Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control
Journal homepage: http://kinetik.umm.ac.id
ISSN: 2503-2267
Vol. 9, No. 4, November, Pp. 389-396

Improving software defect prediction using a combination of ant colony optimization-based feature selection and ensemble technique

Windi Eka Yulia Retnani [*1], Muhammad 'Ariful Furqon[2], Juni Setiawan[1]
Information Technology Department, Universitas Jember, Indonesia[1]
Informatics Department, Universitas Jember, Indonesia[2]

Article Info
Keywords:
Software Defect, Ant Colony Optimization, SMOTE, Ensemble

Article history:
Received: June 07, 2024
Accepted: August 01, 2024
Published: November 01, 2024

Cite:
W. E. Y. Retnani, M. 'Ariful . Furqon, and J. Setiawan, "Improving Software Defect Prediction Using a Combination of Ant Colony Optimization-based Feature Selection and Ensemble Technique", *KINETIK*, vol. 9, no. 4, Nov. 2024.
https://doi.org/10.22219/kinetik.v9i4.2038

*Corresponding author.
Windi Eka Yulia Retnani
E-mail address:
windi.ilkom@unej.ac.id

Abstract
Software defect prediction plays a vital role in enhancing software quality and minimizing maintenance costs. This study aims to improve software defect prediction by employing a combination of Ant Colony Optimization (ACO) for feature selection and ensemble techniques, particularly Gradient Boosting. This research utilized three NASA MDP datasets: MC1, KC1, and PC2, to evaluate the performance of four machine learning algorithms: Random Forest, Support Vector Machine (SVM), Decision Tree, and Naïve Bayes. The data preprocessing comprised handling class imbalance using SMOTE and converting categorical data into numerical representations. The results indicate that the integration of ACO and Gradient Boosting significantly enhances the accuracy of all four algorithms. Notably, the Random Forest algorithm achieved the highest accuracy of 99% on the MC1 dataset. The findings suggest that combining ACO-based feature selection with ensemble techniques can effectively boost the performance of software defect prediction models, offering a robust approach for early detection of potential software defects and contributing to improved software reliability and efficiency.

1. Introduction

According to the National Institute of Standards and Technology, system losses in software defects amount to $60 billion. The cost of repairing software defects during design stage can reach 60 times the software development cost, and 100 times if the software defects are found after the release [1]. In general, software defect prediction is also a cost-sensitive problem [2]. Predictions on software defects are carried out using data sets from previous software [3]. The NASA dataset is one of the publicly available datasets, which is very popular and has been widely used in developing software defect prediction models, because obtaining NASA dataset is very easy [4]. The NASA dataset shows that there is more non-defective data than defective data, causing data imbalance and noise. This is a problem frequently found in software defect predictions, so that the prediction results tend to produce many classes, namely no defect class. To overcome this problem, it is necessary to modify the prediction technique by adding other techniques or combining other algorithms [5].

Software quality can be improved in various ways, but the most effective method is to prevent defects by predicting the probability of defects [6]. The most commonly used method for predicting software defects is the Prediction method [5]. The majority of current software defect predictions use supervised machine learning algorithms to predict defects in software applications [7].

Four previous studies discussed how machine learning algorithms work in predicting software defects in NASA datasets. This research uses an ensemble technique classified as a boosting method, namely Gradient boosting, because it takes shorter time and has good performance [8], and confirms that Support Vector Machine algorithm can be used to predict software defects in NASA dataset [9]. A study [10] found that Random Forest algorithm has a good accuracy in measuring model performance. Apart from that, [11] stated that one of the feature selection techniques, namely Ant Colony Optimization, can be used to select good features to improve the performance of the algorithm model. Ant Colony Optimization utilizes a probability-based decision-making process, in which each ant constructs a solution by iteratively selecting the next step based on the combination of the desirability of the solution and the amount of pheromone present[12]. Meanwhile, according to research by [2] the Naïve Bayes algorithm and Decision Tree are the algorithms that can be used to compare several other algorithms.

