International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-15 Issue-6, May 2026

Accomplishments and Challenges: A Research Study for Software Defect Prediction

Rajesh Prasad, Sunil Kadam, Vinayak Patil, Pramod Jadhav, Vinod Patil, Amol Kadam

Abstract: ***Investigation and prediction of defects in software is one of the important solutions to ensure software quality and reliability. Machine learning algorithms are used across a wide array of fields to solve real-world problems by building large, complex models. Many researchers have made significant contributions by developing predictive models for software defects using statistical and machine-learning approaches. But only a few frameworks have discussed the issue of building a universal software defect prediction model. Most existing models have been trained on limited datasets, which results in good performance on the training data but poor performance on unseen data. These limitations have motivated researchers to explore and develop more generalised and universal models for software defect prediction. Moreover, the growing complexity of contemporary software systems. Such limitations have encouraged researchers to investigate and build more generalised and universal models for software defect prediction. In addition, the increasing complexity of modern software systems and the rapid growth of software repositories have driven a demand for intelligent prediction techniques capable of handling heterogeneous data. Research is being conducted to investigate advanced machine learning and deep learning methods, including ensemble learning and transfer learning, to enhance prediction accuracy and adaptability across different software projects. These approaches aim to reduce development and maintenance costs and increase the overall reliability and performance of software products.***

Keywords: Software Defect, Cost, Class Imbalance, Data Extract

Nomenclature:
TNB: Transfer Naive Bayes
RQ: Research Question
VAB-SVM: Aware Boosting with Support Vector Machine
ULR: Univariate Logistic Regression
RFR: Random Forest
PCA: Principal Component Analysis

Manuscript received on 24 April 2026 | First Revised Manuscript received on 03 May 2026 | Second Revised Manuscript received on 09 May 2026 | Manuscript Accepted on 15 May 2026 | Manuscript published on 30 May 2026.

*Correspondence Author(s)
Dr. Rajesh Prasad, Department of Computer Engineering, Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune (Maharashtra), India. Email ID: rsprasad@bvucoep.edu.in, ORCID ID: 0000-0002-5148-3060
Dr. Sunil Kadam*, Department of Mechanical Engineering, Bharati Vidyapeeth (Deemed to be University) College of Engineering, Kolhapur (Maharashtra), India. Email ID: sunil.kadam@bharatividyapeeth.edu, ORCID ID: 0000-0003-2128-4850
Prof. Vinayak Patil, Department of Mechanical Engineering, Bharati Vidyapeeth (Deemed to be University) College of Engineering, Navi Mumbai (Maharashtra), India. Email ID: vinayak.patil@bvcoenm.edu.in, ORCID ID: 0000-0003-4672-8331
Dr. Pramod Jadhav, Department of CSBS, Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune (Maharashtra), India. Email ID: pajadhav@bvucoep.edu.in, ORCID ID: 0000-0003-1069-0853
Dr. Vinod Patil, Associate Professor, Department of Electronics and Telecommunication, Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune (Maharashtra), India. Email ID: vhpatil@bvucoep.edu.in, ORCID ID: 0000-0002-3328-9248
Dr. Amol Kadam, Associate Professor, Department of CSBS, Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune (Maharashtra), India. Email ID: akkadam@bvucoep.edu.in, ORCID ID: 0000-0002-2350-4397

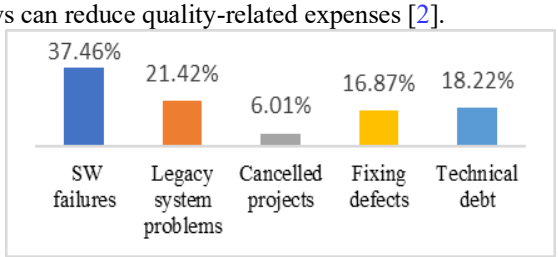
© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open-access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

I. INTRODUCTION

Since the early 1970s, work has been started in the field of software fault prediction. (F. Akiyama, IFIP Congress (1),1971). Within the SAMPLE project, Akiyama conducted a study using lines of code as software metrics. Using univariate logistic regression, S.S. Rathore et al. examine the relationship between object-oriented parameters at the class architecture. S. Wang et al tackled classification problems with imbalanced distributions using McCabe and Halstead as software metrics. J. Nam et al. proposed novel defect prediction approaches for an unlabeled dataset. David Bowes et al examined each fault on the four models to assess the degree of unpredictability in predictions [1].

