

# The influence of machine learning on the predictive performance of cross-project defect prediction: empirical analysis

Yahaya Zakariyau Bala<sup>1,4</sup>, Pathiah Abdul Samat<sup>1</sup>, Khaironi Yatim Sharif<sup>3</sup>, Noridayu Manshor<sup>2</sup>

<sup>1</sup>Department of Software Engineering and Information System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Selangor, Malaysia

<sup>2</sup>Department of Computer System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Selangor, Malaysia

<sup>3</sup>Department of Computer and Information Science, Universiti Teknologi PETRONAS, Seri Iskandar, Malaysia

<sup>4</sup>Department of Computer Science, Faculty of Science, Federal University of Kashere, Gombe, Nigeria

## Article Info

### Article history:

Received Dec 12, 2023

Revised Mar 26, 2024

Accepted Mar 29, 2024

### Keywords:

Cross-project  
Defect prediction  
Machine learning  
Random forest  
Software defect

## ABSTRACT

This empirical investigation delves into the influence of machine learning (ML) algorithms in the realm of cross-project defect prediction, employing the AEEEEEM dataset as a foundation. The primary objective is to discern the nuanced influences of various algorithms on predictive performance, with a specific focus on the F1 score metric as evaluation criterion. Four ML algorithms have been carefully assessed in this study: random forest (RF), support vector machines (SVM), k-nearest neighbors (KNN), and logistic regression (LR). The choice of these algorithms reflects their prevalence in software defect prediction literature and their diversity. Through rigorous experimentation and analysis, the investigation unveils compelling evidence affirming the superiority of RF over its counterparts. The F1 score utilized as evaluation metric, capturing the delicate balance between precision and recall, essential in defect prediction scenarios. The nuanced examination of algorithmic efficacy provides practical insights for developers and practitioners navigating the challenges of cross-project defect prediction. By leveraging the rich and diverse AEEEEEM dataset, this study ensures a comprehensive exploration of algorithmic influences across varied software projects. The findings not only contribute to the academic discourse on defect prediction but also offer practical guidance for real-world application, emphasizing the pivotal role of RF as a tool in enhancing predictive accuracy and reliability.

This is an open access article under the [CC BY-SA](#) license.



## Corresponding Author:

Pathiah Abdul Samat

Department of Software Engineering and Information System

Faculty of Computer Science and Information Technology, Universiti Putra Malaysia

43400, Selangor, Malaysia

Email: pathiah@upm.edu.my

## 1. INTRODUCTION

Ensuring the reliability and quality of software products has become a top priority in the ever-changing world of software development. Among the myriad challenges faced by software engineers, the prediction and prevention of defects stand out as pivotal tasks in the pursuit of robust and resilient software systems. Early identification and mitigation of potential defects can translate into substantial cost savings, improved product quality, and enhanced user satisfaction. In recent times, the infusion of machine learning (ML)

algorithms into the realm of software defect prediction has offered promising avenues for addressing these challenges [1]-[6]. Cross-project defect prediction, a subdomain of defect prediction, involves the transfer of knowledge gained from one software project (source) to predict defects in another (target), as shown in Figure 1. Traditional defect prediction models often struggle with the characteristics of different project contexts, making cross-project prediction a compelling approach for leveraging existing data and knowledge [7]-[12].

ML algorithms, with their capacity to discern patterns and relationships within data, have emerged as powerful tools for enhancing the accuracy of defect prediction models [13], [14]. The predetermined set of training data is fed into a ML algorithm. The algorithm then learns from the training dataset (source) and generates rules for class label prediction for a fresh set of data (target) [15]-[20]. During the learning phases, mathematical procedures are used to create and improve the prediction function. The training data that was used in this method has a predetermined output value and an attribute input value. The result that is frequently known is compared to the expected ML algorithm quality. Until the optimal prediction accuracy is attained or the maximum number of loops is reached, this is repeated using training data.