Support Vector machine is a binary classification of datasets, where the decision boundaries to solve the learning sample is the maximum margin hyperplane in svm, which is very good in terms of robustness [13]. Different from other algorithm that usually find the best single line between classes to classify the data [14], SVM separates the classes by using three separating lines, one for main separating line and two other lines are the support line [15]. Decision Tree is a machine learning algorithm with the simplest modeling techniques to classify data by using graphs, such as trees that

show leaf structure and node [16]. Naive Bayes is a probability classification based on Bayes' theorem and considering the impact of attribute values on certain classes. The goal of Naïve Bayes is to simplify the calculations [17]. Random Forest is an ensemble bagging-based classifier as basic learner, this model divides the training data into several trees' decision [11]. Random Forest is a parallel ensemble Machine Learning classifying algorithm based on the bagging technique [18].

Different from the results of the previous researches, this research focused on comparing four machine learning algorithms with an innovative approach by combining Ant Colony Optimization-based feature selection and ensemble technique. While previous studies have highlighted the strengths of each algorithm in predicting software defects, a comprehensive comparison incorporating these advanced methods remains unexplored. By integrating Ant Colony Optimization for feature selection, this study can enhance the performance of each algorithm by selecting the most relevant features, thereby improving the prediction accuracy. Additionally, employing an ensemble technique, such as Gradient Boosting, can improve model performance by combining the strengths of each model and reducing the prediction errors. This approach can also address data imbalance issues, frequently overlooked in current research, by ensuring that the ensemble method can effectively handle varied data distributions.

2. Research Method

This research employs the following methods to compare the Random Forest, Support Vector Machine (SVM), Decision Tree, and Naïve Bayes algorithms in predicting software defect using a combination of ensemble and Ant Colony Optimization. Figure 1 illustrates the research stages throughout the study.

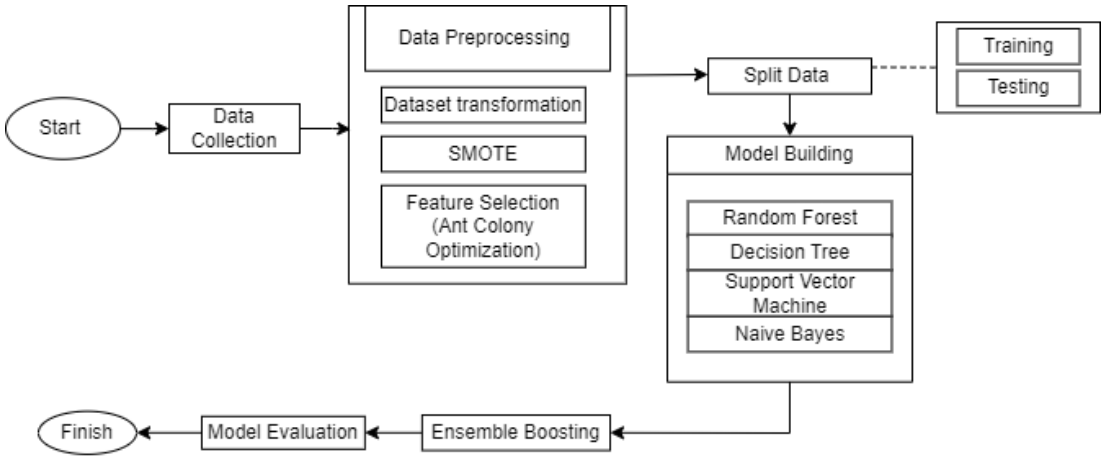


Figure 1. Research Stages

The data is obtained from the NASA MDP Software Defect Datasets website in arff format, which is a file that describes a list of data containing defect and no defect data [19]. In the data collection process, researchers used three NASA MDP (Metric Data Program) datasets, namely MC1, KC1 and PC2, because these data have better accuracy than several other datasets. The NASA MDP (Metric Data Program), MC1, KC1 and PC2 datasets, each has a different amount of data, namely the MC1 dataset has a total of 39 attributes with 2031 data, the KC1 dataset has a total of 22 attributes with 1183 data, the PC2 has a total of 36 attributes with 786 records. Detailed information regarding the dataset used in this study is provided in Table 1.