For cross-project D, Ryu et al. contributed by addressing the problem of outliers that lower Cross-project prediction accuracy. S. Amasaki et al addressed the issue of noisy information in the inter-project data that can affect defect prediction performance. S. Huang et al. contributed to improving data quality for inter-project fault prediction by addressing data deficiencies and high dimensionality. L. Gong et al addressed the problem of class imbalance in cross-project defect prediction. Q. Yu et al contributed to improving the performance of multiple project faults. prediction by attribute subset selection and attribute ranking approach.

Despite the excellent work of many academics, the Krasner Quality Report estimates that the total cost of software quality for external issues and mistakes is about $635 billion. Fig. 1 shows cost areas related to poor-quality software, as presented in the Krasner quality report. Krasner quality report concludes that finding and fixing software flaws can reduce quality-related expenses [2].



[Fig.1: The Area of Cost Relates to Poor Software Quality]

The work is organised logically as follows: Section I presents the introduction to the topic. Section II presents related work done for bug prediction. A general framework for software defect prediction is

© Copyright: All rights reserved.

described in Section III. Software defect prediction issues and achievements are discussed in Section IV. In Section V, we talked about the answers to our research-related questions. The study's results are presented in the final part.

Both theoretical and experimental researchers' contributions to defect prediction are described in the literature. Our objective is to address the following research question (RQ) in software fault prediction.

RQ1. How to build software fault prediction model which perform better?
RQ2. Which software parameters have significance for creating a model for accurately predicting defects?
RQ3. How can we reduce the effect of class imbalance on the prediction of software faults?
RQ4: How to deal with extremely large dataset when predicting software defects?

II. RELATED WORK

For each module that was a part of the SAMPLE project, Akiyama analysed software system bugs in 1971. In this study, Akiyama observed a relationship between the number of errors and decision-making in the code and found a total of 546 defects across nine modules. This work used lines of code, and a defect measurement technique is required for large projects. McCabe and Chidamber et al. described software metrics that are helpful for bug prediction. McCabe cyclomatic complexity metrics give the number of paths to be tested, and Chidamber et al. discussed object-oriented design parameters that help control an object-oriented application [3]. Albert Endres described that errors can occur in the program due to technological, organisational, historical, and other factors. John E. Gaffney described several faults in relation to lines of code and the number of decision jumps. A formula for predicting the number of faults is developed using Akiyama assembly code data. B. Compton et al. worked on predicting software defects in the ADA package. This study was conducted using pre-delivery defects as predictors and Complexity as a predictor of defect density as a metric. J. Munson et al. built a statistical discriminant model to classify program modules as fault-prone or not. After examining experiments, this work concludes that there is a relationship between program faults and certain orthogonal complexity domains. N. Ohlsson et al. performed a study to find the relationship between design metrics and the number of function test failures. This work predicts fault-prone software in telephone switches at Ericsson Telecom.

Graves et al. count coding errors to assess the severity of software failures. A software module fault can be determined by analysing change history using change management data. Y. Brun et al. proposed a technique to identify program properties responsible for code errors. Machine learning models applied to these program properties will classify and rank the properties that caused errors. It is important to assess the quality and reliability of open-source software, as it is developed in various styles. T. Gyimothy et al presented work for open-source software bug prediction by discovering relationships between the object-oriented metrics and fault-proneness of classes. A. Schroter et al used design data and usage relationships between software components for finding software faults. Kim et al predicted faults by analysing the version history of software systems. This work predicts future faults at the file and function levels. T. Zimmermann et al. addressed the problem of multiple-project defect prediction models for projects with limited or inadequate data. It examined 622 multiple-project predictions and found that only 3.4% were successful, posing a significant obstacle to cross-project defect forecasting [4].

