Cureus Journal of Computer Science

Open Access Review Article

Machine Learning Approaches in Software Fault Prediction: A Review

Ruchika Aggarwal [1, 2] , Kamaljit Kaur [1]

1. Department of Computer Science, Shri Guru Granth Sahib World University, Fatehgarh Sahib, IND 2. Department of Computer Application, CGC Landran, Mohali, IND

Corresponding author: Ruchika Aggarwal , ruchikaphd01@gmail.com

Received 12/17/2025
Review began 01/07/2026
Review ended 01/28/2026
Published 01/28/2026

DOI:
https://doi.org/10.7759/s44389-026-00021-7

Abstract

This paper reviews recent developments in the use of machine learning methods for predicting software faults to enhance software reliability and quality. To address the various challenges concerning software fault prediction, we conducted a review of 45 articles published between 2023 and 2025 from IEEE, Springer, ACM, and ScienceDirect digital libraries. The paper covers topics such as factors influencing software fault prediction, prediction techniques, datasets and software metrics, evaluation metrics, model selection criteria, and challenges associated with current solutions. The results indicate that Support Vector Machine and Random Forest are superior in terms of accuracy, precision, recall, and F1-score. The use of public datasets, particularly those from the PROMISE and NASA Metric Data Program repositories containing product metrics, is widespread and contributes to improved model performance. The purpose of this study is to advance research in software fault prediction and support the development of high-quality software products by improving defect predictability. This review will be useful to researchers, as it presents the latest overview of the existing literature on software defect prediction.

Categories: AI applications, Software Testing, Machine Learning (ML)
Keywords: software fault prediction, artificial intelligence, machine learning, classification, quality assurance

Introduction And Background

Software fault prediction (SFP) has become essential for ensuring software quality, as more than half of development costs are spent on detecting and correcting defects. With the global software market valued at $3.7 trillion of which 23% is devoted to quality assurance and testing, early detection of faults is critical to reducing cost and preventing catastrophic failures in large, complex systems. SFP resolves this problem by locating modules that are prone to defects to ensure that testing activities are prioritized in areas where they are most required to achieve better quality software at reduced costs [1].

SFP identifies defect-prone modules through creation of various classification and categorization models utilizing machine learning (ML) approaches. It belongs to the software development life cycle where we forecast the fault with an ML methodology with previous data. It is a systematic approach that facilitates production of quality products and low-cost software within the short potential time to satisfy the customer expectations [2].

ML has thus emerged as a key enabler for SFP. ML algorithms are able to automatically learn the patterns that are historically learned in software metrics, including code complexity, change history, and developer activity to provide an accurate prediction of the modules or components most likely to contain a fault. The scalability of the ML to large data volumes, the capability to adapt to the changing software, and the combination of various features make it specifically applicable to the contemporary software engineering setting [3].

Literature review

In order to enhance the quality of software, the main focus should be on creation of a superior SFP model. It is possible to introduce ML/AI-based models and development tools that will enable the detection and correction of defects during development and will avoid the need to use high resources. The following literature review shows the recent research (2023-2025) on ML techniques, datasets, and evaluation metrics.

Software Fault Prediction

The growing sophistication of software systems has facilitated the increased attention to software defect prediction (SDP), where recent research investigates application scenarios, ML/deep learning (DL) methods, datasets, metrics of evaluation, and validation methodology [4]. According to the research conducted by SDP, it is demonstrated that both the traditional ML models and the latest models of DL exhibit a considerable enhancement in the defect classification and increase the reliability of the software,

decreased the cost of development, and increased the quality assurance [5]. Detecting the factors that made the system faulty and allowing it to be improved continuously are also provided by AI-driven detection methods that enhance system adaptability [6].

Current literature highlights the strong effectiveness of neural networks, DL, ensemble methods, and feature-selection strategies, with the NASA MDP and PROMISE repositories being the most frequently used public datasets [7]. The studies emphasize the importance of considering numerous influencing factors in the prediction process, the benefits of early fault detection, the effectiveness of different classifiers, the practical application of ML for predicting flaws in Python programs, and the advantages of applying SFP [8,9]. Comparative analyses indicate that supervised models, particularly neural networks and Support Vector Machines (SVM), generally outperform unsupervised models, although autoencoders may be useful for detecting anomalies in large codebases [10].

Supervised Machine Learning in SFP

ML has found extensive application in SDP, and research indicates that ML algorithms yield different outcomes across datasets, resulting in unequal defect prediction rates and no single model consistently outperforming the others. Initial experiments comparing Naïve Bayes, K-Nearest Neighbor, neural networks, and SVM using area under the curve (AUC) values identified the highest-performing classifiers [11]. Research on feature selection has also shown that dimensionality reduction methods do not significantly affect computational efficiency or prediction accuracy when neural networks are applied for classification [12].

Further comparison of Logistic Regression, SVM, and Random Forest (RF) was made, and RF was considered as the most accurate model, even better than SVM and by far better than Logistic Regression [13]. Optimization-based investigations on dimensionality reduction with Principal Component Analysis (PCA) and hyperparameter optimization with random search were able to note that K-Nearest Neighbor had the highest accuracy, precision, and recall in all NASA Metric Data Programme (MDP) datasets, and that Synthetic Minority Over-sampling Technique (SMOTE) was successful in class imbalance reduction [14]. On the same note, the use of hyperparameter optimization tools of NDSGA-II and Hyperband indicated that RF once again performed better and generalization was guaranteed by 10-fold cross-validation [15].

Developments in ensemble methods can be seen in the dynamic classifier, which combines multiple models and applies ant-colony optimization (ACO) to select features, achieving an accuracy of 94.13% and improved minority-class classification with random oversampling [16]. Comparative analyses of RF, XGBoost, SVM, and neural networks - enhanced through preprocessing steps such as SMOTE, PCA, and RFE - identified XGBoost as the top performer in F1-score and receiver operating characteristic area under the curve (ROC-AUC), while also providing interpretability through feature importance and SHAP analysis [17]. Hybrid ensemble models combining multiple ML algorithms across PROMISE datasets (CM1, JM1, KC1, PC1) demonstrated that a heterogeneous ensemble comprising K-Nearest Neighbor, GaussianNB, SVM, and neural networks outperformed other hybrids in accuracy, recall, precision, F1-score, and ROC-AUC, highlighting the advantages of hybridization for cross-project prediction [18].

The proposed classifier model will be a multi-classifier system that integrates the strengths of the most successful classifiers. Using 16 publicly available datasets from the NASA MDP repository within the PROMISE benchmark, experimental results confirm that significant improvements in SFP can be achieved with the assistance of LGBM, XGBoost, and voting classifiers in a multi-classifier approach [19].