Table 1. MC1 Dataset Description

Loc_Blank	Call_Pairs	…	Number_of_Lines	Loc_Total	Defective
8.0	0.0	…	76.0	55.0	b'N'
2.0	0.0	…	20.0	16.0	b'N'
11.0	0.0	…	183.0	53.0	b'N'
0.0	0.0	…	7.0	5.0	b'N'
0.0	0.0	…	9.0	5.0	b'N'
…	…	…	…	…	…
5.3	2.4	…	17.6	20.8	b'N'

Data preprocessing techniques were employed to enhance the datasets quality and suitability for classification analysis. These techniques aimed to refine and standardize the dataset by addressing missing values and irrelevant attributes. The preprocessing steps included in-depth procedures to handle missing data, ensuring the dataset remained

robust and suitable for subsequent analysis [20]. The problem and dataset in question determine the technique used. One of which is oversampling techniques, such as SMOTE (Synthetic Minority Over-sampling Technique) [21]. SMOTE is able to reach better performance than the modified version in certain dataset and classification algorithm. Therefore, this research used SMOTE, which is the benchmark of oversampling method that has been successfully applied in many cases to improve the generalization performance of classifier in minority class. SMOTE even still has good performance when the number of the samples in the minority class is quite small [22]. Table 2 comprehensively explains the specific preprocessing techniques utilized in this study.

Table 2. Data Preprocessing Techniques

Preprocessing Technique	Description	Function
Data Transformation	Attribute labels were renamed, and their categorical format was converted into numeric representations as part of the data preprocessing stage.	LabelEncoder()
SMOTE	Handling class imbalance for improving performance to produce a good model in increasing accuracy.	Imblearn.over_sampling import SMOTE
Feature Selection (Ant Colony Optimization)	Selected best feature from several existing attributes, by determining the number of iteration	Num_ants

Data splitting is the process of separating a dataset into subsets for building a good machine learning model [23]. At this stage, testing data and training data were generated and tested in the next stage. At this stage, scikit-learn library 'train_test_split()' is used to divide the dataset into two subsets, namely testing and training [24]. Data splitting is needed to separate the dataset and produce training data and testing data that can be used in training machine learning models and validating models. As previously discussed, the dataset used this time has an imbalance in the type of classification results, where the classification of safe programs is more dominant than the classification of programs labeled as malware. Therefore, a data balancing method must be applied [25]

The model employed in this study included four machine-learning algorithms, namely Random Forest, Support Vector Machine, Decision Tree, and Naïve Bayes. These algorithms were selected based on a comprehensive review of prior studies, which highlighted their superior accuracy and diverse advantages. The model was constructed by utilizing functions accessible from the Sklearn library in Python, coupled with hyperparameter optimization techniques to enhance each classifier's performance [26]. Table 3 outlines the specific parameter scenarios employed in the model-building process.

Table 3. Parameter Scenarios

Classifier	Parameter
Random Forest	RandomForestClassifier()
Support Vector Machine	SVC(gamma="auto",kernel='linear',degree=3,C=1, random_state=99)
Decision Tree	DecisionTreeClassifier(max_depth=10,criterion="gini")
Naïve Bayes	GaussianNB()

Ensemble technique, specifically Gradient Boosting, is selected because it is a good technique for increasing the accuracy of four algorithms being used, namely Random Forest, Decision Tree, Support Vector Machine and Naïve Bayes. From several techniques, Gradient Boosting succeeded in obtaining good accuracy in every algorithm with good accuracy.

Model evaluation was used to measure the performance of the algorithms being used, consisting of confusion matrix, accuracy, precision and recall. This research also compared the results of the original algorithm with the combined algorithms, namely Ant Colony Optimization and ensemble technique, particularly Gradient Boosting.

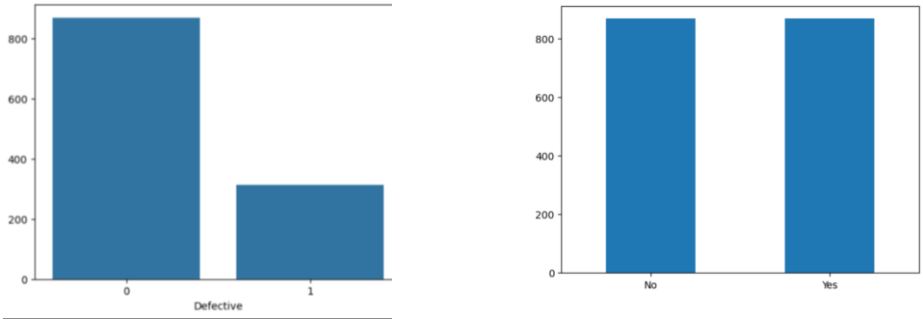
3. Results and Discussion

Before starting the model building phase using the NASA dataset, a comprehensive data preprocessing is necessary to optimize the suitability of datasets analysis. This preprocessing phase included several important steps to improve data integrity and facilitate practical model construction. First, a transformation stage was carried out on the categorical column, namely the Defective attribute, and it is converted into a numerical representation using the LabelEncoder() function from Sklearn. Table 4 provides an overview of the post-transformation dataset, illustrating the results of this preprocessing procedure.