Burak Turhan presented work based on a correlation analysis that will investigate the relationship between software metrics and the number of defects. This work focused on the Pearson correlation parameter frequencies in the fault prediction model using stepwise linear regression. From the investigated projects, the highest correlation coefficient is obtained for two process metrics: Number of updates and Number of faults in previous releases. Product metrics such as Weighted Methods per Class, coupling between object classes, Response for a Class, and LOC are strongly correlated with the number of defects. B. Turhan et al presented work to find the defect prediction performance effects by using mixed project data. This work was tested on 10 proprietary projects from two different organisations to examine the performance of models for mixed-project data using naive Bayes and NN-filtering (Nearest Neighbour) [5].

Singh Rathore et al. focused on identifying relationships among classes in architectural spaces, using object-based parameters, given the object-oriented software program's vulnerability to errors, to evaluate coupling, cohesion, complexity, and inheritance design attributes. This work was tested on the PROMISE dataset using univariate logistic regression, and the findings indicate that models built using correlation and complexity measurements are more likely to contain errors. Also, better results, with greater accuracy, can be achieved by combining coupling and complexity metrics. Xingya Wang focused on building a model that will work with source and target data from several organisations. The model is built using Transfer Naive Bayes (TNB), a sample-weighting algorithm and a transfer learning method. The defect prediction model is built using weighted training data [6]. E. Shihab et al. addressed how to identify risky changes in software through a study of industrial software projects. Study performed with respect to six factors, such as the time when changes are made, the size of a change, the files modified by the change, the code modified by the change, the purpose of an Initial Modification Request, and the developers' experience with changing an Initial Modification Request. E. Shihab et al conclude that risky changes in software can be identified with high accuracy for the developer model. Risky changes can be identified by the number of lines added, the files modified by the change, and the developers' experience with Initial Modification Requests.

Ankur Tagra performed predictions on reopened bugs using a 4-dimensional (work habits, bug report, bug fix, and team) study structure. Top Node analysis performed on 24 factors from Eclipse, Apache

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

and OpenOffice open-source projects to find which factors are the best predictors of a bug being reopened. This work shows that bug reopening depends on the comment text and the bug's last status [7].

F. Zhang et al. developed a methodology for predicting all defects using a context-aware rank transformation approach, clustering projects by predictor relevance. This approach transforms raw values for all parameters across different projects into levels on the equivalent scale. The findings show that rank conversion yields results equivalent to log translation, and the proposed approach can be used to build a universal defect prediction model [8]. Duksan Ryu et al. addressed the issue of outliers that reduce cross-project prediction accuracy by using a transfer-learning-based model called VAB-SVM (Aware Boosting with Support Vector Machine). Experimental results show that the Area Under Curve and H-measure with VAB-SVM increase for almost all combinations.

Prateek Anand worked on the issue of future defects caused by code churn, using the test instance flow of execution in the code. This approach will determine the order of faulty features in the next release before the testing phase. Four industrial projects and 12 data sets were used to evaluate this defect-prediction technique at the feature level. Observations indicate a strong relationship between a higher impact score and the discovery of genuine bugs during testing. This lists attributes prone to errors and is useful for assigning more testing resources to high-ranking features. S. Amasaki et al addressed the problem of noisy information in the data that can affect cross-project defect prediction performance. Data are simplified using the Burak-Filter method. This work tested 14 open-source software using Logistic Regression, Naive Bayes, Random Forests, and SVM as prediction models.

Shruthi Puranik et al.'s work addressed the problem of bug prediction models that perform well on the training dataset but sometimes underperform on the test dataset. This study developed an approach that uses marginal R^2 values to estimate error-proneness scores through regression as an intermediate step, selecting the minimum number of best-performing metrics.

Using linear and non-linear heterogeneous ensemble approaches that integrate the outputs of many base learners, Santosh Singh Rathore et al. addressed the problem of determining the number of faults in the provided software modules. Work tested on the PROMISE data repository with a five-fold cross-validation approach used for intra-release prediction [9]. For Inter-releases prediction, older releases are used as the training and validation datasets, and the newer release is used as the testing dataset. The result says that the ensemble method performs significantly better than the individual participating base learners. Song Huang et al. addressed data deficiency and high dimensionality to improve data quality in cross-project defect prediction. This work proposed a novel three-stage defect prediction model: feature selection as the first step, the second step is to construct a homogeneous training dataset, and the last step is a Bayes classifier to classify defects.