Unsupervised Machine Learning in SFP

ML methods are widely applied in SDP to classify modules as defective or non-defective. However, SDP faces challenges such as redundant and correlated features, irrelevant attributes, missing values, and class imbalance, all of which reduce prediction reliability. Both supervised and unsupervised ML approaches have been used to address these issues, with unsupervised methods gaining increasing attention due to their ability to operate without labeled data or manual feature engineering [20]. Unsupervised learning has shown success in fields such as computer vision and natural language processing, and relevant techniques - including Apriori, ECLAT, FP-growth, PCA, and clustering (hierarchical and partition-based) - have been widely explored in recent SDP studies [21].

One of the most commonly used clustering strategies is K-means. An SDP model based on K-means and similarity measures (Cosine, Jaccard, hybrid) combined with a neural network classifier achieved an accuracy of 94.3% on object-oriented metrics [22]. Another hybrid method, which merged an autoencoder with K-means, reduced dimensionality and clustering error, achieving 96% accuracy, 93% precision, and 87% recall on NASA PROMISE datasets, while also reducing computation time [23].

Ensemble Learning in SFP

SDP is important for improving software quality in addition to reducing the cost of tests by identifying the defective modules during the early phase. There has been extensive experimentation with ensemble-based methods because they allow numerous classifiers to be used to enhance the predictive robustness of the method. A smart two-stage ensemble model based on RF, SVM, naïve Bayes, and artificial neural networks with optimization of the parameters using an iterative method and a voting process has shown better performance on seven NASA MDP datasets and has outperformed 20 state-of-the-art methods [24].

Additional comparative studies indicate that the ensemble and boosting-based models are still better than the traditional classifiers. Python-written experiments indicated that XGBoost had the highest accuracy, precision, recall, and F1-score on the Eclipse 2.0 dataset, and AdaBoost demonstrated a good performance on a variety of measures [25]. On the same note, Bayes Net together with C4.5, Multi-Layer Perceptron and RF applied to SDP exhibits better precision, recall, F1-score, and accuracy when combined in ensemble setting than when applied as individual classifiers, which explains the success of ensemble learning in SDP [26].

NASA software fault dataset is used to test the proposed approach after cleaning of data and correcting the imbalance in the classes with the help of SMOTE; this dataset includes 21 software metrics. The model with the highest accuracy is a finely adjusted RF classifier, which was modeled at 82.96%, with an F1 score of 89.53. Comparative studies have demonstrated that the ensemble procedures, such as AdaBoost and voting classifiers, provide high-quality predictive performance [27-29]. Despite the ability to model complex patterns effectively, DL models have high hyperparameter tuning demands; this limitation limits their practical use in SFP [30].

Deep Learning in SFP

The prediction of software faults has been a popular area of research in DL methods. To monitor progress and detect faults in development with the assistance of a project management tool, this study used three XP/TDD-oriented projects totaling over 472 KLOC. The data were used to evaluate five base classifiers and their ensembles (RF) using Apache ActiveMQ and Eclipse data. The study also included DL techniques such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), Bidirectional Long Short-Term Memory (Bi-LSTM), and hybrid Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM) models, which have shown strong performance on large-scale data. Results from an n-way ANOVA test confirmed that DL models are preferable to traditional ML techniques with 95% confidence for large-scale fault prediction [31].

In [32], Bi-LSTM combined with random oversampling and SMOTE was tested on PROMISE datasets, achieving accuracy scores of 90 to 99 and F-measures of 100, which exhibit high performance on Ant, Camel, Ivy, JEdit, Log4j, and Xerces. Recent research has also highlighted that DL models have the capacity to obtain syntactic and semantic information that is vital for accurate prediction. A performance comparison of LSTM across seven public datasets in [33] revealed that LSTM outperformed deep belief network, CNN, and RNN models in terms of precision, recall, and F1-score. Additionally, an ADASYN-based ML process using SelectKBest and Min-max normalization, together with a deep neural network classifier on the CM1 dataset, achieved an accuracy of 97.67 and high overall predictive performance [34].

Conventional SDP techniques rely on manual feature extraction and statistical models, which limits their scalability to large datasets with high complexity. Recent literature shows that DL networks, including CNNs, RNNs, LSTMs, and multilayer perceptrons, can automatically learn discriminative features and achieve higher predictive performance. Experiments on publicly available datasets such as PROMISE and BugHunter indicate that combining these techniques with feature selection and imbalance correction strategies such as SMOTE results in models that are more accurate and resilient compared to traditional ML approaches [35-38].

The imbalance of classes is also a significant issue in SDP, and in many cases, the learning results become biased. To address this problem, data-level, algorithm-level, and ensemble-based methods such as SMOTE, cost-sensitive learning, bagging, and boosting have been extensively studied [39-41]. SDP involves several connected steps; therefore, investigators must correlate the quality of the dataset, the models, the metrics used to evaluate the models, and the relevance to the purpose of the research, because no single model is optimal in all settings [40]. Experimental results also support the idea that optimized and ensemble-based ML models outperform traditional models in defect prediction [42-45].

Overall, the literature reviewed shows that no single ML method is consistently stable across all SFP situations. Table *1* presents a critical analysis of the literature. Dataset characteristics, preprocessing strategies, class imbalance, and evaluation metrics all play a major role in achieving high model performance. Therefore, further studies should focus on standard experimental designs, explanatory hybrid models, and the generalization of methods to other projects to improve the predictability and usability of SFP tools. Comparative analysis indicates that feature engineering and validation strategies, combined with enhancements in dataset quality, contribute more to SFP performance improvements than

classifier choice alone. While hybrid and DL models are reportedly more accurate, their scalability and interpretability remain major challenges. Additionally, the lack of cross-project validation and real-time deployment highlights a long-standing gap between research and industrial practice.

Analysis Dimension	Key Observations from Literature	Limitations Identified	Research Implications
Datasets	Mostly NASA and PROMISE datasets used	Poor diversity, small size, legacy	Need validated, modern, cross-project datasets
Prediction Type	Heavy reliance on binary fault prediction	Disregards the types and seriousness of faults	Multi-label and holistic prediction required
Feature Selection	Ensemble FS and metaheuristic FS enhance performance	High computation, dataset dependency	Exemplary and dynamic FS structures were needed
Modelling Approaches	Hybrid ML–DL models give better results than single model	Excessive Complication, less clarity	Improve accuracy with more explanation
Class Imbalance	SMOTE is mostly preferred	Noise and overfitting risks	State-of-the-art resampling and cost-effective learning
Evaluation Metrics	Accuracy is mostly used	Misleading under imbalance	Use MCC, G-mean
Cross-Project Prediction	Good intra-project performance	The performance reduces among projects	Requirement in Transfer learning and adaptation

TABLE 1: In-depth analysis of the existing literature

DL: Deep Learning; MCC: Matthews Correlation Coefficient; ML: Machine Learning; SMOTE: Synthetic Minority Over-sampling Technique

The primary aim of this paper is to provide an extensive overview of the literature and existing methodologies in SFP, which combines and harmonizes various different aspects related to SFP in different dimensions of the topic during the last two years. The contributions of this paper are summarized as follows:

This paper thoroughly summarizes recent SFP literature published in IEEE, ACM, Springer, and ScienceDirect, giving an up-to-date overview of ML developments in SFP between 2023 and 2025.