However, the impact of these algorithms on cross-project prediction performance remains an area of active exploration. The motivation behind this empirical investigation stems from the recognition that while ML algorithms offer promise in defect prediction, their efficacy can vary significantly across different project environments. Factors such as project size, complexity, and development methodologies introduce variability that may influence algorithm performance. Empirical investigation will identify which algorithms are more effective in this context. This helps in understanding the practical applicability of different algorithms and their ability to generalize across diverse projects. By analyzing empirical results, researchers and practitioners can make informed decisions about selecting the most suitable ML algorithms for cross-project defect prediction, improving the overall reliability and effectiveness of software quality assurance processes. In this work, we empirically investigated the influence of four difference ML algorithms random forest (RF), support vector machines (SVM), k-nearest neighbors (KNN), and logistic regression (LR) on predictive performance of cross-project defect prediction.

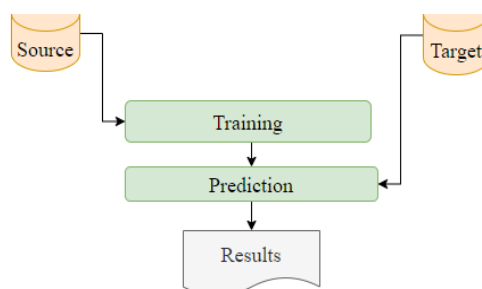


Figure 1. Cross project defect prediction (CPDP) process

## 2. METHOD

The goals of this study are to investigate the impact of different ML algorithms on the performance of cross-project defect prediction and to identify algorithms that exhibit superior cross-project generalization and enhance overall defect prediction performance. To achieve these goals, the following research questions were addressed:

- RQ1: how does a choice of ML algorithm affect the overall prediction performance of cross-project defect prediction?
- RQ2: which ML algorithms demonstrate the highest prediction performance in cross-project defect prediction scenarios?
- RQ3: how does the diversity of training datasets impact the performance of ML algorithms in cross-project defect prediction?

We started by grouping the datasets in to source and target for training and testing, respectively. The KNN, RF, SVM and LR are trained on source project and tested on target project. Their performance was measure using F1\_score in Figure 2.

### 2.1. Machine learning algorithm

The investigation of ML's influence encompasses evaluating various algorithms such as RF, SVM, KNN, and LR. These algorithms are integral to predictive analytics, each offering unique strengths in handling different types of data and patterns. The diverse set of algorithms aims to capture the breadth of approaches employed in defect prediction and their adaptability to cross-project scenarios.

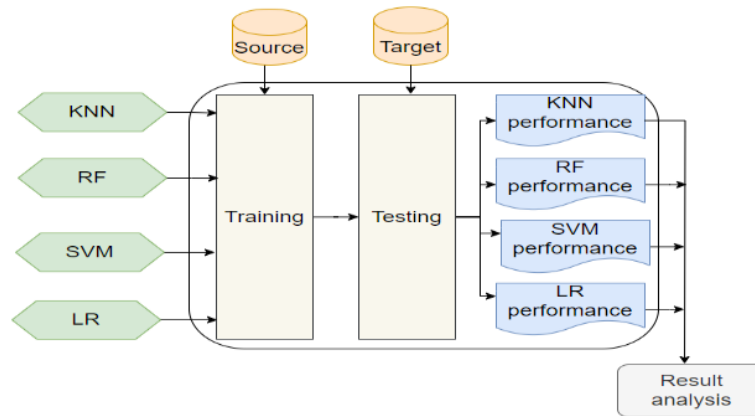


Figure 2. Research framework

### 2.1.1. K-nearest neighbor

The KNN algorithm is a simple, non-parametric, and lazy learning algorithm used for classification and regression tasks. It works by identifying the  $k$  nearest data points to a given input and making predictions based on the majority class (for classification) or the average value (for regression) of these neighbors. KNN's performance highly depends on the choice of  $k$  and the distance metric used, making it sensitive to noisy and irrelevant features. The object among the neighbors whose correct classification is known and selected [21], [22] in their study reported that KNN performed better than the compared classification models.