The problem of splitting bug information into distinct module subsets and training each subset with learning approaches before combining the results for accessible methods was addressed by Santosh Singh Rathore et al. This work gives strategies that dynamically choose learning strategies to predict the number of software faults [10]. This model was tested on PROMISE data and Eclipse bug data, and the outcome indicates that the dynamic selection method produced improved or at least equivalent outcomes. Lina Gong et al. addressed the problem of class imbalance in cross-project defect prediction using Stratification embedded in the nearest-neighbour method [11].

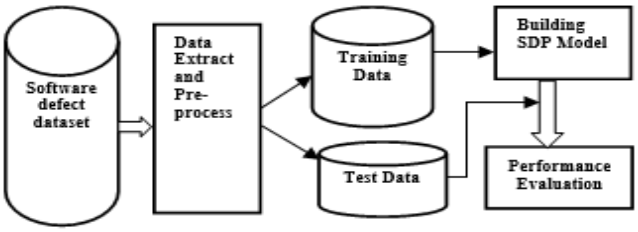
J. Sun et al proposed a geodesic flow kernel software fault prediction method to solve the issue of different distributions of training and testing datasets for cross-project defect prediction. Qiao Yu et al. investigated the problem of varying project fault prediction performance using attribute subset selection and attribute ranking approaches. The results show that both the attribute subset selection and attribute ranking approaches can enhance the performance of multiple project defect prediction. Table 1 shows some researchers' work in fault prediction.

Table I: Work Summary for Software Fault Prediction

Author	Technique/Methods Used	Dataset used	Evaluation Metrics
Rathore et al	univariate logistic regression (ULR), Naive Bayes, Random Forest	PROMISE	Area under ROC curve, Spearman's Correlation
D. Bowes et al	Naive Bayes, Random Forest, SVM, RPart	NASA	Matthew's correlation coefficient, F-measure
L. Gong et al	Cluster-based oversampling with noise filtering, SMOTE, Borderline SMOTE, ADASYN, random oversampling	NASA, AEEEM, ReLink and SOFTLAB	Recall and bal values
S. S. Rathore et al.	Linear and non-linear heterogeneous ensemble methods	PROMISE	Average relative error, Average absolute error, F-measure
S. S. Rathore et al.	Dynamic classifier with Bagging, Boosting	Eclipse, PROMISE	Average relative error, Average absolute error, F-measure
B. Pes	AdaBoost, Random Forest, Support Vector Machine, nearest Neighbours	Omentum, noglioma, SLC, uterus, lymphoma	Matthew's correlation coefficient, F-measure
Z. Li et al	MAHAKIL, SMOTE, ROS, ADASYN, Random Forest, C4.5, KNN, SVM	PROMISE	F-measure, Pf-proportion of defect-free, Pd-proportion of defective
H. He *et al*	principal component analysis, random and adaptive synthetic sampling, RIPPER classifier	NASA, MDP	AUC, F-measure, Accuracy, Recall,
Rehan et al	SVM	UCI datasets	G-mean
ZHENG et al	I Ada Boost, SWIM Boost	AEEEM, NASA	G-mean, AUC measures

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

III. GENERAL FRAMEWORK FOR SOFTWARE DEFECT PREDICTION



[Fig.2: General Framework for Software Defect Prediction]

Fig.2 shows a general framework for software defect prediction. Its working is as follows:

A. Data Extraction and Pre-Processing: A large dataset was extracted from the source using chunks or a changing-data-format technique. Extracted data need to be preprocessed because machine learning algorithms, such as Random Forest (RF), do not handle null values.

B. Training and Testing Datasets: The prediction model is trained on the training dataset and evaluated on the testing dataset.

C. Model Building: Various machine learning and statistical approaches, such as supervised, unsupervised, and deep learning, as well as sampling and filtering, can be used to build robust predictive models.