A comprehensive literature classification of SFP is done in terms of datasets, type of prediction, feature selection methods, modelling paradigms, and class imbalance management methods.

The ML classifiers, benchmark datasets, and evaluation metrics are thoroughly analyzed, showing the most important factors that affect the performance of SFP models and how to choose ML models based on the aspects of data-related, model-related and goal-oriented.

Key challenges at the data, model, DL, and universal levels are found, and solutions and future research directions are provided so as to assist in the development of robust and generalizable SFP models.

Review

Methodology

The proposed study follows a systematic literature review methodology, adhering to the reporting protocols outlined in the Preferred Reporting Items for Systematic Review and Meta-Analysis (PRISMA). Using this approach, we specifically examine and analyze the usefulness of ML algorithms in SFP. This involves identifying relevant studies, developing a search strategy, and applying eligibility criteria. A systematic literature review can be described as a structured data-gathering process focused on a specific subject, designed to ensure that the collected data meets established standards of relevance and quality while directly addressing the guiding research questions. In this process, published research such as conference proceedings, journal articles, book chapters, and other scholarly works is thoroughly examined to ensure the completeness and comprehensiveness of the review [6].

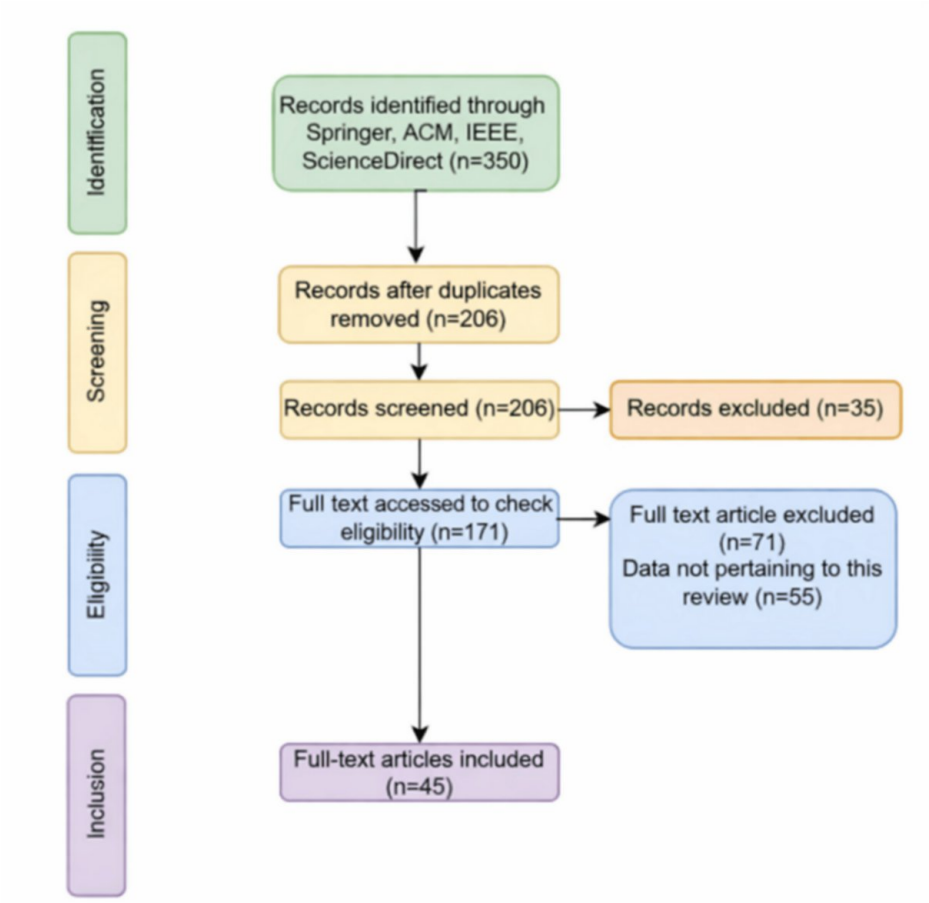

FIGURE 1: PRISMA literature review schematic

PRISMA: Preferred Reporting Items for Systematic Reviews and Meta-Analyses

PRISMA aims at selecting the most suitable papers to analyse. Hence, its recommendations offer a general structure for systematic reviews, meta-analysis, and reporting [9]. There are four fundamental stages in this methodology, and they include the identification phase, the screening phase, the eligibility phase, and the inclusion phase. These stages and processes are outlined in Figure *1*. The figure provides a graphical summary of a systematic review in a well-organized format, showing the papers, the number of records found, included, and rejected at each phase, and an explanation of why records were rejected.

Identification Criteria

This methodology aims to investigate how the various fault prediction algorithms can be applied to address issues of interpretability, absence of domain knowledge, and data quality. The primary aim is to increase the efficiency of software fault algorithms, as well as their reliability, particularly when it comes to algorithms implementing AI-based approaches such as ML and DL. Some of the issues in software systems that these techniques are expected to predict and identify include syntax, logical, run-time, design, and requirements faults.

Search Strategy Criteria

Springer, ACM, IEEE Xplore Digital Library, and ScienceDirect databases that were used to extract the research publications were screened by using the criteria. Peer reviewed publications on arbitrary years of publication of 2023-2025 were sampled to identify material related to software fault detection algorithms through the use of artificial intelligence. The search query was ((software fault prediction OR software defect prediction OR software error prediction) AND (artificial intelligence OR AI OR machine learning OR neural network)).

Eligibility Criteria

The database was searched to retrieve research papers, which were divided based on the inclusion and exclusion criteria. The software fault detection algorithms published between 2023 and 2025 were thoroughly reviewed. Table 2 presents the inclusion and exclusion criteria.

