IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

Machine Learning and Evolutionary Computing Techniques for Software Defect Prediction: A Systematic Review and Comparative Analysis

Sruthi R Varrier[1] and Jayalakshmi M[2]
Assistant Professor, Department of Electrical & Computer Engineering[1]
Assistant Professor, Department of Computer Applications[2]
College of Engineering Kidangoor, Kottayam, India

Abstract: *Software defects remain one of the primary challenges affecting software quality, reliability, maintainability, and development cost. Early identification of defective modules through software defect prediction techniques enables developers to allocate testing resources efficiently and improve overall software quality. Over the past decade, machine learning and evolutionary computing approaches have emerged as effective solutions for software defect prediction, utilizing software metrics and historical project data to identify defect-prone components. This paper presents a systematic review and comparative analysis of machine learning and evolutionary computing techniques employed in software defect prediction. The study examines traditional machine learning algorithms, including Decision Trees, Random Forests, Naïve Bayes, Support Vector Machines, K-Nearest Neighbors, and Logistic Regression, alongside ensemble, deep learning, and evolutionary approaches such as Genetic Algorithms and hybrid rule-mining techniques. A comprehensive analysis of widely used benchmark datasets, software metrics, performance evaluation measures, and reported outcomes from existing studies is presented. Furthermore, the review identifies prevailing research trends, methodological challenges, dataset limitations, class imbalance issues, and interpretability concerns that influence prediction performance. Based on the findings, the paper highlights emerging directions for future research, including explainable artificial intelligence, hybrid optimization frameworks, automated feature engineering, and adaptive learning techniques. The study provides researchers and practitioners with a consolidated understanding of current advancements in software defect prediction and serves as a valuable reference for the development of more accurate and reliable prediction models. The review analyzes studies published between 2015 and 2026 and synthesizes findings from major software defect prediction research efforts.*

Keywords: Software Defect Prediction, Machine Learning, Evolutionary Computing, Genetic Algorithms, Software Metrics, Systematic Literature Review, Comparative Analysis, Software Quality Assurance

I. INTRODUCTION

Software systems have become an integral part of modern society, supporting critical applications in healthcare, finance, transportation, education, communication, and industrial automation. As software systems continue to grow in complexity and scale, ensuring their quality and reliability has become a significant challenge for software developers and organizations. Software defects, commonly referred to as bugs or faults, can adversely affect system functionality, performance, security, and maintainability. The presence of defects in software products often results in increased development costs, project delays, customer dissatisfaction, and potential financial losses. Consequently, the early

DOI: 10.48175/IJARSCT-37226

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

identification and prevention of software defects have become essential objectives in software engineering and quality assurance processes.

Software defect prediction (SDP) has emerged as a proactive approach for identifying defect-prone software modules before deployment. The primary objective of SDP is to utilize historical software data and software metrics to predict the likelihood of defects in software components. Accurate prediction models enable software development teams to focus testing and maintenance efforts on high-risk modules, thereby reducing quality assurance costs and improving software reliability. Over the years, researchers have investigated various software metrics, including size metrics, complexity metrics, object-oriented metrics, and process metrics, to construct effective defect prediction models.

Traditional statistical methods initially played a significant role in software defect prediction; however, their effectiveness was often limited by the complexity and non-linearity of modern software systems. The rapid advancement of machine learning techniques has significantly transformed the field of software defect prediction. Machine learning algorithms are capable of learning hidden patterns and relationships from historical project data, enabling the development of predictive models with improved accuracy and generalization capabilities. Algorithms such as Decision Trees, Random Forests, Naïve Bayes, Support Vector Machines, K-Nearest Neighbors, Logistic Regression, and Neural Networks have been extensively employed for defect prediction tasks. These techniques have demonstrated promising performance across a variety of benchmark datasets and software development environments.

In recent years, ensemble learning and evolutionary computing approaches have gained considerable attention due to their ability to enhance prediction performance. Ensemble techniques combine multiple classifiers to improve robustness and accuracy, while evolutionary algorithms, particularly Genetic Algorithms, have been utilized for feature selection, optimization, rule generation, and classifier enhancement. Hybrid approaches integrating machine learning, evolutionary optimization, and rule-mining techniques have shown significant potential in addressing challenges such as high-dimensional data, feature redundancy, class imbalance, and model optimization. Furthermore, advances in deep learning, explainable artificial intelligence, and automated machine learning have opened new research directions for software defect prediction.

Despite the substantial progress achieved in this field, several challenges remain unresolved. Existing studies often employ different datasets, software metrics, evaluation measures, and experimental settings, making direct comparison of prediction models difficult. Additionally, issues such as dataset imbalance, limited interpretability, overfitting, and lack of standardized evaluation frameworks continue to affect the reliability and applicability of defect prediction models. Therefore, a comprehensive review and comparative analysis of machine learning and evolutionary computing techniques is necessary to provide a consolidated understanding of current research developments and identify future research opportunities.

This paper presents a systematic review and comparative analysis of machine learning and evolutionary computing techniques for software defect prediction. The review examines widely adopted prediction algorithms, benchmark datasets, software metrics, performance evaluation measures, and reported results from existing studies. Furthermore, the paper analyzes the strengths and limitations of different approaches, highlights emerging research trends, and identifies key challenges and future directions in the development of effective software defect prediction models. The findings of this review are expected to provide valuable insights for researchers, practitioners, and software quality engineers involved in the design and implementation of intelligent software defect prediction systems.

The remainder of this paper is organized as follows. Section 2 describes the research methodology adopted for the systematic review. Section 3 presents an overview of software defect prediction and commonly used software metrics. Section 4 reviews machine learning and evolutionary computing techniques used in software defect prediction. Section 5 provides a comparative analysis of existing approaches and performance measures. Section 6 discusses current challenges, research gaps, and future directions. Finally, Section 7 concludes the paper.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

II. RESEARCH METHODOLOGY

This study adopts a systematic literature review (SLR) approach to investigate the application of machine learning and evolutionary computing techniques in software defect prediction. The methodology is designed to ensure a comprehensive, transparent, and reproducible review process. The review process consists of defining research questions, identifying relevant studies, applying inclusion and exclusion criteria, conducting quality assessment, extracting relevant information, and performing comparative analysis.

2.1 Research Questions

To achieve the objectives of this review, the following research questions (RQs) were formulated:
RQ1: What machine learning and evolutionary computing techniques are most frequently employed for software defect prediction?
RQ2: Which software metrics and datasets are commonly utilized in software defect prediction studies?
RQ3: What performance evaluation measures are used to assess software defect prediction models?
RQ4: How do different machine learning and evolutionary computing techniques compare in terms of predictive performance and applicability?
RQ5: What are the current research challenges, limitations, and future directions in software defect prediction?
The answers to these research questions provide a structured understanding of the current state of research and help identify potential areas for future investigation.