### 2.1.2. Random forests

RF is one of the effective ML classification algorithms which combines multiple classification trees [23]. During the classification process, each tree in RF makes a classification of each sample, then the final classification is obtained by voting [24]. In their study reported RF as the most effective classification.

### 2.1.3. Support vector machine

SVM is one of the most effective supervised ML algorithms used for both regression and classification. The SVM work on the principle that two groups can be separated by drawing decision boundaries between two classes of data point in a hyperplane and subsequently finding the optimal hyperplane [25], [26] investigated the predictive performance of SVM against eight statistical and ML algorithms on software defect datasets obtained from NASA. Results indicated that the SVM performed better.

### 2.1.4. Logistic regression

LR is one of the ML algorithms in which the relationship between features and labels is modeled as a probability distribution  $P(y|x)$ , where  $y$  is a label that can be either 0 for non-defective or 1 for defective and  $x$  refers to the data point represented as a set of features [27], in their study, used LR as a meter classifier and reported better results.

## 2.2. Datasets

We collected open-source projects (datasets) with varying sizes, domains, and characteristics from AEEEM repository to ensure diversity. Several open-source datasets are frequently utilized in software defect prediction (SDP) investigations, and these are some of them [28]. Table 1 shows that there are five software projects (datasets) in AEEEM, each with 71 features.

Table 1. Datasets

Project	#Modules	#Features	#Defect	Defect ratio
Equinox (EQ)	325	71	129	40%
Eclipse JTD core (JDT)	997	71	206	21%
Mylyn (ML)	1862	71	245	13%
Partial differential equation (PDE)	1492	71	209	14%
Lucene (LC)	399	71	64	9%

### 2.3. Evaluation metrics

Recall, precision, and F1\_score were the three evaluation metrics or measurements that were employed to assess each model. A commonly used metric in software defect prediction research is the F1\_score [29]. This is the precision and recall represented harmonically, as determined by (3). Precision: evaluate the model's ability to correctly identify non-defective modules.

$$Precision = \frac{TP}{(TP+FP)} \quad (1)$$

Recall: evaluate the model's ability to correctly identify defective modules.

$$Recall = \frac{TP}{(TP+FN)} \quad (2)$$

F1\_score: defined the precision and recall harmonic representation. The performance is better the higher the F-measure.

$$F1\_score = \frac{(2 \times Precision \times Recall)}{(Precision + Recall)} \quad (3)$$

Where,  $TP$  refers to the number of predicted non-defective module as non-defective,  $FP$  refers to number of predicting defective module as non-defective and  $FN$  refers to number of predicting non-defective module as defective.

### 2.4. Experiment design

We divided the datasets into training (source) and test sets (target) for each algorithm evaluation. To be in conformity with the previous studies in SDP, we arrange all the datasets in pairs. For instance, when EQ was used as a source, each of the other projects was used as a target i.e.,  $EQ \Rightarrow JDT, ML, PDE, LC$ . Consistent experimental conditions were ensured to isolate the impact of the ML algorithms. All the experiments were conducted using Jupiter notebook python.

### 2.5. Statistical evaluation

To determine which ML algorithm is better, we first examine whether the performance difference between any two predictors is caused by chance. We employed a non-parametric statistical tool (wilcoxon signed-rank test) to compare pairs of predictors. The results were presented in the form of the win/draw/loss i.e., the number of datasets upon which one predictor is better, equal, or lower than another predictor. To determine the practical size of the difference, we used cliff delta test based on the criteria specified in Table 2.