Criteria	Decision
The published studies in the given timeframe of the review (e.g., 2023-2025) are considered to offer the current relevance.	Inclusion
Studies should be directly coupled with the prediction of the software defects/faults or improve SFP models.	Inclusion
Only peer-reviewed journal articles are considered.	Inclusion
Articles in other languages other than English are filtered to ensure interpretability.	Exclusion
Research which does not specialize in software fault prediction, or machine learning	Exclusion

TABLE 2: Inclusion and exclusion criteria for PRISMA

PRISMA: Preferred Reporting Items for Systematic Reviews and Meta-Analyses; SFP: Software Fault Prediction

Discussion and analysis

Paper Collection Methodology and Results

The articles were selected from IEEE Xplore, Springer, ACM, and ScienceDirect employing the search query (Software Fault Detection, Machine Learning, Classification, and NASA PROMISE), and a total of 350 papers were retrieved from these sources. The papers that were initially accessed were further screened on the basis of eligibility and non-eligibility criteria.

Figure 2 highlights the continued applicability of SFP across various ML methods. The number of primary studies conducted in this field is shown on the y-axis, while the years of study are displayed on the x-axis. It is worth noting that research publications peaked in 2025, with around 17 studies, whereas 2023 had the fewest, with 12 studies in this field.

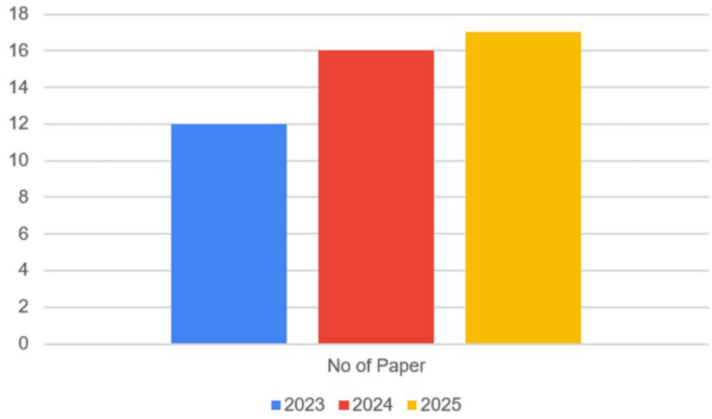

FIGURE 2: Temporal distribution of ML papers

ML: Machine Language

The articles were selected from IEEE Xplore, Springer, ACM, and ScienceDirect using the search query (Software Fault Detection, Machine Learning, Classification, and NASA PROMISE), and a total of 350 papers were retrieved from these sources. The papers initially accessed were then further screened based on eligibility and non-eligibility criteria. The systematic methodological selection of pertinent studies provides a high level of coverage of existing SFP methods. Based on the gathered literature, the classification of the most commonly used classifiers in SFP models is analyzed in the following section, and their effectiveness under various experimental conditions is investigated.

Popular Classifiers Used in SFP model

The researchers used more than 27 techniques and algorithms in the selected primary studies. These techniques were employed to compare and enhance prediction performance. All applied techniques are categorized into 10 classes in Figure 3. The analysis of the most widely used techniques for SFP is shown in Figure 4, which indicates that 76% of the techniques fall under Bayesian, decision tree, neural, kernel-based, and ensemble groups, indicating their strong and consistent performance.

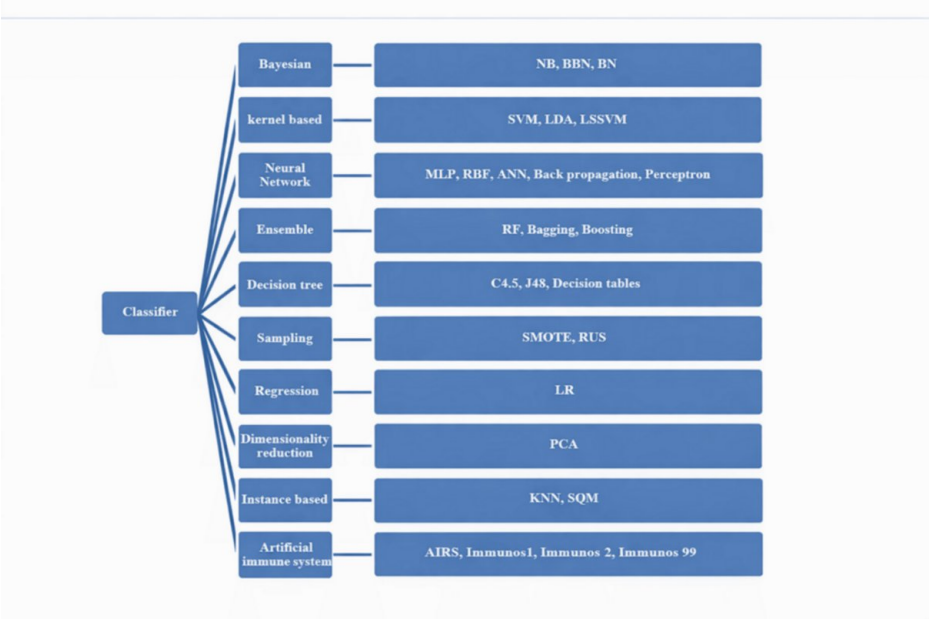

FIGURE 3: Techniques used in previous studies

ANN: Artificial Neural Network; BBN: Bayesian Belief Network; BN: Bayesian Network; KNN: K-Nearest Neighbors; LDA: Linear Discriminant Analysis; LR: Logistic Regression; LSSVM: Least Squares Support Vector Machine; MLP: Multi-Layer Perceptron; NB: Naive Bayes; PCA: Principal Component Analysis; RBF: Radial Basis Function; RF: Random Forest; RUS: Random Under-Sampling; SMOTE: Synthetic Minority Over-sampling Technique; SQM: Software Quality Management; SVM: Support Vector Machine

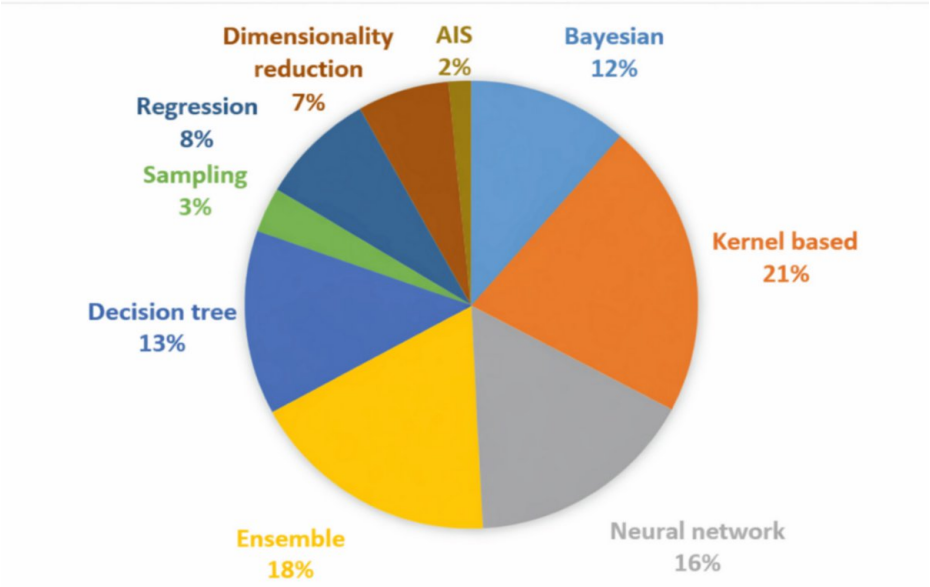

FIGURE 4: Distribution of techniques

Despite the different classifiers that have been suggested in SFP, the performance of each classifier varies

with the datasets and metrics that are used. Therefore, the discussion of commonly used datasets and metrics should be provided to make any meaningful comparisons of SFP models.