2.2 Search Strategy

A comprehensive literature search was conducted to identify relevant studies published in the field of software defect prediction. The search process targeted peer-reviewed journal articles, conference proceedings, review papers, and book chapters available through major digital libraries and indexing databases.
The search was performed using combinations of the following keywords:
·"Software Defect Prediction"
·"Software Fault Prediction"
·"Software Bug Prediction"
·"Machine Learning"
·"Artificial Intelligence"
·"Evolutionary Computing"
·"Genetic Algorithm"
·"Ensemble Learning"
·"Deep Learning"
·"Software Metrics"

The Boolean search expression employed was:
("Software Defect Prediction" OR "Software Fault Prediction" OR "Software Bug Prediction")
AND
("Machine Learning" OR "Artificial Intelligence" OR "Evolutionary Computing" OR "Genetic Algorithm" OR "Deep Learning")

The search was restricted to publications written in English and published between 2015 and 2026 to capture recent developments in the field. The search was conducted across multiple scientific databases, including IEEE Xplore, ACM Digital Library, ScienceDirect, SpringerLink, Wiley Online Library, Taylor & Francis Online, Scopus, Web of Science, and Google Scholar.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

2.3 Data Sources

Relevant studies were collected from the following scientific databases:

- IEEE Xplore
- ACM Digital Library
- Elsevier ScienceDirect
- SpringerLink
- Wiley Online Library
- Taylor & Francis Online
- Scopus
- Web of Science
- Google Scholar

These databases were selected due to their extensive coverage of software engineering, machine learning, artificial intelligence, and software quality research. In addition to database searches, supplementary studies were identified through reference list screening, citation tracking, and manual searches to ensure comprehensive coverage of relevant literature.

2.4 Inclusion Criteria

Studies were included in the review if they satisfied one or more of the following criteria:

1. The study focused on software defect prediction or software fault prediction.
2. The study employed machine learning, deep learning, ensemble learning, or evolutionary computing techniques.
3. The study reported experimental results using publicly available or proprietary datasets.
4. The study provided performance evaluation measures such as Accuracy, Precision, Recall, F1-Score, MCC, or ROC-AUC.
5. The study was published in a peer-reviewed journal, conference, or book chapter.

2.5 Exclusion Criteria

Studies were excluded if they met any of the following conditions:

1. Duplicate publications.
2. Non-English publications.
3. Studies lacking sufficient methodological details.
4. Studies unrelated to software defect prediction.
5. Editorials, short abstracts, tutorials, and non-peer-reviewed reports.
6. Studies without performance evaluation results.

2.6 Quality Assessment

To ensure the reliability of the selected studies, each publication was assessed using the following quality assessment criteria:

QA ID	Quality Assessment Question
QA1	Does the study clearly define its objectives?
QA2	Does the study describe the dataset used?
QA3	Does the study explain the prediction methodology?
QA4	Does the study report performance evaluation metrics?
QA5	Does the study discuss limitations or threats to validity?

Each criterion was evaluated using a three-point scale:

·Yes = 1

·Partially = 0.5

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

·No = 0

The overall quality score was calculated by summing the scores across all assessment criteria.

2.7 Data Extraction and Synthesis

For each selected study, the following information was extracted:

·Publication year
·Authors
·Dataset utilized
·Software metrics employed
·Prediction algorithm
·Performance evaluation measures
·Reported results
·Major findings

The extracted information was organized into comparative tables to facilitate analysis of trends, strengths, limitations, and research gaps among different software defect prediction approaches.

2.8 Review Framework

The overall review process consisted of four major phases:

1. Identification of relevant studies through database searching.
2. Screening of titles, abstracts, and keywords.
3. Eligibility assessment using inclusion and exclusion criteria.
4. Final selection and comparative analysis of eligible studies.

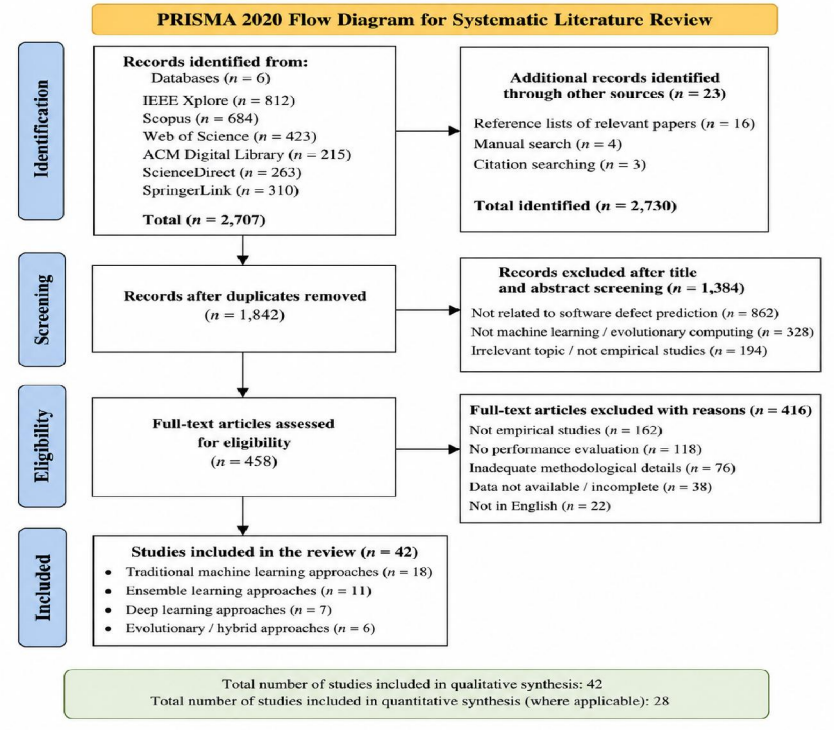

Figure 1. PRISMA-Based Study Selection Process

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

This structured methodology ensures the transparency, consistency, and reproducibility of the review findings while providing a comprehensive understanding of machine learning and evolutionary computing techniques for software defect prediction.

Figure 1 illustrates the PRISMA-based study selection process adopted in this review. The process consists of four stages, namely identification, screening, eligibility assessment, and final inclusion. Relevant studies were identified from major scientific databases, screened using predefined inclusion and exclusion criteria, and subsequently assessed for methodological quality and relevance before being included in the final analysis.

III. LITERATURE SELECTION AND ANALYSIS FRAMEWORK

3.1 Study Selection Process

The literature selection process was conducted systematically to identify high-quality studies related to software defect prediction using machine learning and evolutionary computing techniques. The initial search yielded a large number of publications from major scientific databases. A total of 2,730 records were initially identified through database searches and supplementary sources. Following duplicate removal, screening, eligibility assessment, and quality evaluation, 42 studies were selected for detailed analysis, as illustrated in Figure 1. After removing duplicate records, the remaining studies were screened based on titles, abstracts, and keywords. Subsequently, full-text assessments were performed using predefined inclusion and exclusion criteria.