Table 2. Cliff's  $\delta$  effectiveness levels

Cliff's $\delta$	Effectiveness levels
$ \delta  < 0.147$	Negligible (N)
$0.147 \leq  \delta  < 0.33$	Small (S)
$0.33 \leq  \delta  < 0.474$	Medium (M)
$ \delta  \geq 0.474$	Large (L)

## 3. RESULTS AND DISCUSSION

This section presents the findings from our investigation into the influence of various ML algorithms on the prediction performance of cross-project defect prediction, guided by three key research questions. First, we examine which algorithm performs best in terms of predictive accuracy and robustness across different projects. Second, we analyze how each algorithm's performance metrics such as F1-score, highlighting their strengths and weaknesses. Finally, we discuss the practical implications of our findings for choosing appropriate algorithms in real-world defect prediction scenarios.

### 3.1. RQ1: how does a choice of machine learning algorithm affect the overall prediction performance of cross-project defect prediction?

Results in Table 3 indicate notable variations in predictive performance of cross-project defect prediction across algorithms. CPDP built using KNN (CPDP\_KNN) achieved average F1-score of 0.54, CPDP built using RF (CPDP\_RF) achieved average F1-score of 0.60, CPDP built using SVM (CPDP\_SVM) achieved average F1-score of 0.58, and CPDP built using LR (CPDP\_LR) achieved average F1-score of 0.52. Varying performance of cross-project defect prediction across different learning algorithms emphasized the impact of algorithm choice on prediction performance of cross-project defect prediction. Understanding how different

algorithms respond to the challenges posed by diverse projects is crucial for developing effective and reliable defect prediction models in real-world software development environments.

### 3.2. RQ2: which machine learning algorithms demonstrate the highest prediction performance in cross-project defect prediction scenarios?

Results in Table 3 indicate notable variations in performance across algorithms. RF achieved the highest F1-score most of the cross-project scenarios, demonstrating its effectiveness in handling cross-project defect prediction tasks. However, to further confirm the superiority of RF, we analyze its performance against each algorithms using statistical test.

#### 3.2.1. RF verses KNN

To further confirm whether the performance achievement of RF over KNN is not by chance. We conducted statistical test and size effect test as shown in Table 4. We performed a Wilcoxon rank sum test on 95% significant level i.e.,  $p=0.05$ . Prior to the test, we established the following hypothesis.

- Null hypothesis ( $H1_0$ ): RF does not achieve better prediction performance compared to the KNN.
- Alternative hypothesis ( $H1_A$ ): RF achieved better prediction performance compared to the KNN.

As shown in the table, the test result obtained for RF against KNN is 0.001. Since the result is less than 0.05. This simply means that, the difference between the performance of RF and KNN is statistically significant. Therefore, the alternative hypothesis is supported, and the null hypothesis is rejected. In addition, RF won against the KNN in 19 out of 20 datasets. Furthermore, to examine the size of effectiveness level. We used Cliff's delta. As shown in the table, the results showed that RF has non-negligible effectiveness on three datasets against KNN. Therefore, we can conclude that RF is more effective than KNN when selected for building CPDP model using AEEEM datasets as source projects.

#### 3.2.2. FR verses SVM

To further confirm whether the performance achievement of RF over SVM is not by chance. We conducted statistical test and size effect test as shown in Table 4. We performed a Wilcoxon rank sum test on 95% significant level i.e.,  $p = 0.05$ . Prior to the test, we established the following hypothesis.

- Null hypothesis ( $H1_0$ ): RF does not achieve better prediction performance compared to the SVM.
- Alternative hypothesis ( $H1_A$ ): RF can achieve better prediction performance compared to the SVM.

As shown in the table, the test result obtained for RF against SVM is 0.248. Since the result is not less than 0.05. this simply means that, the difference between the performance of RF and SVM is not statistically significant. Therefore, the null hypothesis is supported, and the alternative hypothesis is rejected. However, RF won against the KNN in 12 out of 20 datasets and loss 6 to SVM. Furthermore, to examine the size of effectiveness level. We used Cliff's delta. As shown in the table, the results showed that RF has non-negligible effectiveness on only one dataset against SVM. Therefore, although, RF outperformed SVM in 12 out of 20 datasets yet we can conclude that the difference between RF and SVM when it comes to building CPDP model using AEEEM datasets as source projects is not statistically significant.