Dataset and Metrics

The authors refer to different software engineering repositories and datasets to create SFP and SDP models. Such databases are classified as public and private. Public datasets are free of charge, whereas private datasets are not readily available. These datasets often contain software metrics (features) and fault labels (target values) for all modules (e.g., class, file, function). The most commonly used datasets are NASA MDP, PROMISE, Eclipse, and AEEEM. In SFP, metrics serve as measurable indicators that help ML models identify patterns associated with defect-prone modules, thereby enabling early detection and improving overall software reliability. The metrics used in previous studies include Product metrics, Process metrics, and Object-Oriented metrics. Overall, during the analysis, it was observed that among all datasets used in SFP, NASA and PROMISE are the most widely used. The analysis of the most widely used datasets is shown in Figure 5. Furthermore, it was observed that among all metrics used in SFP, Product metrics are the most commonly used. The analysis of the most widely used metrics for SFP is shown in Figure 6.

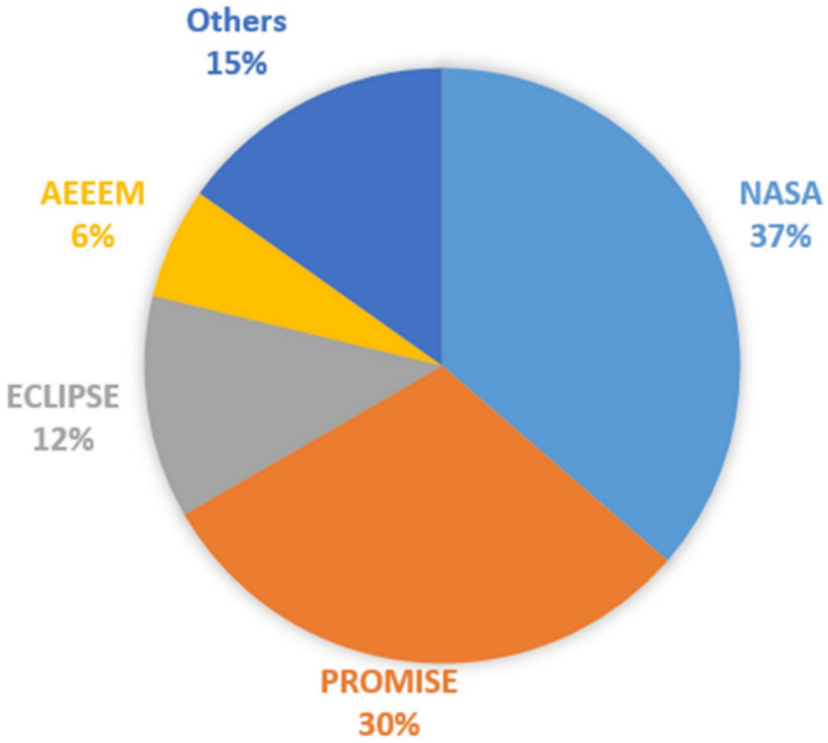

FIGURE 5: Distribution of dataset

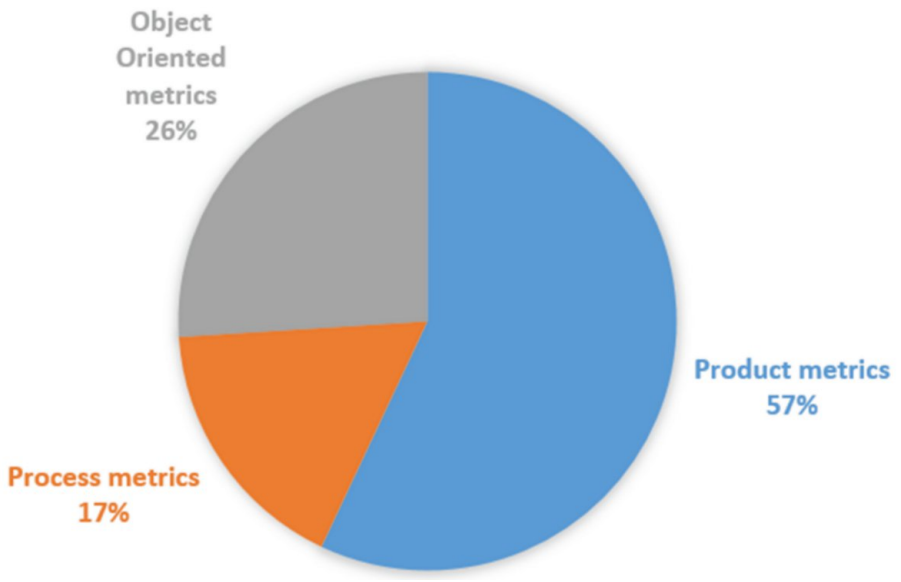

FIGURE 6: Distribution of metrics

Although datasets and software metrics have been the building blocks of SFP models, their usefulness should be tested using suitable evaluation measures. As a result, the choice of appropriate performance measurements is vital in the context of the truthful evaluation and comparison of the predictive performance of SFP models.

Evaluation Metrics

Researchers have used a number of performance measures to evaluate the workability of used or proposed defect prediction methods. However, most performance measures are founded on the components of a confusion matrix. Figure *7* defines the different metrics of evaluation used in SFP. During the analysis, it has been observed that some of the common measures used to determine the performance of defect prediction models include F-measure, area under ROC curve (AUC), recall, precision, and accuracy. Figure *8* shows that 89% of the studies of choice usually do. The other measures that have been employed to measure the performance of prediction models are the Matthews correlation coefficient, specificity, decision cost, G-mean, precision-recall curve, kappa statistic, and standard deviation error among other measures.