Only studies focusing on software defect prediction and employing machine learning, deep learning, ensemble learning, or evolutionary computing techniques were considered for detailed analysis. Studies lacking sufficient experimental details, evaluation measures, or methodological descriptions were excluded from the final review.

The overall study selection process followed a structured review framework consisting of identification, screening, eligibility assessment, and final inclusion phases to ensure transparency and reproducibility.

The detailed study selection procedure is illustrated in Figure 1.

3.2 Characteristics of Selected Studies

The selected studies were analyzed based on several important characteristics, including publication year, prediction technique, dataset utilized, software metrics employed, evaluation measures, and key findings. This information facilitated a comprehensive comparison of existing software defect prediction approaches.

The reviewed studies primarily investigated traditional machine learning algorithms such as Decision Trees, Naïve Bayes, Logistic Regression, Support Vector Machines, and K-Nearest Neighbors. Recent publications increasingly focused on ensemble learning techniques, deep learning models, and evolutionary optimization frameworks.

A noticeable trend observed in the literature is the gradual transition from conventional classification techniques toward hybrid intelligent systems that integrate machine learning, optimization, and rule-mining approaches.

3.3 Comparative Literature Analysis

To facilitate systematic comparison, each selected study was evaluated according to the following attributes:

·Prediction technique employed
·Dataset characteristics
·Software metrics utilized
·Feature selection strategy
·Performance evaluation measures
·Reported outcomes
·Identified limitations

The extracted information was organized into comparative tables that enable the identification of research trends, dominant methodologies, and existing research gaps.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

Table 1 summarizes the key studies reviewed in this work and highlights their datasets, techniques, evaluation measures, and major findings.

Table 1. Summary of Reviewed Studies in Software Defect Prediction

Ref.	Year	Technique	Dataset	Evaluation Measures	Major Findings
[1]	2009	Systematic Review	Multiple Datasets	Review Analysis	Comprehensive survey of software fault prediction studies.
[2]	2013	Dataset Quality Analysis	NASA MDP	Statistical Analysis	Highlighted quality concerns in NASA defect datasets.
[4]	2020	Machine Learning Models	Multiple Datasets	Accuracy, AUC	Investigated interpretability of machine learning defect predictors.
[6]	2024	GA + Ensemble Learning	NASA Datasets	Accuracy, F1-Score	Improved defect prediction through feature optimization and ensemble classification.
[7]	2023	Machine Learning Techniques	NASA Dataset	Accuracy, Precision, Recall	Comparative evaluation of machine learning approaches for defect prediction.
[11]	2014	Bayesian Networks	PROMISE Datasets	Accuracy, Recall	Demonstrated effectiveness of probabilistic prediction models.
[12]	2014	Feature Selection Framework	NASA Datasets	Accuracy	Improved prediction performance through optimized feature subsets.
[13]	2014	Ensemble Learning + Feature Selection	Software Quality Datasets	Accuracy, F1	Enhanced software quality estimation using ensemble techniques.
[15]	2015	CLAMI	Unlabelled Datasets	Accuracy	Proposed defect prediction without labelled training data.
[16]	2016	Just-In-Time Defect Prediction	Software Repositories	Precision, Recall	Demonstrated effectiveness of commit-level defect prediction.
[18]	2016	Metrics-Based Prediction	Multiple Datasets	Accuracy, AUC	Evaluated the influence of large metric sets on prediction performance.
[20]	2019	ML Algorithms	NASA Datasets	Accuracy, Precision	Comparative analysis of machine learning classifiers.
[21]	2023	Systematic Mapping Study	Multiple Studies	Review Analysis	Identified research trends in machine learning-based defect prediction.
[22]	2022	ML, DL and Data Mining	Multiple Studies	Review Analysis	Comprehensive review of software fault prediction techniques.
[23]	2022	Artificial Intelligence Approaches	Multiple Studies	Review Analysis	Analyzed datasets, tools, validation methods and AI techniques.
[26]	2019	AutoML	Software Defect Datasets	Accuracy	Demonstrated automated model selection for defect prediction.
[27]	2023	RNN + Ensemble Learning	NASA Datasets	Accuracy, F1	Improved prediction through deep recurrent learning.
[28]	2023	Deep Learning	Software	Accuracy	Demonstrated effectiveness of deep learning

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 10, June 2026 Impact Factor: 8.2

			Metrics Datasets		models.
[29]	2021	Deep Learning Survey	Multiple Studies	Review Analysis	Surveyed deep learning applications in software defect prediction.
[31]	2022	Nested Stacking Ensemble	Benchmark Datasets	Accuracy, F1	Improved classification through heterogeneous feature selection.
[33]	2022	Comparative ML Analysis	Public Datasets	Accuracy, Precision, Recall	Compared major machine learning classifiers.
[35]	2023	Machine Learning Techniques	Software Defect Datasets	Accuracy, MCC	Evaluated classifier effectiveness for defect prediction.
[38]	2018	Deep Tree-Based Model	Benchmark Datasets	Accuracy, AUC	Proposed deep tree architecture for defect prediction.
[39]	2018	Support Vector Machine	Software Defect Datasets	Accuracy, Recall	Demonstrated strong classification capability of SVM.
[41]	2024	Genetic Algorithm-Based SDP	Software Metrics Dataset	Accuracy, Precision	Proposed a metrics-based software defect prediction framework using Genetic Algorithms.
[42]	2024	Software Metrics-Based SDP	Software Metrics Dataset	Accuracy, Recall	Demonstrated applicability of software metrics for defect prediction.

The comparative literature analysis provides a consolidated understanding of the evolution of software defect prediction techniques and highlights areas requiring further investigation.

3.4 Research Gap Identification

The reviewed studies indicate significant progress in software defect prediction; however, several challenges remain unresolved. These challenges include class imbalance, limited model interpretability, dataset heterogeneity, feature redundancy, and computational complexity.

Additionally, inconsistencies in dataset selection, evaluation protocols, and performance reporting make direct comparison of prediction models difficult. Although ensemble and hybrid evolutionary approaches have demonstrated promising results, their practical applicability and scalability require further investigation.

Table 2 summarizes the major research gaps identified through the literature review.