D. Performance Measures: F-measure, precision, recall, and accuracy, which are obtained from the confusion matrix, can be used to evaluate prediction models. The confusion matrix evaluates the following criteria:

i. **TP (True Positive):** Predict defective class as defective
ii. **FP (False Positive):** Predict non-defective class as defective
iii. **FN (False Negative):** Predict defective class as no-defective
iv. **TN (True Negative):** Predict non-defective class as non-defective

IV. SOFTWARE DEFECT PREDICTION ACHIEVEMENTS AND ISSUES

A. High-Dimensional Dataset Problem

Dealing with a large number of data features in a high-dimensional dataset is a serious challenge. This issue can be addressed with a dimensionality reduction technique in two ways: 1) Deriving new attributes by combining attributes. 2) Selecting only those that are correlated with the target class.B. Pes et al performed research on mix feature selection with sampling-based balancing methods and cost-sensitive learning.

The performance of the software defect prediction model is affected by the presence of unnecessary and irrelevant features in a high-dimensional dataset. Feature selection can solve this problem. Z. Xu et al. used 32 feature selection methods from five categories (ranking attributes as a filter, evaluating attribute subsets based on filters, evaluating attribute subsets based on wrappers, attribute selection based on clustering methods, and an extraction-based feature selection method) [12].

For high-precision prediction, it is necessary to pay attention to the dataset's quality given its characteristics. To assess the unpredictable nature of a defect dataset, Z. He et al. developed a method that uses the minimum Mean Absolute Error. This work concludes that there are three main causes of the lack of predictability in fault datasets. 1. Missing important information 2. Attribute values are distributed irrationally 3. Difference in defect correcting capacities.

B. Class Imbalance Problem

In some defect classification datasets, target classes do not have equal sample counts, which affects the accuracy of the prediction model. The model will achieve good accuracy on trial data, but on experimental data, it will perform worse on balanced error rate because it favours the class with larger samples. To address this class imbalance problem, Y. Ou et al. proposed an SVM algorithm using a Gaussian kernel with different cost parameters [13].

Z. Li et al addressed the problem of unbalanced data between and within classes for software fault prediction. The inter-class imbalance problem is addressed by 3 conditions in the distribution of sample data: 1. The non-faulty sample is significantly bigger than the faulty sample. 2. Non-faulty samples outnumber faulty samples. 3. The amount of a faulty sample exceeds that of a non-faulty sample. Large new samples are created in thinly distributed areas. In contrast, fewer new samples are created in thickly distributed areas to preserve distribution balance in the presence of intra-class imbalance.

He *et al. proposed an ensemble MultiBoost-based prediction model to reduce dataset deviation and variance*. An Ensemble MultiBoost model is used with principal component analysis (PCA) for dimensionality reduction and redundancy removal, and the combined sampling method is used to address class imbalance.

The class in the dataset that expresses imbalance among or within classes can be used for class-imbalance learning. S. Wang et al. addressed the issue of whether and how class-imbalance learning methods can benefit software defect prediction. Class imbalance problems can be at the data or algorithm levels. Data-level class imbalance can be addressed using various sampling techniques, while algorithm-level class imbalance can be addressed with cost-sensitive learning algorithms. The author tackled the class imbalance problem by undersampling and ensemble methods.

C. Outlier and Noise Problem

While addressing class imbalance, it may introduce unnecessary noise instances. To address this issue, L. Gong et al. developed a cluster-based oversampling method with noise reduction to mitigate SDP class imbalance. In this work, the data are split into K clusters for faulty instances, and a Closest list noise detection method is applied to filter noise instances produced by oversampling faulty instances.

Identifying exceptional events, interesting patterns, and unexpected behaviour in datasets requires paying attention to remove outliers. H. Liu et al. proposed a method that detects outliers in two steps. In the

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

first step, it calculates the Local Projection Score by projecting the nearest neighbours into the low-dimensional subspace. Then, determine outliers based on the Local Projection Score..

D. Software Metrics Problem

The performance of the defect prediction model depends on the characteristics of software metrics. Considering this, software practitioners are always concerned about the minimum number of software parameters required for an efficient fault prediction model. H. Wang et al suggested that threshold-based selection of attributes to eliminate unnecessary and duplicate software parameters. The experiments show that, on average, three software parameters are sufficient to create models that predict software faults.