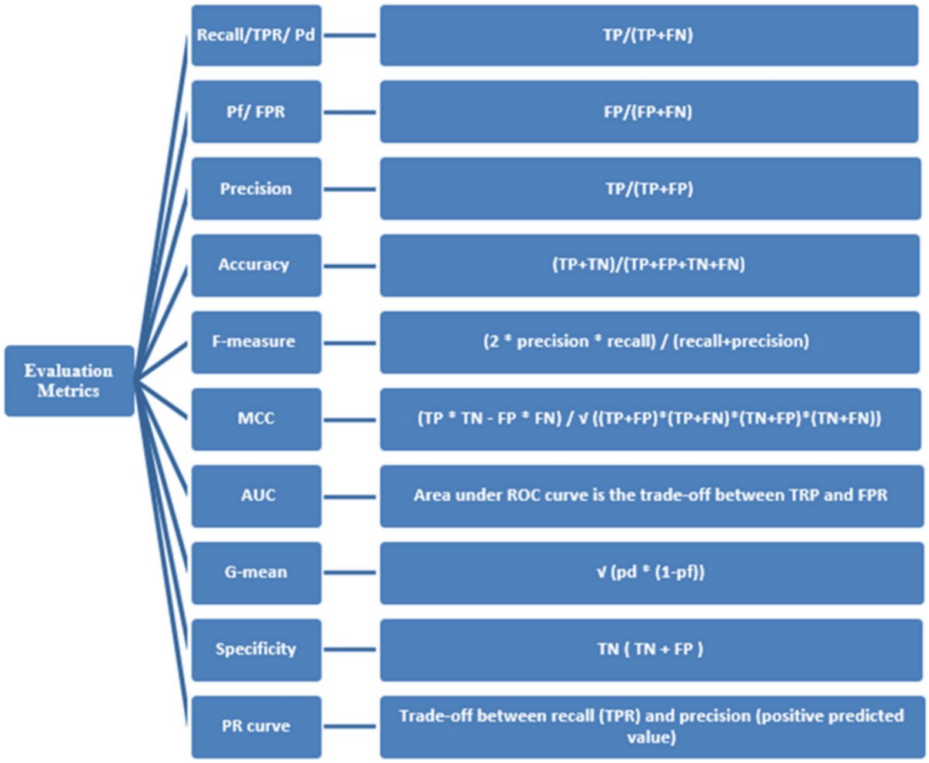

FIGURE 7: Evaluation metrics

AUC: Area Under the Curve; FN: False Negative; FP: False Positive; FPR: False Positive Rate; MCC: Matthews Correlation Coefficient; PR: Precision-Recall; ROC: Receiver Operating Characteristic; TN: True Negative; TP: True Positive; TPR: True Positive Rate

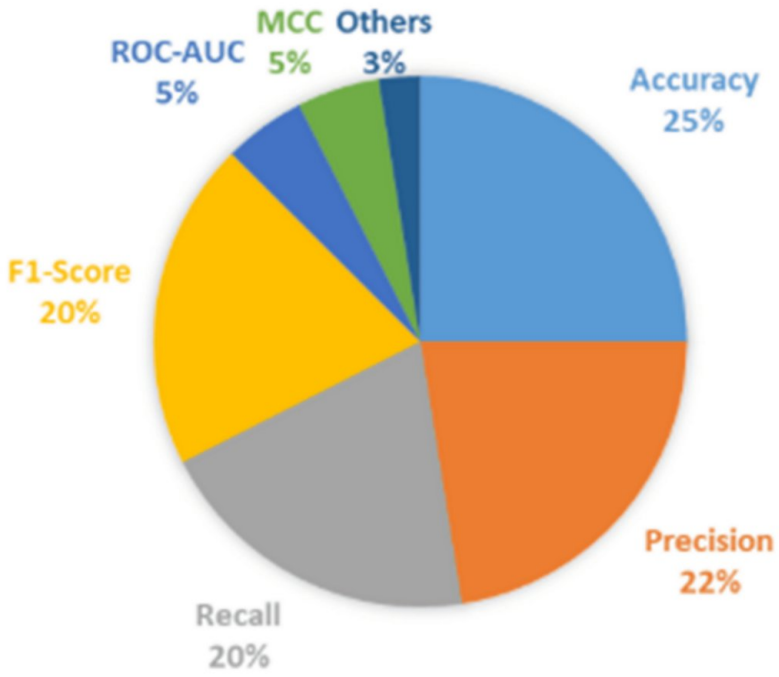

FIGURE 8: Distribution of evaluation metrics

MCC: Matthews Correlation Coefficient; ROC-AUC: Receiver Operating Characteristic-Area Under the Curve

Even though the evaluation metrics can offer a quantitative view or analysis towards the effectiveness of SFP models, the reported performance is also affected by a number of underlying factors that affect model reliability and generalization. Thus, it is critical to review the key factors that influence the performance of the SFP model.

Key factors affecting SFP model performance

The performance of SFP models is shaped by several interrelated factors that influence both model learning and generalization.

a) Overfitting and noisy dataset undermine reliability by causing the model to learn project-specific noise rather than meaningful defect patterns.

b) The choice and quality of software metrics such as complexity, coupling, and process measures directly affect predictive strength, as irrelevant or redundant metrics dilute useful information. Variations in software parameters, including project domain and development practices, further impact a model's ability to generalize across projects.

c) Evaluation methodology also plays a critical role; weak validation practices or imbalanced datasets can lead to misleading performance estimates, whereas robust methods like k-fold or cross-project validation produce more trustworthy results.

d) Among all factors, feature selection shows the strongest positive influence, as removing redundant and irrelevant features reduces overfitting and consistently improves accuracy, recall, and F-measure across studies.

Once the factors influencing the performance of SFP have been analyzed, researchers are then able to select machine learning models that are rationally selected in accordance to characteristics of datasets, model-based and goal-oriented.

Model selection criteria

According to the comparative analysis of the existing literature, the selection of a suitable ML model for SFP is mainly based on the following three major factors.

Data-related considerations: supervised learning is applied when there are small to moderately sized, labeled datasets; however, where unlabeled data are available, unsupervised learning can be used; deep learning models can be effectively used when there are extensive datasets available.

Model factors: Ensemble learning methods enhance prediction resilience by integrating several base learners, but DL models represent complex and nonlinear connections between the software metrics at the cost of higher computational expenditure.

Goal-related aspects: Supervised learning is mostly used to classify faults and predict defects precisely and unsupervised learning is used to detect the modules of the computer software that may be prone to fault in advance.

The model selection is also dictated by the computational resources and training complexity, especially in the adoption of DL models.

Challenges and its solutions

We classify these challenges into four broad categories, namely, model, data, DL, and universal challenges, and summarize the solutions to these problems that are present in the extant literature. The various challenges and its solutions are described in Table *3*.