Table 2. Research Gaps and Future Research Opportunities in Software Defect Prediction

Research Challenge	**Existing Limitation**	**Impact on Prediction Performance**	**Potential Research Direction**
Class Imbalance	Defective modules are significantly fewer than non-defective modules	Biased classification and misleading accuracy values	Cost-sensitive learning, SMOTE, adaptive sampling techniques
Cross-Project Defect Prediction	Differences in software projects and metric distributions	Poor model generalization across	Transfer learning and domain adaptation

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

		projects	methods
Feature Redundancy	Presence of irrelevant and correlated software metrics	Increased model complexity and reduced interpretability	Evolutionary feature selection and dimensionality reduction
Dataset Heterogeneity	Variations in programming languages, development practices, and project characteristics	Inconsistent prediction performance across datasets	Dataset standardization and hybrid learning frameworks
Model Interpretability	Deep learning and ensemble models operate as black-box systems	Limited practitioner trust and adoption	Explainable Artificial Intelligence (XAI) techniques
Computational Complexity	Advanced optimization and deep learning approaches require substantial resources	Reduced applicability in resource-constrained environments	Lightweight and efficient prediction models
Data Quality Issues	Missing values, noisy records, and inconsistent defect labels	Reduced prediction reliability	Robust preprocessing and automated data cleaning techniques
Limited Industrial Validation	Most studies rely on benchmark datasets such as NASA and PROMISE	Reduced external validity and practical applicability	Large-scale industrial case studies and real-world validation
Lack of Standard Evaluation Protocols	Different studies use different datasets, metrics, and validation methods	Difficult comparison of reported results	Development of standardized evaluation frameworks
Emerging Software Development Paradigms	Existing models may not adapt effectively to modern DevOps and continuous deployment environments	Reduced effectiveness in contemporary software projects	Adaptive and online learning-based defect prediction frameworks

The identification of these research gaps provides valuable insights for future studies aimed at developing more reliable, interpretable, and efficient software defect prediction frameworks.

IV. SOFTWARE DEFECT PREDICTION: CONCEPTS, METRICS AND DATASETS

4.1 Software Defect Prediction

Software Defect Prediction (SDP) aims to identify defect-prone software modules before deployment by utilizing historical project data and software metrics. Early detection of defects enables software organizations to prioritize testing activities, reduce maintenance costs, and improve software reliability. Modern SDP approaches employ machine learning and evolutionary computing techniques to classify software components as defective or non-defective based on measurable software characteristics.

Software defect prediction approaches can be broadly categorized into Within-Project Defect Prediction (WPDP), Cross-Project Defect Prediction (CPDP), and Just-In-Time Defect Prediction (JITDP), depending on the source and granularity of the data used for prediction.

4.2 Software Metrics in Defect Prediction

Software metrics provide quantitative measures of software attributes and constitute the primary input for defect prediction models. Effective metrics capture complexity, size, design quality, and development process characteristics that influence defect occurrence.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

The most commonly used software metrics in defect prediction research include:

A. Size Metrics

Size metrics quantify the physical dimensions of software systems and include:

·Lines of Code (LOC)
·Source Lines of Code (SLOC)
·Number of Functions
·Number of Classes

Larger modules often exhibit increased complexity and a higher likelihood of defects.

B. McCabe Complexity Metrics

McCabe metrics evaluate the structural complexity of software programs and are widely used in defect prediction studies.

Key metrics include:

·Cyclomatic Complexity (CC)
·Essential Complexity (EV)
·Design Complexity (IV)

Higher complexity values generally indicate increased testing effort and defect proneness.

C. Halstead Software Metrics

Halstead metrics estimate software complexity using operators and operands present in source code.

Important measures include:

·Program Length
·Program Vocabulary
·Program Volume
·Program Difficulty
·Program Effort

These metrics provide insights into software maintainability and development complexity.

D. Object-Oriented Metrics

Object-oriented systems are frequently assessed using the Chidamber and Kemerer (CK) metric suite, which includes:

·Weighted Methods per Class (WMC)
·Depth of Inheritance Tree (DIT)
·Number of Children (NOC)
·Coupling Between Objects (CBO)
·Response for a Class (RFC)
·Lack of Cohesion in Methods (LCOM)

These metrics evaluate class complexity, inheritance, coupling, and cohesion.

E. Process Metrics

Process metrics capture software evolution and development activities, including:

·Code Churn
·Number of Revisions
·Commit Frequency
·Number of Developers
·Change Set Size

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

Process metrics are particularly useful in modern repository-based defect prediction models.

4.3 Benchmark Datasets

The evaluation of software defect prediction models relies heavily on publicly available benchmark datasets. The most frequently used datasets include:

NASA MDP Datasets

·CM1
·KC1
·KC2
·PC1
·PC2
·JM1
·MW1

PROMISE Repository Datasets

The PROMISE repository provides a collection of software engineering datasets containing software metrics and defect labels for empirical evaluation of prediction models.

These datasets facilitate objective comparison of different machine learning and evolutionary computing techniques.

4.4 Summary

Software metrics serve as the foundation of software defect prediction models. Size, complexity, object-oriented, and process metrics provide valuable information for identifying defect-prone modules. Public benchmark datasets such as NASA MDP and PROMISE repositories have played a significant role in advancing research on machine learning and evolutionary computing approaches for software defect prediction.

V. MACHINE LEARNING AND EVOLUTIONARY COMPUTING TECHNIQUES FOR SOFTWARE DEFECT PREDICTION

Machine learning and evolutionary computing techniques have significantly advanced software defect prediction by enabling the automatic identification of defect-prone software modules using historical software metrics and defect data. These techniques differ in terms of learning mechanisms, computational complexity, interpretability, and predictive performance. Based on their underlying principles, the approaches reported in the literature can be broadly categorized into traditional machine learning methods, ensemble learning techniques, deep learning models, and evolutionary computing approaches.

5.1 Traditional Machine Learning Techniques

Traditional machine learning algorithms constitute the foundation of software defect prediction research. These techniques learn patterns from software metrics and historical defect information to classify software modules as defective or non-defective.

5.1.1 Decision Tree

Decision Tree classifiers construct hierarchical decision structures by recursively partitioning the feature space according to attribute values. Due to their interpretability and ease of implementation, Decision Trees have been extensively employed in software defect prediction studies [20], [33]. They are particularly useful for identifying influential software metrics associated with defect occurrence. However, the predictive performance of individual decision trees may be affected by overfitting and data noise [1], [21].

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

5.1.2 Naïve Bayes

Naïve Bayes is a probabilistic classification technique based on Bayes' theorem and the assumption of conditional independence among features. Despite this simplifying assumption, Naïve Bayes has demonstrated competitive performance across numerous software defect datasets [11], [20]. Its computational efficiency and ability to handle high-dimensional data make it a popular baseline classifier in defect prediction research [1], [22].

5.1.3 K-Nearest Neighbors

K-Nearest Neighbors classifies software modules by examining the class labels of neighboring instances within the feature space. The algorithm requires minimal model construction and can effectively capture local data patterns [20], [33]. Nevertheless, its performance is sensitive to feature scaling, distance measures, and the selection of the neighborhood parameter [22].

5.1.4 Logistic Regression

Logistic Regression is one of the earliest statistical learning approaches adopted for software defect prediction. The method estimates the probability of defect occurrence through a logistic function and provides interpretable classification results [20], [33]. While Logistic Regression performs well on linearly separable datasets, its ability to model complex non-linear relationships is limited [1], [22].