Table 4. Data Transformation Results

Loc_Blank	Branch_Count	…	Num_Unique_Operators	Loc_Total	Defective
9.0	11.0	…	21.0	41.0	1
0.0	1.0	…	6.0	7.0	0
18.0	31.0	…	28.0	184.0	1
1.0	5.0	…	12.0	23.0	1
5.0	17.0	…	15.0	96.0	1
…	…	…	…	…	…
2.0	5.0	…	12.0	9.0	1

This stage used SMOTE (Synthetic Minority Over-Sampling Technique) as a resampling technique commonly used in machine learning to handle class imbalance problems in datasets. SMOTE is useful for improving performance so as to produce a good model that increases the accuracy. SMOTE handles class imbalance, prevents overfitting, and improves model performance. Figure 2 shows the results of one of the datasets before and after SMOTE.



a b

Figure 2. SMOTE Results (a) before SMOTE; (b) after SMOTE

This stage is the process of selecting the most relevant features from the dataset or features that will be used in building a prediction model which aims to improve model performance by reducing unused features. The Ant Colony Optimization (ACO) algorithm begins with initializing variables that are important for the next process, such as the number of ants (num_ants), as well as variables to track the best features and best accuracy. Figure 3 shows an example of a flowchart in the feature selection process, and Table 5 shows the results of selecting the best features from each dataset.

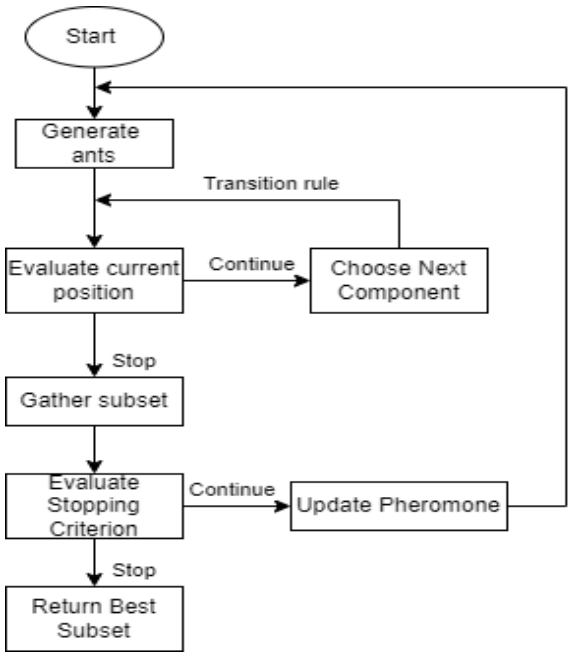


Figure 3. Flowchart ACO Process

Table 5. Result the Best feature

No	Dataset	Best Feature
1	KC1	Halstead_effort
		Halstead_content
		Halstead_length
		Halstead_difficulty
		Num_unique_operands
		Halstead_volume
		Loc_comments
		Loc_blank
		Branch_count
		Halstead_error_est
2	MC1	Call_pairs
		Num_operands
		Normalized_cylomatic_complexity
		Num_operators
		Branch_count
		Halstead_effort
		Halstead_prog_time
		Global_data_complexity
		Halstead_level
		Condition_count
3	PC2	Multiple_condition_count
		Edge_count
		Halstead_volume
		Parameter_count
		Halstead_effort
		Loc_executable
		Halstead_error_est
		Condition_count
		Num_unique_operators
		Cyclomatic_density

This study combined four different algorithms, namely Random Forest, Support Vector Machines, Decision Trees and Naive Bayes. This study employed various evaluation methods to assess the algorithm performance, including confusion matrix, accuracy, precision, and recall.

At this stage, ensemble technique was used, namely Gradient Boosting, because it is a good technique for increasing the accuracy of the four algorithms being used, namely Random Forest, Decision Tree, Support Vector Machine and Naïve Bayes. From several techniques, Gradient Boosting succeeded in obtaining good accuracy in every algorithm with good accuracy. Figure 4 shows an example diagram of the ensemble technique process using gradient boosting. The results of the gradient boosting can be seen in Table 6.