Category	Challenges	Solutions
Model Challenges	Overfitting, Feature selection difficulty, Sequence models sensitive to hyperparameters	Apply meta-heuristic feature selection, Utilize self-attention mechanisms and optimized LSTMs
Data Challenges	Heterogeneous and small datasets, Limited defect samples and class imbalance, Noisy instances and incomplete code fragments	Use SMOTE, resampling and models suitable for small-data learning, Manual verification of defect labels; heuristic extraction of incomplete code
Deep Learning Challenges	High dependency on large labeled datasets, Difficulty in hyperparameter optimization, Need for high-performance hardware, Black-box nature of neural networks	Expand datasets, Apply automated hyperparameter optimization (Bayesian, grid search), Utilize GPUs/TPUs or cloud-based computation, Integrate explainability tools (SHAP, LIME)
Universal Challenges	Software quality and security concerns, Interpretability issues, Generalizability	Incorporate comments, commit logs and metadata, Improve feature learning via embeddings; use cost-sensitive learning Choose diverse, balanced datasets; evaluate across benchmarks

TABLE 3: Various challenges faced in SFP with its solutions

GPUs: Graphics Processing Units; LIME: Local Interpretable Model-agnostic Explanations; LSTMs: Long Short-Term Memory; SHAP: SHapley Additive exPlanations; SMOTE: Synthetic Minority Over-sampling TEchnique; SFP: Software Fault Prediction; TPUs: Tensor Processing Units

Conclusions

The current review provided an extensive analysis of the recent progress (2023-2025) of SFP through the use of machine learning, ensemble, and DL techniques. In the literature, it is evident that SFP will play a critical role in saving costs of software, as well as increasing quality, by detecting early modules likely to cause defects. It is revealed in the analysis that none of the models is always better than the rest, rather the quality of the data used, preprocessing techniques, feature selection, dealing with class imbalances, and the evaluation strategies all play a significant role in determining the performance of predictions. Hybrid and ensemble methods, especially those using tree-based learners and gradient boosting, are more accurate, and DL models have potential when using large and complex data, but do not easily support interpretability and scaling. Comparative studies show that SVMs and RF always score high in most cases on different sets of data. In general, the results underline the importance of more empirical studies to overcome current limitations in SFP and integrate cutting-edge experimental design, interpretable and comprehensible models, and workable datasets as opposed to focusing on the selection of classifiers, thus the necessity to close the gap between the academic research sphere and the world of feasible software engineering applications. Overall, this review gives a complete idea of the latest SFP trends and a background to stack the research directions in the future in terms of software quality assurance.

Additional Information

Author Contributions

All authors have reviewed the final version to be published and agreed to be accountable for all aspects of

the work.

Concept and design: Ruchika Aggarwal , Kamaljit Kaur

Acquisition, analysis, or interpretation of data: Ruchika Aggarwal , Kamaljit Kaur

Drafting of the manuscript: Ruchika Aggarwal , Kamaljit Kaur

Critical review of the manuscript for important intellectual content: Ruchika Aggarwal , Kamaljit Kaur

Supervision: Kamaljit Kaur

Disclosures

Conflicts of interest: In compliance with the ICMJE uniform disclosure form, all authors declare the following: **Payment/services info:** All authors have declared that no financial support was received from any organization for the submitted work. **Financial relationships:** All authors have declared that they have no financial relationships at present or within the previous three years with any organizations that might have an interest in the submitted work. **Other relationships:** All authors have declared that there are no other relationships or activities that could appear to have influenced the submitted work.

Data Availability Statements

The datasets (and/or code) supporting this study are available from the corresponding author upon reasonable request.

References

1. Mustaqeem M, Alam M, Mustajab S, Alshanketi F, Alam S, Shuaib M: Comprehensive bibliographic survey and forward-looking recommendations for software defect prediction: datasets, validation methodologies, prediction approaches, and tools. IEEE Access. 2025, 13:866-903. 10.1109/access.2024.3517419
2. Khalid A, Badshah G, Ayub N, Shiraz M, Ghouse M: Software defect prediction analysis using machine learning techniques. Sustainability. 2023, 15:5517. 10.3390/su15065517
3. Zaidi HZ, Ullah U, Arshad M, Aljuaid H, Rauf MA, Sarwar N, Sajid R: Machine learning approaches for software defect prediction. Applied Computational Intelligence and Soft Computing. 2025, 2025:1-29. 10.1155/acis/7933078
4. Li C, Li D, Li H, Wong WE, Zhao M: A systematic review of learning-based software defect prediction. Journal of Internet Technology. 2025, 26:501-511. 10.70003/160792642025072604009
5. Kumar S, Awasthi S: Artificial intelligence approaches for predicting software defects: a comprehensive review. International Journal of Research and Development in Applied Science and Engineering (IJRDASE). 2025, 25:1-4.
6. Ndlovu L, Cossa B, Makokoe C, et al.: Software fault detection algorithms using artificial intelligence: a review and classification. 2024 International Conference on Electrical, Computer and Energy Technologies (ICECET), Sydney, Australia. 2024, 1-6. 10.1109/ICECET61485.2024.10698379
7. Kumar R, Kaur K: A comparative analysis of techniques, datasets, feature selection methods, and evaluation metrics in software fault prediction. International Journal of Emerging Science and Engineering (IJESE). 2025, 13:25-41.
8. Rafique A, Bhatti S: Machine learning techniques for software fault prediction: A distinctive systematic literature review. International Conference on Engineering, Natural Sciences, and Technological Developments (ICENSTED 2024). 2024, 855-860.
9. Kaur G, Pruthi J, Gandhi P: Machine learning based Software Fault Prediction models. Karbala International Journal of Modern Science. 2023, 9:9. 10.33640/2405-609x.3297
10. Pathak H: AI models for software defect prediction: comparative study. Innovative Journal of Applied Science. 2025, 2:28. 10.70844/ijas.2025.2.28
11. Sunil A, Sahu RK, Karsoliya S: Software defect prediction using supervised machine learning: a systematic literature review. International Journal of Advanced Research and Multidisciplinary Trends (IJARMT). 2025, 2:80-95.
12. Cauvery G, Suresh D: Software defect prediction using machine learning techniques. Data Analytics and Artificial Intelligence. 2023, 3:30-33.
13. Sushma, Gr P: Software bug prediction using supervised machine learning algorithms. International Journal of Scientific Development and Research (IJSDR). 2023, 8:358-364.
14. Nuruddin Siswantoro MZF, Yuhana UL: Software defect prediction based on optimized machine learning models: a comparative study. Teknika. 2023, 12:166-172. 10.34148/teknika.v12i2.634
15. Elshamy N, AbouElenen A, Elmougy S: Automatic detection of software defects based on machine learning. International Journal of Advanced Computer Science and Applications. 2023, 14:353-364. 10.14569/ijacsa.2023.0140340
16. Kaliraj S, Sahasranth VGP, Sivakumar V: A holistic approach to software fault prediction with dynamic classification. Automated Software Engineering. 2024, 31:70. 10.1007/s10515-024-00467-4
17. Durga, SinhaA: Enhancing software reliability through intelligent fault prediction using machine learning. International Journal of Scientific Research in Computer Science Engineering and Information Technology. 2025, 11:945-956. 10.32628/cseit25113376