Having information about a bug will provide an early indication of which software component will fail. This information can be used as software metrics for predicting software bugs. C. Couto et al mapped correlations between class-level faults and software quality parameters using Granger causality. A Granger causality test between two stationary time series says that x and y, then it checks whether x helps predict *y*. By establishing a relationship between time series of parameters and time series of faults, Couto et al. were able to correlate historical metric values with the causes of 64% to 93% of the reported system problems.

There are two ways to pre-process software parameters: attribute extraction, which creates an entirely new collection of features, and feature selection, which selects the most significant attributes. Y. Xia et al. proposed ReliefF-LC for software metrics selection, which considers software parameter duplication and their relationships with modules prone to defects. The experimental results are evaluated using ReliefF and InfoGain attribute selection methods, showing that the recommended approach can enhance fault prediction performance.

Interpreting defect prediction models without removing redundant metrics can be misleading or lead to a false relationship between two metrics [14]. To detect redundant metrics, J. Jiarpakdee et al. performed experiments on 101 publicly available datasets using variance inflation factors, variable clustering, and redundancy analysis. The result found that 10%-67% of metrics across the 101 public defect datasets are redundant. Shihab et al. addressed the issue of quantifying the impact of code and process metrics on the detection of post-release defects. This work used Logistic regression to identify collinear metrics affecting post-release defects.

V. RESEARCH QUESTION DISCUSSION

A. RQ1. How to Build a Software Fault Prediction Model that Performs Better?

From our study, we conclude that the performance of the software fault prediction model depends on:

i. *Dataset:* It is a key factor in model performance, as the model uses the data and applies statistical and machine learning techniques to achieve accurate results. If the dataset is less noisy and has no outliers, performance can be achieved. The second dataset should have a strong relationship between relevant features and the target class. A training dataset is important because the model built on it is evaluated on the testing dataset [15].

Feature selection plays an important role in accurate defect prediction because it serves as a predictor for the model. Removing irrelevant and unnecessary features improves the model's performance..

ii. *Modelling Techniques:* Researchers used various approaches to build predictive models, including single-model, hybrid, and ensemble methods. The modelling technique can be decided by the type of problem and the available dataset. For labelled data, supervised techniques (regression, classification) can be used, and for unlabelled data, unsupervised techniques (clustering, principal component analysis) can be used.

B. RQ2. Which Software Parameters Have Significance for Creating a Model for Accurately Predicting Defects?

Process metrics and product metrics are used to measure software quality. Process metrics like pre-release and post-release need to be incorporated into the fault prediction model. Code metrics such as lines of code, cyclomatic complexity, number of classes, number of methods, and number of interfaces are successfully used to build a defect prediction model.. The mapping of software parameters to the fault-prone category is significant when building a model.

C. RQ3. How to Minimise the Impact of Class Imbalance on Software Fault Prediction?

Two sampling techniques—undersampling, which removes instances of the majority class, and oversampling, which adds instances of the minority class—can mitigate class imbalance. Many researchers have addressed class imbalance and have solved it using various techniques.

D. RQ4. How to Deal with an Extremely Large Dataset when Predicting Software Defects?

If you are using a high-dimensional dataset, you may encounter issues such as outliers, noisy or redundant information, and too many attribute selections when building a high-performance model. It can be mitigated through dimensionality reduction, feature selection, and outlier removal.

VI. CONCLUSION

Since the first study (1971) on fault prediction, research progress and contributions in the field of software defect prediction have been remarkable. In this paper, we started with our research question, and our study focused on the literature exploring issues such as different types of datasets and their performance effects, the relationship between faults and software metrics, and modelling techniques used according to problem type and model evaluation criteria. Many studies have addressed issues in SDP very well; however, many open challenges remain, such as class imbalance, high-dimensional datasets, and robust models that generalise to other projects. The paper aims to

© Copyright: All rights reserved.

study previous defect prediction studies and start our research with respect to open challenges in SDP.

FUTURE SCOPE

The focus of future research could be on developing a model for cross-project fault prediction; this will improve software reliability and robustness. To develop and generalise the model, we should test on more datasets, such as commercial or licensed software data. Another area for future work is to identify features that are important for predicting the correct class when detecting software faults. Decreased feature count may boost the accuracy of classifiers. We should pay attention to other problem areas, such as data security, outliers, incorrect data, and class imbalance.