5.1.5 Support Vector Machine

Support Vector Machine (SVM) is a supervised learning technique that identifies an optimal hyperplane for separating defective and non-defective software modules. By utilizing kernel functions, SVM can effectively model non-linear relationships within software metrics data. Numerous empirical studies have reported strong predictive performance for SVM, particularly on benchmark defect prediction datasets [39], [20], [33].

5.2 Ensemble Learning Techniques

Ensemble learning combines multiple base classifiers to improve predictive performance and reduce model variance. These approaches have gained considerable attention due to their robustness and ability to address limitations associated with individual classifiers.

5.2.1 Random Forest

Random Forest is an ensemble classifier that constructs multiple decision trees using bootstrap sampling and random feature selection. The final prediction is obtained through majority voting among the individual trees. Random Forest has consistently demonstrated superior performance in software defect prediction due to its robustness against overfitting and ability to handle high-dimensional datasets [20], [35], [31].

5.2.2 Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) represents one of the most successful boosting algorithms in contemporary machine learning research. The algorithm sequentially constructs decision trees while minimizing prediction errors through gradient optimization. Recent studies have reported competitive performance of XGBoost across various software defect prediction datasets, making it one of the most promising ensemble techniques in the field [31], [35].

5.3 Deep Learning Techniques

The increasing availability of software repositories and computational resources has facilitated the adoption of deep learning techniques for software defect prediction. Unlike traditional machine learning methods, deep learning models automatically learn hierarchical feature representations from raw input data.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 10, June 2026 Impact Factor: 8.2

Artificial Neural Networks (ANNs), Deep Neural Networks (DNNs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTM) networks have been investigated for defect prediction tasks [27], [28], [29]. These approaches have demonstrated the ability to capture complex non-linear relationships among software metrics [29]. However, their practical adoption is often constrained by computational requirements, data availability, and limited interpretability [22], [29].

5.4 Evolutionary Computing Techniques

Evolutionary computing approaches have emerged as effective optimization mechanisms for software defect prediction. These methods mimic natural evolutionary processes to explore large search spaces and identify optimal solutions.

5.4.1 Genetic Algorithms

Genetic Algorithms (GAs) are among the most widely adopted evolutionary techniques in software defect prediction research. Through iterative processes involving selection, crossover, and mutation, GAs optimize feature subsets, classifier parameters, and decision rules. The application of Genetic Algorithms has been shown to improve prediction accuracy by eliminating redundant features and enhancing model generalization capabilities.

The application of Genetic Algorithms in software defect prediction has received considerable attention due to their ability to optimize feature subsets and improve classification performance. Recent studies have demonstrated that evolutionary optimization can effectively identify relevant software metrics and enhance prediction effectiveness. Rajeev and Aravinthan proposed software metrics-based defect prediction frameworks utilizing Genetic Algorithms for feature optimization and classification, demonstrating the applicability of evolutionary approaches for improving defect prediction models. These studies highlight the potential of Genetic Algorithms in addressing feature redundancy and supporting the development of more robust software defect prediction systems [41], [42]. Genetic Algorithms have also been successfully combined with machine learning classifiers and feature selection techniques to improve prediction performance and model generalization [6], [13].

5.4.2 Hybrid Evolutionary Approaches

Recent studies have increasingly focused on hybrid frameworks that integrate evolutionary optimization with machine learning algorithms. These approaches leverage the optimization capabilities of evolutionary computing while retaining the predictive strengths of machine learning classifiers. Examples include Genetic Algorithm-based feature selection combined with Random Forest, Support Vector Machine, and rule-mining frameworks. Such hybrid approaches have demonstrated promising results in addressing issues related to feature redundancy, class imbalance, and model optimization [6], [13], [41].

5.5 Comparative Summary

Table 3 presents a comparative overview of the major machine learning and evolutionary computing techniques employed in software defect prediction.

Table 3. Comparative Overview of Machine Learning and Evolutionary Computing Approaches for Software Defect Prediction

Approach Category	Representative Techniques	Major Advantages	Limitations	Typical Applications
Traditional Machine Learning	Decision Tree, Naïve Bayes, KNN, Logistic Regression, SVM	Simple implementation, lower computational cost, good interpretability	Limited ability to model highly complex relationships	Baseline software defect prediction, small and medium-sized datasets
Ensemble Learning	Random Forest, XGBoost	High prediction accuracy, robustness against overfitting,	Increased computational complexity and reduced	Large-scale software defect prediction and

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 10, June 2026 Impact Factor: 8.2

		effective handling of high-dimensional data	interpretability	industrial applications
Deep Learning	ANN, DNN, CNN, RNN, LSTM	Automatic feature extraction, capability to learn complex non-linear patterns	Requires large datasets, high computational resources, limited explainability	Large software repositories and repository mining applications
Evolutionary Computing	Genetic Algorithms	Effective feature selection, optimization of classifier parameters, global search capability	Computationally intensive and parameter-sensitive	Feature optimization and rule generation
Hybrid Evolutionary Approaches	GA-RF, GA-SVM, Evolutionary Rule Mining Frameworks	Improved prediction accuracy, enhanced feature optimization, better generalization capability	Increased implementation complexity and training cost	Advanced software defect prediction frameworks

The literature indicates a gradual transition from traditional statistical and machine learning techniques toward ensemble, deep learning, and hybrid evolutionary approaches. While Random Forest and XGBoost consistently demonstrate strong predictive performance, evolutionary and hybrid approaches offer additional benefits through feature optimization and adaptive learning capabilities. Consequently, contemporary research increasingly focuses on integrating machine learning and evolutionary computing techniques to develop robust and accurate software defect prediction models.

VI. COMPARATIVE ANALYSIS OF SOFTWARE DEFECT PREDICTION TECHNIQUES

The effectiveness of software defect prediction models has been extensively investigated using various machine learning, ensemble learning, deep learning, and evolutionary computing approaches. Although numerous algorithms have been proposed, their performance often varies depending on the characteristics of the dataset, software metrics employed, feature selection techniques, and evaluation measures. Consequently, a comprehensive comparative analysis is essential for understanding the strengths and limitations of different prediction approaches.

6.1 Comparative Analysis Based on Prediction Performance

Prediction performance remains one of the primary criteria for evaluating software defect prediction models. Most studies employ metrics such as Accuracy, Precision, Recall, F1-Score, Matthews Correlation Coefficient (MCC), and Area Under the ROC Curve (AUC) to assess classifier effectiveness [7], [18], [20].

Traditional machine learning algorithms, including Decision Trees, Naïve Bayes, Logistic Regression, and K-Nearest Neighbors, generally provide satisfactory prediction performance with relatively low computational complexity. However, their performance may be limited when dealing with complex and highly imbalanced datasets.

Support Vector Machines have consistently demonstrated strong classification performance due to their ability to model complex decision boundaries. Nevertheless, their effectiveness is often influenced by kernel selection and parameter tuning.

Among ensemble approaches, Random Forest and XGBoost have emerged as some of the most successful techniques in software defect prediction [31], [35]. Their ability to combine multiple learners and reduce prediction variance often results in improved classification accuracy and robustness.