18. Kumar H, Saxena V: Software defect prediction using hybrid machine learning techniques: a comparative study. Journal of Software Engineering and Applications. 2024, 17:155-171. 10.4236/jsea.2024.174009
19. Gupta N, Sinha RR: A novel developed supervised machine learning system for classification and prediction of software faults using NASA dataset. International Journal on Recent and Innovation Trends in Computing and Communication. 2023, 11:715-729. 10.17762/ijritcc.v11i10s.7710
20. Batool A: Software defect prediction using clustering: a comprehensive literature review. International Journal of Computations, Information and Manufacturing (IJCIM). 2023, 3:57-65.
21. Bojja RR: A comparative study of supervised and unsupervised learning approaches. International Multidisciplinary Research Journal Reviews (IMRJR). 2025, 2:80-88. 10.17148/IMRJR.2025.020411
22. Arya A, Malik SK: Software fault prediction using K-mean-based machine learning approach. International Journal of Performability Engineering. 2023, 19:133-133. 10.23940/ijpe.23.02.p6.133143
23. Arasteh B, Golshan S, Shami S, Kiani F: Sahand: a software fault-prediction method using autoencoder neural network and K-means algorithm. Journal of Electronic Testing. 2024, 40:229-243. 10.1007/s10836-024-06116-8
24. Ali M, Mazhar T, Arif Y, et al.: Software defect prediction using an intelligent ensemble-based model. IEEE Access. 2024, 12:20376-20395. 10.1109/access.2024.3358201
25. Sharma A, Amrendra K, Ranjan P: Comparative analysis of ensemble classifiers over machine learning classifiers for early software quality prediction. Proceedings of the Recent Advances in Artificial Intelligence for Sustainable Development (RAISD 2025). Atlantis Press, The Netherlands; 2025. 196:351-366. 10.2991/978-94-6463-787-8_29
26. Mehta A, Batra I, Fergina A: Boosting software fault prediction accuracy with ensemble learning. Engineering Proceedings. 2025, 107:63. 10.3390/engproc2025107063
27. Thomas NS, Kaliraj S: An improved and optimized random forest based approach to predict the software faults. SN Computer Science. 2024, 5:530. 10.1007/s42979-024-02764-x
28. Siddika A, Begum M, Al Farid F, Uddin J, Karim HA: Enhancing software defect prediction using ensemble techniques and diverse machine learning paradigms. Eng. 2025, 6:161. 10.3390/eng6070161
29. Ali M, Mazhar T, Al-Rasheed A, Shahzad T, Yasin Ghadi Y, Amir Khan M: Enhancing software defect prediction: a framework with improved feature selection and ensemble machine learning. PeerJ Computer Science. 2024, 10:e1860. 10.7717/peerj-cs.1860
30. Mohammed A, Kora R: A comprehensive review on ensemble deep learning: Opportunities and challenges. Journal of King Saud University Computer and Information Sciences. 2023, 35:757-774. 10.1016/j.jksuci.2023.01.014
31. Borandag E: Software fault prediction using an RNN-based deep learning approach and ensemble machine learning techniques. Applied Sciences. 2023, 13:1639. 10.3390/app13031639
32. Khleel NAA, Nehéz K: Software defect prediction using a bidirectional LSTM network combined with oversampling techniques. Cluster Computing. 2024, 27:3615-3638. 10.1007/s10586-023-04170-z
33. Phuong HTM, Ngan DTK, Binh NT: A comparative study of deep learning techniques in software fault prediction. The University of Danang - Journal of Science and Technology. 2024, 22:1-5. 10.31130/ud-jst.2024.266e
34. Gadiraju RK: A novel machine learning method for fault prediction and reliability in software systems. International Journal of Scientific Research in Science, Engineering and Technology. 2025, 12:1226-1238. 10.32628/ijsrset2512163
35. Liang Q: Research on software defect prediction model based on deep learning. Highlights in Science, Engineering and Technology. 2024, 122:23-29. 10.54097/y0w76b47
36. Alkaberi W, Assiri F: Predicting the number of software faults using deep learning. Engineering, Technology & Applied Science Research. 2024, 14:13222-13231. 10.48084/etasr.6798
37. Phung K, Aydin ME, Ogunshile E: Deep learning architectures for software fault prediction: the impact of error-type metrics and class imbalance. Concurrency and Computation: Practice and Experience. 2026, 38:e70472. 10.1002/cpe.70472
38. Modanlou Jouybari M, Tajary A, Fateh M, Abolghasemi V: A novel deep neural network structure for software fault prediction. PeerJ Computer Science. 2024, 10:e2270. 10.7717/peerj-cs.2270
39. Mehta A, Kaur N, Kaur A: A review of software fault prediction techniques in class imbalance scenarios. International Journal of Performability Engineering. 2025, 21:123-130. 10.23940/ijpe.25.03.p1.123130
40. Zhang Y, Liu N: Investigation and research on several key issues of software defect prediction. IET Software. 2025, 2025:1-28. 10.1049/sfw2/6615496
41. Raj A, Chavan DM, Agarwal P: Enhancing software fault prediction through data balancing techniques and machine learning. IAES International Journal of Artificial Intelligence (IJ-AI). 2025, 14:4787-4801. 10.11591/ijai.v14.i6.pp4787-4801
42. Tufail S, Riggs H, Tariq M, Sarwat AI: Advancements and challenges in machine learning: a comprehensive review of models, libraries, applications, and algorithms. Electronics. 2023, 12:1789. 10.3390/electronics12081789
43. Kumar A, Ansari MA: The systematic review study of significance of machine learning techniques in software defect prediction. International Journal of Engineering and Technology Research (IJETR). 2024, 9:10-25.
44. Kumar SA, Prasanna BS, Mani AG, Deepika G, Reddy BVAK: Software bug prediction using machine learning. International Journal of Scientific Research in Engineering and Management (IJSREM). 2024, 08:1-5. 10.55041/ijsrem31425
45. Lafi M, Farhan KA, Abusukhon A: Machine learning model for fault prediction in object-oriented systems. 2025 12th International Conference on Information Technology (ICIT), Amman, Jordan. 2025, 644-646. 10.1109/ICIT64950.2025.11049103