proposed model is excellent at software defects prediction for imbalanced class compared to the LR classification algorithm.

REFERENCES

[1] A. S. Andreou dan S. P. Chatzis, "Software defect prediction using doubly stochastic Poisson processes driven by stochastic belief networks," *J. Syst. Softw.*, vol. 122, hal. 72–82, Des 2016, doi:

10.1016/j.jss.2016.09.001.

[2] A. Iqbal *et al.*, "Performance Analysis of Machine Learning Techniques on Software Defect Prediction using NASA Datasets," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 5, hal. 300–308, 2019, doi: 10.14569/IJACSA.2019.0100538.

[3] M. A. Memon, M.-U.-R. Magsi, M. Memon, dan S. Hyder, "Defects Prediction and Prevention Approaches for Quality Software Development," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 8, hal. 451–457, 2018, doi: 10.14569/IJACSA.2018.090857.

[4] D. Bowes, T. Hall, dan J. Petrić, "Software defect prediction: do different classifiers find the same defects," *Softw. Qual. J.*, vol. 26, no. 2, hal. 525–552, Jun 2018, doi: 10.1007/s11219-016-9353-3.

[5] Y. Shao, B. Liu, S. Wang, dan G. Li, "A novel software defect prediction based on atomic class-association rule mining," *Expert Syst. Appl.*, vol. 114, hal. 237–254, Des 2018, doi: 10.1016/j.eswa.2018.07.042.

[6] X. Jing, F. Wu, X. Dong, dan B. Xu, "An Improved SDA Based Defect Prediction Framework for Both Within-Project and Cross-Project Class-Imbalance Problems," *IEEE Trans. Softw. Eng.*, vol. 43, no. 4, hal. 321–339, Apr 2017, doi: 10.1109/TSE.2016.2597849.

[7] R. S. Wahono, "A Systematic Literature Review of Software Defect Prediction : Research Trends , Datasets , Methods and Frameworks," *J. Softw. Eng.*, vol. 1, no. 1, hal. 1–16, 2015.

[8] N. Gayatri, S. Nickolas, dan A. V Reddy, "Feature Selection Using Decision Tree Induction in Class level Metrics Dataset for Software Defect Predictions," in *Proceedings of the World Congress on Engineering and Computer Science (WCECS) 2010*, 2010, vol. I, hal. 124–129.

[9] Y. Peng, G. Wang, dan H. Wang, "User preferences based software defect detection algorithms selection using MCDM," *Inf. Sci. (Ny).*, vol. 191, hal. 3–13, 2012, doi: 10.1016/j.ins.2010.04.019.

[10] Z. Sun, Q. Song, dan X. Zhu, "Using Coding Based Ensemble Learning to Improve Software Defect Prediction," *IEEE Trans. Syst. Man, Cybern. Part C (Applications Rev.*, vol. 42, no. 6, hal. 1806–1817, 2012, doi: 10.1109/TSMCC.2012.2226152.

[11] Ö. F. Arar dan K. Ayan, "Software defect prediction using cost-sensitive neural network," *Appl. Soft Comput.*, vol. 33, hal. 263–277, Agu 2015, doi: 10.1016/j.asoc.2015.04.045.

[12] G. Fan, X. Diao, H. Yu, K. Yang, dan L. Chen, "Software Defect Prediction via Attention-Based Recurrent Neural Network," *Sci. Program.*, vol. 2019, hal. 1–14, Apr 2019, doi: 10.1155/2019/6230953.

[13] B. Turhan, G. Kocak, dan A. Bener, "Data mining source code for locating software bugs: A case study in telecommunication industry," *Expert Syst. Appl.*, vol. 36, no. 6, hal. 9986–9990, Agu 2009, doi: 10.1016/j.eswa.2008.12.028.

[14] R. Batuwita dan V. Palade, "FSVM-CIL : Fuzzy Support Vector Machines for Class Imbalance Learning," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 3, hal. 558–571, 2010, doi: 10.1109/TFUZZ.2010.2042721.

[15] S. S. Maddipati dan M. Srinivas, "An Hybrid Approach for Cost Effective Prediction of Software Defects," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 2, hal. 145–152, 2021, doi: 10.14569/IJACSA.2021.0120219.

[16] K. Sahu dan R. K. Srivastava, "Soft computing approach for prediction of software reliability," *ICIC Express Lett.*, no. March 2019, 2021, doi: 10.24507/icicel.12.12.1213.

[17] A. R. P. Periasamy dan A. Mishbahulhuda, "Data Mining Techniques in Software Defect Prediction," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 7, no. 3, hal. 301–303, Mar 2017, doi: 10.23956/ijarcsse/V7I3/0173.