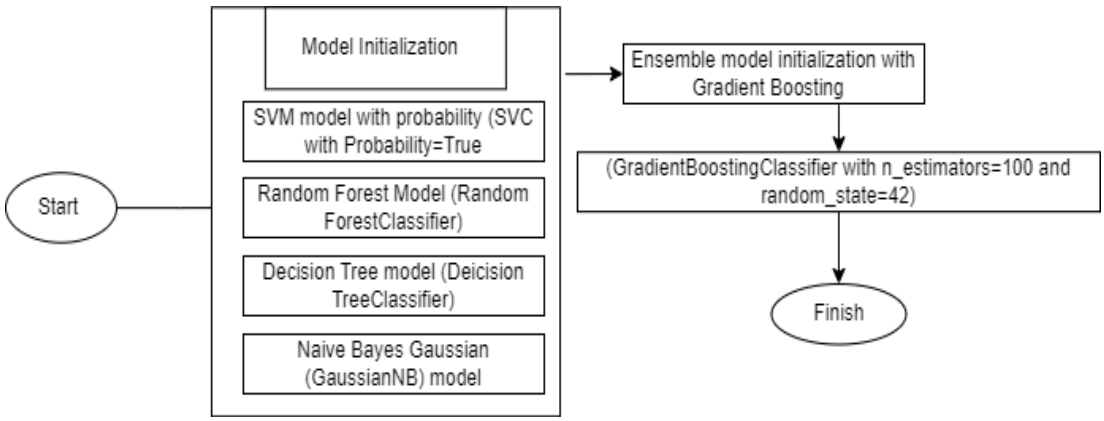


Figure 4. Ensemble Technique Process

Table 6. Ensemble Technique Results

No	Algorithm	Dataset	Accuracy
1	Support Vector Machine	KC1	0,65
		MC1	0,88
		PC2	0,91
2	Random Forest	KC1	0,84
		MC1	0,99
		PC2	0,97
3	Decision Tree	KC1	0,76
		MC1	0,98
		PC2	0,97
4	Naïve Bayes	KC1	0,59
		MC1	0,72
		PC2	0,83

The final stage of this research was the model evaluation stage, where the model evaluation was used to measure the performance of the algorithms being used, namely confusion matrix, accuracy, precision and recall. This research also compared the results of the original algorithm with the combined algorithms, namely Ant Colony Optimization and ensemble technique, particularly Gradient Boosting. Table 7 shows the comparative results of the model evaluation.

Table 7. Comparative Model Evaluation Results

Algorithm	Dataset	Model Evaluation					
		Original algorithm			Combined algorithms		
		Accuracy	Precision	Recall	Accuracy	Precision	Recall
Decision Tree	KC1	0,73	0,73	0,69	0,76	0,78	0,73
	MC1	0,94	0,91	0,97	0,98	0,97	0,98
	PC2	0,95	0,94	0,97	0,96	0,97	0,96
Random Forest	KC1	0,83	0,85	0,78	0,84	0,85	0,82
	MC1	0,99	0,98	0,99	0,99	0,98	1,00
	PC2	0,98	0,97	0,99	0,97	0,97	0,98
Support Vector Machine	KC1	0,61	0,64	0,44	0,65	0,75	0,47
	MC1	0,79	0,77	0,81	0,88	0,84	0,93
	PC2	0,85	0,78	1,00	0,91	0,84	1,00
Naïve Bayes	KC1	0,60	0,67	0,35	0,59	0,71	0,31
	MC1	0,62	0,73	0,32	0,72	0,68	0,81
	PC2	0,81	0,84	0,77	0,83	0,83	0,83

From Table 7, it can be concluded that the four algorithms obtained good accuracy after being combined with Ant Colony Optimization and ensemble technique. By using Ant Colony Optimization and ensemble techniques, the four algorithms showed a significant increase in accuracy. Random Forest algorithm achieved the highest accuracy of 0.99 on the MC1 dataset both before and after the combination, showing its stable performance. After the combination, Decision Tree also showed a significant increase in accuracy, from 0.94 to 0.98 on dataset MC1, and from 0.95 to 0.96 on dataset PC2. The Support Vector Machine algorithm experienced an increase in accuracy from 0.79 to 0.88 on the MC1 dataset. These results show that the combination of Ant Colony Optimization and ensemble techniques can improve the accuracy of the algorithms, especially on the Decision Tree algorithm with an accuracy of 0.98 on dataset MC1. In addition, Random Forest algorithm is declared stable because it obtained good accuracy.