#### 3.2.3. FR verses LR

To further confirm whether the performance achievement of RF over LR is not by chance. We conducted statistical test and size effect test as shown in Table 4. We performed a Wilcoxon rank sum test on 95% significant level i.e.,  $p = 0.05$ . Prior to the test, we established the following hypothesis.

- Null hypothesis ( $H1_0$ ): RF does not achieve better prediction performance compared to the LR.
- Alternative hypothesis ( $H1_A$ ): RF can achieve better prediction performance compared to the LR.

As shown in the table, the test result obtained for RF against LR is 0.005. Since the result is less than 0.05. This simply means that, the difference between the performance of RF and LR is statistically significant. Therefore, the alternative hypothesis is supported, and the null hypothesis is rejected. In addition, RF won against the LR in 17 out of 20 datasets and loss only 2 to LR. Furthermore, to examine the size of effectiveness level. We used Cliff's delta. As shown in the table, the results showed that RF has non-negligible effectiveness on five datasets against LR. Therefore, we can conclude that RF is more effective than LR if selected for building CPDP model using AEEEM datasets as source projects.

### 3.3. RQ3: how does the diversity of training datasets impact the performance of ML algorithms in cross-project defect prediction?

Table 5 show significant differences in each ML algorithm's predictive performance across different training datasets (source). For example, KNN achieved an average F1-score of 0.52 when using EQ as training datasets; an average F1-score of 0.58 when using JDT; an average F1-score of 0.55 when using LC; an average

F1-score of 0.54 when using ML; and an average F1-score of 0.52 when using PDE. Likewise with regard to every other algorithm. The varying performance of different ML algorithms across diverse training datasets implies that algorithm effectiveness is context-dependent. Each algorithm reacts differently to changes in the characteristics of the training data, such as project size, domain diversity, and temporal variations. This suggests that there is no one-size-fits-all solution in cross-project defect prediction; the choice of algorithm should be carefully considered based on the specific attributes of the training dataset.

Table 3. F1\_score on AEEEM dataset

Source→target	CPDP_KNN	CPDP_RF	CPDP_SVM	CPDP_LR
EQ→JDT	0.58	0.66	0.55	0.36
EQ→LC	0.47	0.53	0.53	0.49
EQ→ML	0.48	0.5	0.57	0.47
EQ→PDE	0.54	0.59	0.53	0.43
JDT→EQ	0.64	0.67	0.54	0.59
JDT→LC	0.51	0.59	0.55	0.55
JDT→ML	0.57	0.61	0.58	0.53
JDT→PDE	0.6	0.61	0.58	0.6
LC→EQ	0.61	0.46	0.63	0.67
LC→JDT	0.51	0.66	0.54	0.57
LC→ML	0.55	0.61	0.6	0.3
LC→PDE	0.52	0.59	0.6	0.41
ML→EQ	0.53	0.6	0.66	0.59
ML→JDT	0.57	0.65	0.55	0.5
ML→LC	0.48	0.57	0.51	0.54
ML→PDE	0.58	0.6	0.6	0.57
PDE→EQ	0.44	0.65	0.64	0.7
PDE→JDT	0.61	0.66	0.69	0.66
PDE→LC	0.51	0.63	0.56	0.5
PDE→ML	0.51	0.52	0.62	0.46
Mean	0.54	0.60	0.58	0.52

Table 4. Statistical test and effect size results

Test	RF and KNN	RF and SVM	RF and LR
Wilcoxon	0.001	0.248	0.005
W/D/L	19/0/1	12/2/6	17/1/2
Cliff's $\delta$ (N/S/M/L)	17/3/0/0	19/1/0/0	15/5/0/0

Table 5. F1\_score on different source projects

Source	CPDP_KNN	CPDP_RF	CPDP_SVM	CPDP_LR
EQ	0.52	0.57	0.55	0.44
JDT	0.58	0.62	0.56	0.57
LC	0.55	0.58	0.59	0.49
ML	0.54	0.61	0.58	0.55
PDE	0.52	0.62	0.63	0.58

#### 4. CONCLUSION

This empirical investigation has provided valuable insights into the influence of ML algorithms on the prediction performance of cross-project defect prediction. The study addressed three key research questions, elucidating the nuanced dynamics associated with algorithm selection in cross-project defect prediction context. The results demonstrated varying performance of cross-project defect prediction on different ML algorithms. This suggests that the choice of algorithm significantly influences the success of defect prediction across diverse software projects.