Deep learning approaches have shown promising results in recent studies by automatically extracting hierarchical feature representations [27], [28], [29]. However, their adoption remains limited due to computational requirements and the need for large-scale datasets.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

Evolutionary computing techniques, particularly Genetic Algorithms and hybrid evolutionary frameworks, have primarily contributed to performance improvement through feature optimization and rule generation [6], [41], [42]. These approaches are frequently integrated with traditional classifiers to enhance prediction accuracy and model generalization.

6.2 Comparative Analysis Based on Software Metrics

The predictive capability of defect prediction models is strongly influenced by the software metrics employed during model construction.

McCabe complexity metrics, including Cyclomatic Complexity, Essential Complexity, and Design Complexity, are among the most frequently utilized predictors due to their ability to capture software structural complexity [18], [41], [42].

Halstead metrics provide valuable information regarding software size, effort, volume, and maintainability. Numerous studies have reported positive correlations between high Halstead metric values and software defect occurrence [18], [41].

Object-oriented metrics such as Coupling Between Objects (CBO), Lack of Cohesion in Methods (LCOM), Weighted Methods per Class (WMC), and Response for a Class (RFC) have demonstrated significant predictive capability in object-oriented software systems [11], [23].

Process metrics, including code churn, commit frequency, and developer activity, have gained increasing attention in recent years because they capture software evolution characteristics that are often associated with defect introduction [16], [18].

The literature suggests that combining multiple categories of software metrics generally produces superior prediction performance compared with using a single metric category.

6.3 Comparative Analysis Based on Benchmark Datasets

Benchmark datasets play a crucial role in the evaluation and comparison of software defect prediction techniques.

The NASA MDP datasets remain among the most extensively utilized benchmark datasets due to their availability and comprehensive software metric information. Datasets such as JM1, KC1, CM1, and PC1 have been widely adopted for empirical evaluation [2], [7], [20].

Similarly, PROMISE repository datasets have become standard benchmarks for assessing software defect prediction models [11], [23].

Despite their widespread adoption, many benchmark datasets exhibit class imbalance, where defective modules represent only a small proportion of the overall dataset. This imbalance can significantly affect classifier performance and remains an active research challenge.

6.4 Comparative Analysis Based on Computational Characteristics

In addition to predictive performance, computational efficiency is an important consideration for practical deployment.

Traditional classifiers such as Naïve Bayes and Logistic Regression typically require minimal computational resources and training time.

Support Vector Machines and ensemble learning techniques generally require greater computational effort but often achieve superior prediction performance.

Deep learning approaches exhibit the highest computational requirements due to their complex architectures and extensive training processes.

Evolutionary algorithms often involve iterative optimization procedures, resulting in increased computational complexity. However, the performance gains achieved through feature optimization may justify the additional computational overhead in many applications.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

6.5 Comparative Summary of Prediction Techniques

Table 4 summarizes the major characteristics of software defect prediction approaches reported in the literature.

Table 4. Comparative Analysis of Software Defect Prediction Algorithms

Algorithm	Prediction Performance	Interpretability	Computational Cost	Scalability	Suitability for Imbalanced Data	Overall Assessment
Decision Tree	Moderate	High	Low	Medium	Moderate	Suitable for baseline prediction and rule extraction
Naïve Bayes	Moderate	High	Low	High	Moderate	Efficient and widely used baseline classifier
K-Nearest Neighbors	Moderate	Medium	Medium	Medium	Low	Effective for local pattern recognition but sensitive to dataset size
Logistic Regression	Moderate	High	Low	High	Moderate	Interpretable statistical prediction model
Support Vector Machine	High	Medium	High	Medium	High	Effective for complex classification problems
Random Forest	High	Medium	Medium	High	High	One of the most widely adopted SDP techniques
XGBoost	Very High	Medium	High	High	High	Consistently achieves strong predictive performance
Artificial Neural Network	High	Low	High	High	Moderate	Effective for modeling non-linear relationships
Deep Learning Models	Very High	Low	Very High	High	High	Suitable for large-scale software repositories
Genetic Algorithms	High	Medium	High	High	High	Effective for feature selection and optimization
Hybrid Evolutionary Models	Very High	Medium	High	High	Very High	Promising direction for next-generation SDP frameworks

6.6 Key Findings

The comparative analysis reveals several important observations:

Ensemble learning techniques generally outperform individual machine learning classifiers in terms of predictive accuracy and robustness.

Random Forest and XGBoost consistently appear among the top-performing techniques across multiple benchmark datasets.

Software metric quality significantly influences prediction performance regardless of the classification algorithm employed.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

Evolutionary computing approaches provide substantial benefits when integrated with machine learning classifiers through feature selection and optimization.

Class imbalance, dataset heterogeneity, and model interpretability remain major challenges in software defect prediction research.

Overall, the literature demonstrates a clear trend toward hybrid intelligent systems that combine machine learning, ensemble learning, and evolutionary computing techniques to achieve improved software defect prediction performance.

VII. RESEARCH CHALLENGES AND FUTURE DIRECTIONS

Despite significant advances in machine learning and evolutionary computing techniques for software defect prediction, several challenges continue to limit the reliability, generalizability, and practical applicability of existing prediction models. Addressing these challenges is essential for the development of more robust and effective software quality assurance systems.

7.1 Research Challenges

7.1.1 Class Imbalance Problem

One of the most widely reported challenges in software defect prediction is class imbalance. In many benchmark datasets, defective software modules constitute only a small proportion of the total instances. Consequently, prediction models may become biased toward the majority class, resulting in misleadingly high accuracy while failing to identify defective modules effectively.

Although various resampling and cost-sensitive learning techniques have been proposed, class imbalance remains a significant obstacle to achieving reliable prediction performance across diverse datasets [6], [30].

7.1.2 Dataset Heterogeneity

Software projects differ considerably in terms of programming languages, development practices, team structures, software architectures, and metric distributions. As a result, models trained on one dataset often fail to generalize effectively to other projects.

This issue is particularly evident in Cross-Project Defect Prediction (CPDP), where differences between source and target projects frequently reduce prediction accuracy [21], [23].

7.1.3 Feature Redundancy and Irrelevance

Software defect datasets often contain numerous software metrics, many of which may be redundant or irrelevant to defect prediction. The inclusion of such features can increase model complexity, reduce interpretability, and negatively affect prediction performance.

Feature selection and dimensionality reduction techniques have been widely investigated to address this challenge; however, identifying universally relevant metric subsets remains an open research problem [12], [41], [42].

7.1.4 Limited Interpretability

While advanced machine learning and deep learning models frequently achieve high predictive performance, they often operate as black-box systems. The lack of transparency can hinder trust and adoption among software practitioners who require explanations for prediction outcomes.

Improving model interpretability while maintaining predictive accuracy remains a critical challenge for future research [4], [25].