DECLARATION STATEMENT

As the article's author, I must verify the accuracy of the following information after aggregating input from all authors.

- **Conflicts of Interest/ Competing Interests:** Based on my understanding, this article has no conflicts of interest.
- **Funding Support:** This article has not been funded by any organizations or agencies. This independence ensures that the research is conducted objectively and without external influence.
- **Ethical Approval and Consent to Participate:** The content of this article does not necessitate ethical approval or consent to participate with supporting documentation.
- **Data Access Statement and Material Availability:** The adequate resources of this article are publicly accessible.
- **Author's Contributions:** The authorship of this article is contributed equally to all participating individuals.

REFERNCES

1. David Bowes, Tracy Hall, and Steve Counsell, "Uncertainty in Software Defect Prediction," *Empirical Software Engineering*, 2016. DOI: https://doi.org/10.1007/s10664-016-9431-3 URL: https://link.springer.com/article/10.1007/s10664-016-9431-3
2. H. Krasner, "Quality Software A 2018 Report," *Consort. IT Softw. Qual.*, 2018.
3. S. Kaliraj, V.G.P. Sahasranth, and V. Sivakumar, "A holistic approach to software fault prediction with dynamic classification," *Automated Software Engineering*, 2024. DOI: https://doi.org/10.1007/s10515-024-00467-4
4. Xin Xia, David Lo, Sinno Jialin Pan, Nachiappan Nagappan, and Xinyu Wang, "HYDRA: Massively Compositional Model for Cross-Project Defect Prediction," *IEEE Transactions on Software Engineering*, Vol. 42, No. 10, pp. 977–998, 2016. DOI: https://doi.org/10.1109/TSE.2016.2543218
5. Seyedrebvar Hosseini and Burak Turhan, "A Comparison of Similarity-Based Instance Selection Methods for Cross-Project Defect Prediction," 2021. URL: https://arxiv.org/abs/2104.01024
6. Xin Xia, David Lo, Xin Wang, and Bo Zhou, "Cross-Project Defect Prediction Using Transfer Learning Techniques," *IEEE Transactions on Software Engineering*, 2016. DOI: https://doi.org/10.1109/TSE.2016.2543218 URL: https://ieeexplore.ieee.org/document/7425176
7. Ankur Tagra, Haoxiang Zhang, Gopi Krishnan Rajbahadur, and Ahmed E. Hassan, "Revisiting Reopened Bugs in Open Source Software Systems," *arXiv preprint*, 2022. URL: https://arxiv.org/abs/2202.08701
8. F. Zhang, Q. Zheng, Y. Zou, and A. E. Hassan, "Cross-Project Defect Prediction Using Context-Aware Rank Transformation," *Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2016. DOI: https://doi.org/10.1109/ICSME.2016 URL: https://ieeexplore.ieee.org/document/
9. Santosh Singh Rathore, S. Kumar, and V. Singh, "Ensemble Learning-Based Software Fault Prediction Using Hybrid Classifiers," *Neural Computing and Applications*, 2019–2021, related studies. DOI: https://doi.org/10.1007/s00521-019-04386-8 URL: https://link.springer.com/article/10.1007/s00521-019-04386-8
10. Santosh Singh Rathore, S. Kumar, and V. Singh, "Hybrid and Ensemble-Based Machine Learning Approaches for Software Fault Prediction," *Neural Computing and Applications*, 2019–2021. DOI: https://doi.org/10.1007/s00521-019-04386-8 URL: https://link.springer.com/article/10.1007/s00521-019-04386-8
11. Lina Gong, Xiaobing Sun, and Zhenyu Chen, "Addressing Class Imbalance in Cross-Project Defect Prediction Using Stratified Nearest Neighbour," *Information and Software Technology*, Vol. 75, pp. 80–94, 2016. DOI: https://doi.org/10.1016/j.infsof.2016.02.001 URL: https://www.sciencedirect.com/science/article/pii/S0950584916300198
12. Misbah Ali, Tehseen Mazhar, Tariq Shahzad, et al., "Analysis of Feature Selection Methods in Software Defect Prediction Models," *IEEE Access*, Vol. 11, pp. 145954–145974, 2023. DOI: https://doi.org/10.1109/ACCESS.2023.3343249 URL: https://ieeexplore.ieee.org/document/10323200
13. Somya R. Goyal, "A Systematic Review on AI-Based Class Imbalance Handling in Software Defect Prediction," *Results in Engineering*, Vol. 27, 2025. DOI: https://doi.org/10.1016/j.rineng.2025.106578 URL: https://doi.org/10.1016/j.rineng.2025.106578
14. Xinghuo Chen, David Lo, and Xin Xia, "Interpretable Software Defect Prediction Models: A Systematic Literature Review," *Information and Software Technology*, 2020. DOI: https://doi.org/10.1016/j.infsof.2020.106371 URL: https://www.sciencedirect.com/science/article/pii/S095058492030126X
15. Ahmed E. Hassan, David Lo, and Xin Xia, "Software Analytics: The Role of Data Quality in Software Defect Prediction," *IEEE Software*, Vol. 35, No. 6, pp. 84–90, 2018. DOI: https://doi.org/10.1109/MS.2018.2875983 URL: https://ieeexplore.ieee.org/document/8453143