The results also unequivocally demonstrated the superiority of RF in cross-project defect prediction. RF consistently outperformed KNN, SVM, and LR across various cross-project scenarios based on F1\_score evaluation metric. RF exhibited the highest F1-score, indicating its robustness in capturing true positives while minimizing false positives and negatives. This underscores RF's effectiveness in providing a balanced and accurate prediction of defects across diverse projects. The findings provide practical guidelines for practitioners in selecting ML algorithms for cross-project defect prediction. RF emerges as a reliable choice, particularly when faced with varied project characteristics.

The robustness of RF across different project types suggests its applicability in real-world software development settings. This could lead to more reliable defect prediction models, aiding software teams in proactively addressing potential issues. As the field of ML continues to evolve, the lessons learned from this

investigation contribute to a more informed and effective approach to algorithm selection in the realm of cross-project defect prediction.

Future research could delve into fine-tuning strategies for RF to optimize its performance further. Exploring hyperparameter adjustments and ensemble configurations may unlock additional potential in enhancing defect prediction accuracy. In addition, investigating RF's performance in dynamic project environments, where codebases evolve over time, could provide valuable insights into the algorithm's adaptability and resilience in scenarios of continuous development.

## ACKNOWLEDGEMENTS

This work was supported by University Putra Malaysia and TETFund Nigeria.