4. Conclusion

The evaluation of the defect software prediction model using Random Forest, Decision Tree, Support Vector Machine, and Naive Bayes algorithms after being combined with Ant Colony Optimization-based feature selection and ensemble technique, namely Gradient Boosting, shows that the accuracy of the four algorithm models had successfully increased by an average of 2% to 10%. Therefore, it can be concluded that the Feature Selection and Ensemble technique can increase the accuracy of the four algorithms being used.

Acknowledgement

We would like to show our gratitude to Universitas Jember for facilitating us in developing this research paper. Hopefully, this research can make a major contribution to the advancement of Computer Science.

References

[1] N. Hidayati, J. Suntoro, and G. G. Setiaji, "Perbandingan Algoritma Klasifikasi untuk Prediksi Cacat Software dengan Pendekatan CRISP-DM," *Jurnal Sains dan Informatika*, vol. 7, no. 2, pp. 117–126, Nov. 2021. https://doi.org/10.34128/jsi.v7i2.313

[2] Y. Liu, W. Zhang, G. Qin, and J. Zhao, "A comparative study on the effect of data imbalance on software defect prediction," in *Procedia Computer Science*, Elsevier B.V., 2022, pp. 1603–1616. https://doi.org/10.1016/j.procs.2022.11.349

[3] N. Grattan, D. Alencar da Costa, and N. Stanger, "The need for more informative defect prediction: A systematic literature review," Jul. 01, 2024, *Elsevier B.V*. https://doi.org/10.1016/j.infsof.2024.107456

[4] A. Saifudin, D. Romi, and S. Wahono, "Pendekatan Level Data untuk Menangani Ketidakseimbangan Kelas pada Prediksi Cacat Software," *Journal of Software Engineering*, vol. 1, no. 2, 2015.

[5] A. Hardoni, D. P. Rini, and S. Sukemi, "Integrasi SMOTE pada Naive Bayes dan Logistic Regression Berbasis Particle Swarm Optimization untuk Prediksi Cacat Perangkat Lunak," *JURNAL MEDIA INFORMATIKA BUDIDARMA*, vol. 5, no. 1, p. 233, Jan. 2021. http://dx.doi.org/10.30865/mib.v5i1.2616

[6] V. Chauhan, C. Arora, H. Khalajzadeh, and J. Grundy, "How do software practitioners perceive human-centric defects?," *Inf Softw Technol*, vol. 176, Dec. 2024. https://doi.org/10.1016/j.infsof.2024.107549

[7] A. Briciu, G. Czibula, and M. Lupea, "A study on the relevance of semantic features extracted using BERT-based language models for enhancing the performance of software defect classifiers," *Procedia Comput Sci*, vol. 225, pp. 1601–1610, 2023. https://doi.org/10.1016/j.procs.2023.10.149

[8] A. S. Dyer *et al.*, "Applied machine learning model comparison: Predicting offshore platform integrity with gradient boosting algorithms and neural networks," *Marine Structures*, vol. 83, May 2022. https://doi.org/10.1016/j.marstruc.2021.103152

[9] X. Dong, Y. Liang, S. Miyamoto, and S. Yamaguchi, "Ensemble learning based software defect prediction," *Journal of Engineering Research*, Nov. 2023. https://doi.org/10.1016/j.jer.2023.10.038

[10] A. Khalid, G. Badshah, N. Ayub, M. Shiraz, and M. Ghouse, "Software Defect Prediction Analysis Using Machine Learning Techniques," *Sustainability*, vol. 15, no. 6, p. 5517, Mar. 2023. https://doi.org/10.3390/su15065517

[11] Y. Al-Smadi, M. Eshtay, A. Al-Qerem, S. Nashwan, O. Ouda, and A. A. Abd El-Aziz, "Reliable prediction of software defects using Shapley interpretable machine learning models," *Egyptian Informatics Journal*, vol. 24, no. 3, Sep. 2023. https://doi.org/10.1016/j.eij.2023.05.011

[12] S. Manakkadu and S. Dutta, "Ant Colony Optimization based Support Vector Machine for Improved Classification of Unbalanced Datasets," in *Procedia Computer Science*, Elsevier B.V., 2024, pp. 586–593. https://doi.org/10.1016/j.procs.2024.05.143

[13] G. Gumelar, Q. Ain, R. Marsuciati, S. Agustanti Bambang, A. Sunyoto, and M. Syukri Mustafa, "Kombinasi Algoritma Sampling dengan Algoritma Klasifikasi untuk Meningkatkan Performa Klasifikasi Dataset Imbalance," 2021.