AUTHOR'S PROFILE



Dr. Rajesh Prasad is currently serving as the Principal of Bharati Vidyapeeth Deemed University College of Engineering. He has more than 25 years of academic and research experience in Computer Science and Engineering. His research interests include Artificial Intelligence, Data Mining, Soft Computing, Information Retrieval, Text Analytics, Machine Learning, and Intelligent Systems. He has published numerous research papers in reputed national and international journals and conferences, and has actively contributed to academic leadership, curriculum development, NBA/NAAC quality initiatives, and research mentoring. He is associated with various professional bodies, including ISTE and CSI, and has guided several undergraduate, postgraduate, and doctoral research projects.



Dr. Sunil Jagannath Kadam is a Professor and Head of the Mechanical Engineering Department at Bharati Vidyapeeth's College of Engineering with more than two decades of teaching and academic experience. He holds a PhD in Mechanical Engineering with a specialisation in Nanotechnology, Automobile Engineering, and I.C. Engines. He has guided numerous UG and PG projects and actively contributed to research through 59 publications and 7 granted patents. Dr. Kadam has organized several national conferences, FDPs, workshops, and industrial training programs. He has also received multiple Best Paper Awards for his research contributions in engineering and technology.



Mr. Vinayak N. Patil: Department of Mechanical Engineering, Bharati Vidyapeeth College of Engineering, Navi Mumbai. Completed M.Tech. from V.J.T.I.Matunga Mumbai. Also have published 07 research papers in reputable international journals indexed in Scopus, Web of Science, etc., and filed 01 patent in 2022; 01 Patent has been granted. I have also attended several workshops and faculty development programs.

© Copyright: All rights reserved.



Dr. Pramod A. Jadhav is an Associate Professor in the Department of Computer Science and Business Systems at Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune. He has extensive experience in teaching, research, and academic administration, with expertise in software engineering, LoT, and emerging technologies. Dr Jadhav has contributed to publications in reputable journals and conferences. He is actively involved in curriculum development, student mentoring, and outcome-based education practices.



Dr. Vinod H. Patil is a working professional and researcher in the field of Electronics and Telecommunication Engineering. He received a PhD in Electronics Engineering in 2020, with the topic "Spectrum Management in Cognitive Radios". He is a reviewer for various reputable international journals and Conferences. Currently working as an Assistant Professor
and Research Guide in Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune (India). His primary research areas are Cognitive Radio, Wireless Sensor Networks, Smart Agricultural Systems, Smart Grid Systems, Machine Learning, and Artificial Intelligence.



Dr. Amol Krishnat Kadam is the Head and Associate Professor in the Department of Computer Science & Business Systems at Bharati Vidyapeeth (Deemed to be University) College of Engineering, with over 18 years of academic experience. His research areas include Artificial Intelligence, Machine Learning, Software Reliability, and Deep Learning. He has published more than 46 international research papers, guided PhD scholars, and received multiple research grants from AICTE, UGC, and other agencies. Dr Kadam also holds several patents in AI, IoT, and software systems.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of the Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)/ journal and/or the editor(s). The Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP) and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

© Copyright: All rights reserved.