## REFERENCES

- [1] W. Wen *et al.*, "A Cross-Project Defect Prediction Model Based on Deep Learning with Self-Attention," *IEEE Access*, vol. 10, pp. 110385-110401, 2022, doi: 10.1109/ACCESS.2022.3214536.
- [2] S. Goyal, "Effective software defect prediction using support vector machines (SVMs)" *International Journal of System Assurance Engineering and Management*, vol. 13, no. 2, pp. 681-696, 2022, doi: 10.1007/s13198-021-01326-1.
- [3] I. Batool, and T. A. Khan, "Software fault prediction using data mining, machine learning and deep learning techniques: A systematic literature review," *Computers and Electrical Engineering*, vol. 100, pp. 107886, 2022, doi: 10.1016/j.compeleceng.2022.107886.
- [4] S. Stradowski, and L. Madeyski, "Industrial applications of software defect prediction using machine learning: A business-driven systematic literature review," *Information and Software Technology*, 107192, 2023, doi: 10.1016/j.infsof.2023.107192.
- [5] C. Pornprasit, and C. K. Tantithamthavorn, "DeepLinedp: Towards a deep learning approach for line-level defect prediction," *IEEE Transactions on Software Engineering*, vol. 49, no. 1, pp. 84-98, 2022, doi:10.1109/TSE.2022.3144348.
- [6] Z. X. Li, Y. Jing, and X. Zhu, "Progress on approaches to software defect prediction," *IET Software*, vol. 12, pp. 161-175, 2018, doi: 10.1049/iet-sen.2017.0148.
- [7] B. Turhan, T. Menzies, A. Bener, and J. D. Stefano, "On the relative value of cross-company and within-company data for defect prediction," *Empirical Software Engineering*, vol. 14, pp. 540-578, 2009, doi: 10.1007/s10664-008-9103-7.
- [8] T. Zimmermann., N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-project defect prediction: A large scale experiment on data vs. domain vs. process" *ESEC/FSE '09: Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, New York, NY, USA, 2009, pp. 91-100, doi: 10.1145/1595696.1595713.
- [9] S. Herbold, A. Trautsch, and J. Grabowski, "Global vs. local models for cross-project defect prediction." *Empirical Software Engineering*, vol. 22, pp. 1866-1902, 2017, doi: 10.1007/s10664-016-9468-y.
- [10] X. Chen, *et al.*, "A survey on cross-project software defect prediction methods," *Chinese Journal of Computers*, vol. 41, pp. 254-274, 2018, doi: 10.11897/SP.J.1016.2018.00254.
- [11] S. Chen, J. M. Ye, and T. Liu, "Domain adaptation approach for cross-project software defect prediction," *Tongfang CNKI (Beijing) Technology Co.*, vol. 31, pp. 266-281, 2020, doi: 10.13328/j.cnki.jos.005632.
- [12] K. Stapor, "Evaluating and comparing classifiers: Review, some recommendations, and limitations," *Proceedings of the 10th International Conference on Computer Recognition Systems CORES 2017*, pp. 12-21, 2017, doi: 10.1007/978-3-319-59162-9.
- [13] M. Jorayeve, A. Akbulut, C. Catal, and A. Mishra, "Machine learning-based software defect prediction for mobile applications: A systematic literature review," *Sensors*, vol. 22, no. 7, pp. 2551, 2022, doi: 10.3390/s22072551.
- [14] M. Ahmad, S. Aftab, S. S. Muhammad, and S. Ahmad, "Machine Learning Techniques for Sentiment Analysis: A Review," *International Journal of Multidisciplinary Sciences and Engineering*, vol. 8, no. 3, pp. 27-32, 2017.
- [15] L. Alzubaidi *et al.*, "A survey on deep learning tools dealing with data scarcity: Definitions, challenges, solutions, tips, and applications," *Journal of Big Data*, vol. 10, no. 1, Apr. 2023, doi: 10.1186/s40537-023-00727-2.
- [16] L. Qiao, X. Li, Q. Umer, and P. Guo, "Deep learning based software defect prediction," *Neurocomputing*, vol. 385, pp. 100-110, 2020, doi: 10.1016/j.neucom.2019.11.067.
- [17] M. K. Thota, F. H. Shajin, and P. Rajesh, "Survey on software defect prediction techniques," *International Journal of Applied Science and Engineering*, vol 17, no. 4, pp. 331-344, 2020, doi: 10.6703/IJASE.202012\_17(4).331.
- [18] P. Manchala, and M. Bisi, "Diversity based imbalance learning approach for software fault prediction using machine learning models," *Applied Soft Computing*, vol. 124, pp. 109069, 2022, doi: 10.1016/j.asoc.2022.109069.
- [19] A. Khalid, G. Badshah, N. Ayub, M. Shiraz, and M. Ghouse, "Software Defect Prediction Analysis Using Machine Learning Techniques," *Sustainability*, vol. 15, no. 6, pp. 5517, 2023, doi: 10.3390/su15065517.
- [20] Y. Zhao, Y. Zhu, Q. Yu, and X. Chen, "Cross-Project Defect Prediction Method Based on Manifold Feature Transformation," *Future Internet*, vol. 13, no. 8, pp. 1-17, 2021, doi: 10.3390/fi13080216.
- [21] M. A. Mabayoje, A. O. Balogun, H. J. Jibril, J. O. Atoyebi, H. A. Mojeed, and V. E. Adeyemo, "Parameter tuning in KNN for software defect prediction: an empirical analysis," *Jurnal Teknologi dan Sistem Komputer*, vol. 7, no 4, pp. 121-126, 2019, doi: 10.14710/jtsiskom.7.4.2019.121-126.
- [22] S. Taneja, C. Gupta, K. Goyal, and D. Gureja, "An enhanced k-nearest neighbor algorithm using information gain and clustering." In *2014 Fourth International Conference on Advanced Computing & Communication Technologies*, pp. 325-329, 2014, doi: 10.1109/ACCT.2014.22.
- [23] I. H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," *Information and Software Technology*, vol. 58, pp. 388-402, 2015, doi: 10.1016/j.infsof.2014.07.005.
- [24] M. Akour, I. Alsmadi, and I. Alazzam, "Software fault proneness prediction: A comparative study between bagging, boosting, and stacking ensemble and base learner methods," *International Journal of Data Analysis Techniques and Strategies*, vol. 9, no. 1, pp. 1-16, 2017, doi: 10.1504/IJDATS.2017.083058.
- [25] B. Yalçiner, and M. Özdeş, "Software defect estimation using machine learning algorithms," In *2019 4th International Conference on Computer Science and Engineering (UBMK)*, pp. 487-491, 2019, doi: 10.1109/UBMK.2019.8907149.