[14] A. John, I. F. Bin Isnin, S. H. H. Madni, and F. B. Muchtar, "Enhanced intrusion detection model based on principal component analysis and variable ensemble machine learning algorithm," *Intelligent Systems with Applications*, vol. 24, Dec. 2024. https://doi.org/10.1016/j.iswa.2024.200442

[15] M. Athoillah and R. K. Putri, "Handwritten Arabic Numeral Character Recognition Using Multi Kernel Support Vector Machine," *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, pp. 99–106, Mar. 2019. https://doi.org/10.22219/kinetik.v4i2.724

[16] I. Muslim Karo Karo and Hendryana, "Klasifikasi Penderita Diabetes Menggunakan Algoritma Machine Learning dan Z-Score," *Jurnal Teknologi Terpadu* , vol. 8 nomor 2, 2022.

[17] H. Nalatissifa, W. Gata, S. Diantika, and K. Nisa, "Perbandingan Kinerja Algoritma Klasifikasi Naive Bayes, Support Vector Machine (SVM), dan Random Forest untuk Prediksi Ketidakhadiran di Tempat Kerja," *Jurnal Informatika Universitas Pamulang*, vol. 5, no. 4, p. 578, Dec. 2021. https://dx.doi.org/10.32493/informatika.v5i4.7575

[18] M. N. Ahmad, Z. Shao, X. Xiao, P. Fu, A. Javed, and I. Ara, "A novel ensemble learning approach to extract urban impervious surface based on machine learning algorithms using SAR and optical data," *International Journal of Applied Earth Observation and Geoinformation*, vol. 132, Aug. 2024. https://doi.org/10.1016/j.jag.2024.104013

[19] S. Stradowski and L. Madeyski, "Industrial applications of software defect prediction using machine learning: A business-driven systematic literature review," Jul. 01, 2023, *Elsevier B.V*. https://doi.org/10.1016/j.infsof.2023.107192

[20] I. Mehmood *et al.*, "A Novel Approach to Improve Software Defect Prediction Accuracy Using Machine Learning," *IEEE Access*, vol. 11, pp. 63579–63597, 2023. https://doi.org/10.1109/ACCESS.2023.3287326

[21] Y. Chachoui, N. Azizi, R. Hotte, and T. Bensebaa, "Enhancing algorithmic assessment in education: Equi-fused-data-based SMOTE for balanced learning," *Computers and Education: Artificial Intelligence*, vol. 6, Jun. 2024. https://doi.org/10.1016/j.caeai.2024.100222

[22] K. Khadijah and P. S. Sasongko, "The Comparison of Imbalanced Data Handling Method in Software Defect Prediction," *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, pp. 203–210, Aug. 2020. https://doi.org/10.22219/kinetik.v5i3.1049

[23] W. Wu, K. Chen, and E. Tsotsas, "Prediction of rod-like particle mixing in rotary drums by three machine learning methods based on DEM simulation data," *Powder Technol*, vol. 448, p. 120307, Dec. 2024. https://doi.org/10.1016/j.powtec.2024.120307

[24] M. Azhari, Z. Situmorang, and R. Rosnelly, "Perbandingan Akurasi, Recall, dan Presisi Klasifikasi pada Algoritma C4.5, Random Forest, SVM dan Naive Bayes," *JURNAL MEDIA INFORMATIKA BUDIDARMA*, vol. 5, no. 2, p. 640, Apr. 2021. http://dx.doi.org/10.30865/mib.v5i2.2937

[25] L. Hakim, Z. Sari, A. Rizaldy Aristyo, and S. Pangestu, "Optimzing Android Program Malware Classification Using GridSearchCV Optimized Random Forest," *Computer Network, Computing, Electronics, and Control Journal*, vol. 9, no. 2, pp. 173–180, 2024.

[26] P. R. Sihombing and I. F. Yuliati, "Penerapan Metode Machine Learning dalam Klasifikasi Risiko Kejadian Berat Badan Lahir Rendah di Indonesia," *MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, vol. 20, no. 2, pp. 417–426, May 2021. https://doi.org/10.30812/matrik.v20i2.1174