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

7.1.5 Computational Complexity

Many contemporary approaches, particularly deep learning and evolutionary optimization techniques, require substantial computational resources and training time. Such requirements may limit their practical deployment in resource-constrained software development environments.

Balancing prediction performance with computational efficiency continues to be an important consideration for software defect prediction systems [27], [29].

7.1.6 Data Quality and Availability

The effectiveness of software defect prediction models depends heavily on the quality of training data. Missing values, noisy records, inconsistent defect labels, and insufficient historical information can significantly affect model reliability [2], [14].

Moreover, many industrial datasets remain inaccessible due to confidentiality concerns, limiting opportunities for comprehensive empirical evaluation.

7.2 Emerging Research Trends

Recent years have witnessed the emergence of several promising research directions that aim to overcome the limitations of conventional software defect prediction approaches.

Explainable Artificial Intelligence

Explainable Artificial Intelligence (XAI) seeks to improve the transparency and interpretability of machine learning models by providing human-understandable explanations for prediction outcomes. The integration of XAI techniques into software defect prediction may facilitate greater practitioner acceptance and trust [4], [25].

Automated Machine Learning

Automated Machine Learning (AutoML) frameworks automate model selection, feature engineering, and hyperparameter optimization. AutoML has the potential to simplify defect prediction model development and improve prediction performance with reduced manual intervention [26].

Deep Learning-Based Prediction

Deep learning techniques continue to gain attention due to their ability to automatically learn complex feature representations from software repositories and source code. Future research is expected to explore more efficient and interpretable deep learning architectures for defect prediction [27], [28], [29].

Hybrid Intelligent Systems

The integration of machine learning, evolutionary computing, ensemble learning, and optimization techniques has emerged as a promising strategy for enhancing prediction accuracy and robustness. Hybrid intelligent systems are likely to play a significant role in future software defect prediction research [6], [41].

Repository Mining and Process Analytics

The increasing availability of software repositories, version control systems, and issue tracking platforms has created opportunities for utilizing process metrics and developer activity information in defect prediction models. Combining software metrics with repository analytics may improve predictive capability and practical applicability [16], [18].

7.3 Future Research Directions

Based on the findings of this review, several future research directions are identified:

1. Development of robust models capable of addressing severe class imbalance problems.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

2. Exploration of transfer learning and domain adaptation techniques for improving cross-project defect prediction.
3. Integration of explainable artificial intelligence methods to enhance model transparency.
4. Design of lightweight prediction models suitable for real-time software quality monitoring.
5. Development of hybrid machine learning and evolutionary computing frameworks for improved feature optimization and classification performance.
6. Investigation of deep learning architectures that balance predictive accuracy, interpretability, and computational efficiency.
7. Utilization of large-scale industrial datasets to improve the external validity of software defect prediction research.

7.4 Summary

Although substantial progress has been achieved in software defect prediction through machine learning and evolutionary computing techniques, several challenges remain unresolved. Issues related to class imbalance, dataset heterogeneity, feature redundancy, interpretability, and computational complexity continue to influence model effectiveness. Emerging paradigms such as explainable artificial intelligence, automated machine learning, deep learning, and hybrid intelligent systems provide promising opportunities for future advancements. Addressing these challenges will contribute to the development of more reliable, scalable, and practical software defect prediction frameworks capable of supporting modern software engineering practices.

VIII. CONCLUSION

Software defect prediction has emerged as a critical research area within software engineering due to its potential to improve software quality, reduce maintenance costs, and support efficient resource allocation during software development. The increasing availability of software repositories and historical defect data has facilitated the application of machine learning and evolutionary computing techniques for identifying defect-prone software modules. Consequently, numerous prediction models have been proposed and evaluated using various software metrics, benchmark datasets, and performance measures.

This paper presented a systematic review and comparative analysis of machine learning and evolutionary computing techniques employed in software defect prediction. The review examined traditional machine learning approaches, including Decision Trees, Naïve Bayes, K-Nearest Neighbors, Logistic Regression, and Support Vector Machines, as well as ensemble learning methods such as Random Forest and XGBoost. Furthermore, the study discussed the growing role of deep learning and evolutionary computing approaches, particularly Genetic Algorithms and hybrid optimization frameworks, in enhancing prediction performance and feature selection processes.

The review revealed that no single prediction technique consistently outperforms all others across different datasets and application scenarios. However, ensemble learning approaches, particularly Random Forest and XGBoost, have demonstrated strong predictive performance and robustness in numerous empirical studies. Deep learning techniques have shown considerable promise in modeling complex software characteristics, although their practical adoption remains constrained by computational requirements and interpretability concerns. Evolutionary computing techniques have proven valuable for feature optimization, parameter tuning, and the development of hybrid intelligent systems that combine the strengths of multiple prediction approaches.

The analysis also highlighted the importance of software metrics in the construction of effective defect prediction models. Complexity metrics, Halstead metrics, object-oriented metrics, and process metrics continue to play a significant role in identifying defect-prone software modules. Public benchmark datasets, particularly those from the NASA MDP and PROMISE repositories, have contributed substantially to the advancement of software defect prediction research by providing common evaluation platforms for comparative studies.

Despite the progress achieved in this field, several challenges remain unresolved. Class imbalance, dataset heterogeneity, feature redundancy, limited interpretability, and computational complexity continue to affect the effectiveness and practical applicability of software defect prediction models. Furthermore, the lack of standardized

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

evaluation frameworks and the limited availability of large-scale industrial datasets present additional obstacles to the development of universally applicable prediction systems.

Emerging research directions, including Explainable Artificial Intelligence (XAI), Automated Machine Learning (AutoML), deep learning, transfer learning, and hybrid evolutionary computing frameworks, offer promising opportunities for addressing existing limitations. Future studies should focus on developing interpretable, scalable, and computationally efficient prediction models capable of adapting to diverse software development environments and project characteristics.

Overall, this review provides a comprehensive synthesis of the current state of research in software defect prediction and offers valuable insights into the strengths, limitations, and future potential of machine learning and evolutionary computing techniques. The findings presented in this paper are expected to assist researchers, practitioners, and software quality engineers in selecting appropriate prediction approaches and identifying promising directions for future investigation in intelligent software quality assurance systems.

REFERENCES

[1] C. Catal and B. Diri, "A systematic review of software fault prediction studies," Expert Systems with Applications, vol. 36, no. 4, pp. 7346–7354, 2009.

[2] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data quality: Some comments on the NASA software defect datasets," IEEE Transactions on Software Engineering, vol. 39, no. 9, pp. 1208–1215, 2013.

[3] N. Li, M. Shepperd, and Y. Guo, "A systematic review of unsupervised learning techniques for software defect prediction," Information and Software Technology, vol. 122, 2020.

[4] G. Esteves, E. Figueiredo, A. Veloso, M. Viggiato, and N. Ziviani, "Understanding machine learning software defect predictions," Automated Software Engineering, vol. 27, no. 3–4, pp. 369–392, 2020.