- [26] O. E. Karim, and O. E. Mahmoud “Predicting defect-prone software modules using support vector machines,” *Journal of Systems and Software*, vol. 81, no. 5, pp. 649–660, 2008, doi: 10.1016/j.jss.2007.07.040.
- [27] Y. Zhang, D. Lo, X. Xia, and J. Sun, “Combined classifier for cross-project defect prediction,” an extended empirical study. *Frontiers of Computer Science*, vol. 12, no. 2, pp. 280-296, 2018, doi: 10.1007/s11704-017-6015-y.
- [28] Y. Z. Bala, P. A. Samat, K. Y. Sharif, and N. Manshor, “Improving Cross-Project Software Defect Prediction Method Through Transformation and Feature Selection Approach,” *IEEE Access*, vol. 11, pp. 2318-2326, 2023, doi: 10.1109/ACCESS.2022.3231456.
- [29] Y. Z. Bala, P. A. Samat, K. Y. Sharif, and N. Manshor, “Cross-Project Software Defect Prediction,” *Journal of Theoretical and Applied Information Technology*, vol. 100, no. 15, pp. 4826-4833, 2022.

## BIOGRAPHIES OF AUTHORS






**Yahaya Zakariyau Bala**    received his B.Sc. and M.Sc. degrees in Computer Science from Adamawa State University Mubi, Nigeria, in 2008 and 2014, respectively, and is currently a Ph.D. student in the Department of Software Engineering and Information System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM). He is currently Lecturer I with the Department of Computer Science, Faculty of Science, Federal University of Kashere, Nigeria. His research interests include software defect prediction and cross-project software defect prediction. He can be contacted at email: balagombi2@gmail.com or gs61002@student.upm.edu.my.






**Pathiah Abdul Samat**    received her B.Sc. and M.Sc. degrees in Computer Science from Universiti Teknologi Malaysia (UTM), in 1996 and 1998, respectively, and Ph.D. degree in Computer Science from Universiti Kebangsaan Malaysia (UKM), in 2012. She is currently a senior lecturer with the Department of Software Engineering and Information System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM). Her research interests include formal software verification, model checking. She can be contacted at email: pathiah@upm.edu.my.



**Khaironi Yatim Sharif**    received the Ph.D. degree from the University of Limerick, Ireland. He is currently an Associate Professor with the Computer and Information Science Universiti Teknologi PETRONAS, Seri Iskandar, Malaysia. He is also an Adjunct Associate Professor with the Shibaura Institute of Technology, Japan. His research interest includes the area of programmers' information need, particularly identifying programmers' information needs with regards to their task contexts such as software maintenance, program comprehension, code concept mapping, fault localization, and agile development. He can be contacted at email: khaironi@upm.edu.my.



**Noridayu Manshor**    received her B.Sc., and M.Sc. degrees in Computer Science from Universiti Putra Malaysia (UPM) and Universiti Teknologi Malaysia (UTM), respectively, and Ph.D. degree in Computer Science from Universiti Saint Malaysia (USM). She is currently a senior lecturer with the Department of Computer System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM). Her research interests include pattern recognition, image processing, and computer vision. She can be contacted at email: ayu@upm.edu.my.