[18] G. Denaro, "Estimating software fault-proneness for tuning testing activities," in *Proceedings of the 22nd international conference on Software engineering - ICSE '00*, 2000, hal. 704–706, doi: 10.1145/337180.337592.

[19] R. Shatnawi dan W. Li, "An Empirical Investigation of Predicting Fault Count, Fix Cost and Effort Using Software Metrics," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 2, hal. 484–491, 2016, doi: 10.14569/IJACSA.2016.070264.

[20] D. Gray, D. Bowes, N. Davey, Y. Sun, dan B. Christianson, "Software defect prediction using static code metrics underestimates defect-proneness," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, Jul 2010, hal. 1–7, doi: 10.1109/IJCNN.2010.5596650.

[21] Haibo He dan E. A. Garcia, "Learning from Imbalanced Data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, hal. 1263–1284, Sep 2009, doi: 10.1109/TKDE.2008.239.

[22] T. M. Khoshgoftaar, K. Gao, dan N. Seliya, "Attribute Selection and Imbalanced Data: Problems in Software Defect Prediction," in *2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, Okt 2010, vol. 1, hal. 137–144, doi: 10.1109/ICTAI.2010.27.

[23] K. Teh, P. Armitage, S. Tesfaye, D. Selvarajah, dan I. D. Wilkinson, "Imbalanced learning: Improving classification of diabetic neuropathy from magnetic resonance imaging," *PLoS One*, vol. 15, no. 12, hal. 1–15, Des 2020, doi: 10.1371/journal.pone.0243907.

[24] X. Sheng, Z. Junhai, W. Xiaolan, dan Y. Ming, "A new resampling method of imbalanced large data based on class boundary," in *2015 International Conference on Machine Learning and Cybernetics (ICMLC)*, Jul 2015, vol. 2, hal. 826–831, doi: 10.1109/ICMLC.2015.7340660.

[25] D. Zhang, W. Liu, X. Gong, dan H. Jin, "A novel improved SMOTE resampling algorithm based on fractal," *J. Comput. Inf. Syst.*, vol. 7, no. 6, hal. 2204–2211, 2011.

[26] F. Charte, A. J. Rivera, J. María, dan F. Herrera, "Knowledge-Based Systems MLSMOTE : Approaching imbalanced multilabel learning through synthetic instance generation," *KNOWLEDGE-BASED Syst.*, vol. 89, hal. 385–397, 2015, doi: 10.1016/j.knosys.2015.07.019.

[27] T. Hall, S. Beecham, D. Bowes, D. Gray, dan S. Counsell, "A Systematic Literature Review on Fault Prediction Performance in Software Engineering," *IEEE Trans. Softw. Eng.*, vol. 38, no. 6, hal. 1276–1304, Nov 2012, doi: 10.1109/TSE.2011.103.

[28] F. Cheng, G. Fu, X. Zhang, dan J. Qiu, "Multi-

objective evolutionary algorithm for optimizing the partial area under the ROC curve," *Knowledge-Based Syst.*, vol. 170, hal. 61–69, Apr 2019, doi: 10.1016/j.knosys.2019.01.029.

[29] M. Bach, A. Werner, J. Żywiec, dan W. Pluskiewicz, "The study of under- and over-sampling methods' utility in analysis of highly imbalanced data on osteoporosis," *Inf. Sci. (Ny).*, vol. 384, hal. 174–190, Apr 2017, doi: 10.1016/j.ins.2016.09.038.

[30] U. R. Salunkhe dan S. N. Mali, "Classifier Ensemble Design for Imbalanced Data Classification: A Hybrid Approach," *Procedia Comput. Sci.*, vol. 85, hal. 725–732, 2016, doi: 10.1016/j.procs.2016.05.259.

[31] G. K. Armah, G. Luo, K. Qin, dan A. S. Mbandu, "Applying Variant Variable Regularized Logistic Regression for Modeling Software Defect Predictor," *Lect. Notes Softw. Eng.*, vol. 4, no. 2, hal. 107–115, Mei 2016, doi: 10.7763/LNSE.2016.V4.234.

[32] K. Ghazvini, M. Yousefi, F. Firoozeh, dan S. Mansouri, "Predictors of tuberculosis: Application of a logistic regression model," *Gene Reports*, vol. 17, hal. 1–4, Des 2019, doi: 10.1016/j.genrep.2019.100527.

[33] F. Gorunescu, *Data Mining: Concepts, models and techniques*, vol. 12. Springer-Verlag Berlin Heidelberg, 2011.