[5] A. Hasanpour, P. Farzi, A. Tehrani, and R. Akbari, "Software defect prediction based on deep learning models: Performance study," 2020.

[6] M. Ali et al., "Enhancing software defect prediction: A framework with genetic algorithms, feature selection, and ensemble classification," Scientific Reports, vol. 14, 2024.

[7] T. Siddiqui and M. Mustaqeem, "Performance evaluation of software defect prediction with NASA dataset using machine learning techniques," International Journal of Information Technology, vol. 15, no. 8, pp. 1–15, 2023.

[8] M. Canaparo, "Software defect prediction: A study on software metrics using statistical and machine learning methods," Proceedings of ISGC, 2022.

[9] L. Perreault and J. Izurieta, "Using classifiers for software defect detection," Proceedings of SEDE, pp. 1–8, 2017.

[10] M. Shah and N. Pujara, "A review on software defects prediction methods," 2020.

[11] A. Okutan and O. T. Yildiz, "Software defect prediction using Bayesian networks," Empirical Software Engineering, vol. 19, no. 1, pp. 154–181, 2014.

[12] S. Agarwal and D. Tomar, "A feature selection based model for software defect prediction," International Journal of Advanced Science and Technology, vol. 65, pp. 39–58, 2014.

[13] K. Gao, T. M. Khoshgoftaar, and R. Wald, "Combining feature selection and ensemble learning for software quality estimation," Proceedings of FLAIRS Conference, 2014.

[14] H. Wang, T. M. Khoshgoftaar, and A. Napolitano, "Software measurement data reduction using ensemble techniques," Neurocomputing, vol. 92, pp. 124–132, 2012.

[15] J. Nam and S. Kim, "CLAMI: Defect prediction on unlabeled datasets," Proceedings of ASE, 2015.

[16] T. Fukushima, Y. Kamei, S. McIntosh, K. Yamashita, and N. Ubayashi, "An empirical study of just-in-time defect prediction," IEEE Transactions on Software Engineering, vol. 42, no. 6, pp. 1–15, 2016.

[17] A. Boucher and M. Badri, "Software metrics thresholds calculation techniques using ensemble learning on selected features," Information and Software Technology, vol. 58, pp. 388–402, 2015.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 10, June 2026 Impact Factor: 8.2

[18] X. Xuan, D. Lo, X. Xia, and Y. Tian, "Evaluating defect prediction using a massive set of metrics," Journal of Systems and Software, vol. 122, pp. 12–29, 2016.
[19] M. Panda, "DBBRBF: Hybrid distribution-based balance instance selection and radial basis function classifier for software defect prediction," 2018.
[20] A. Iqbal, S. Aftab, U. Ali, and Z. Nawaz, "Performance analysis of machine learning techniques on software defect prediction using NASA datasets," International Journal of Advanced Computer Science and Applications, vol. 10, no. 5, pp. 300–308, 2019.
[21] S. Stradowski and L. Madeyski, "Machine learning in software defect prediction: A business-driven systematic mapping study," Information and Software Technology, vol. 155, Art. no. 107128, 2023.
[22] 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, Art. no. 107886, 2022.
[23] J. Pachouly, S. Ahirrao, K. Kotecha, G. Selvachandran, and A. Abraham, "A systematic literature review on software defect prediction using artificial intelligence: Datasets, validation methods, approaches and tools," Engineering Applications of Artificial Intelligence, vol. 111, 2022.
[24] N. Li, M. Shepperd, and Y. Guo, "A systematic review of unsupervised learning techniques for software defect prediction," Information and Software Technology, vol. 122, Art. no. 106287, 2020.
[25] G. Esteves, E. Figueiredo, A. Veloso, M. Viggiato, and N. Ziviani, "Understanding machine learning software defect predictions," Automated Software Engineering, vol. 27, no. 3–4, pp. 369–392, 2020.
[26] K. Tanaka, A. Monden, and Z. Yucel, "Prediction of software defects using automated machine learning," Proc. IEEE/ACIS International Conference on Software Engineering, 2019, pp. 490–494.
[27] E. Borandag, "Software fault prediction using an RNN-based deep learning approach and ensemble machine learning techniques," Applied Sciences, vol. 13, no. 3, 2023.
[28] A. Faizin, A. Iqbal, E. Suharto, A. P. Widodo, and H. Faizal, "Software defect prediction using deep learning," AIP Conference Proceedings, vol. 2738, no. 1, 2023.
[29] E. N. Akimova et al., "A survey on software defect prediction using deep learning," Mathematics, vol. 9, no. 11, p. 1180, 2021.
[30] R. Bahaweres, F. Agustian, I. Hermadi, A. Suroso, and Y. Arkeman, "Software defect prediction using Neural Network based SMOTE," Proc. EECSI, 2020, pp. 71–76.
[31] L. Q. Chen, C. Wang, and S. L. Song, "Software defect prediction based on nested stacking and heterogeneous feature selection," Complex & Intelligent Systems, vol. 8, no. 4, pp. 3333–3348, 2022.
[32] A. Al-Nusirat, F. Hanandeh, M. K. Kharabsheh, M. Al-Ayyoub, and N. Al-Dhfairi, "Dynamic detection of software defects using supervised learning techniques," International Journal of Communication Networks and Information Security, vol. 11, no. 1, pp. 185–191, 2022.
[33] M. Azam, M. Nouman, and A. R. Gill, "Comparative analysis of machine learning techniques to improve software defect prediction," Journal of Computing and Information Sciences, vol. 5, no. 2, pp. 41–66, 2022.
[34] M. Pavana, L. Pushpa, and A. Parkavi, "Software fault prediction using machine learning algorithms," Proc. International Conference on Advanced Electrical and Computer Technology, 2022, pp. 185–197.
[35] A. Khalid, G. Badshah, N. Ayub, M. Shiraz, and M. Ghouse, "Software defect prediction analysis using machine learning techniques," Sustainability, vol. 15, no. 6, 2023.
[36] A. Sunil, R. K. Sahu, and S. Karsoliya, "Software Defect Prediction using Supervised Machine Learning: A Systematic Literature Review," International Journal of Advanced Research and Multidisciplinary Trends, vol. 2, no. 3, pp. 80–95, 2025.
[37] R. Aggarwal, "Machine Learning Approaches in Software Fault Prediction: A Review," Cureus Journals, vol. 1, no. 1, pp. 1–15, 2026.
[38] H. K. Dam, T. Pham, S. W. Ng, T. Tran, J. Grundy, A. Ghose, T. Kim, and C. J. Kim, "A deep tree-based model for software defect prediction," Information and Software Technology, vol. 2018.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

[39] J. A. Reshi and S. Singh, "Predicting software defects through SVM: An empirical approach," International Journal of Software Engineering and Applications, 2018.
[40] G. Destefanis, L. Yousefi, M. Shepperd, A. Tucker, S. Swift, S. Counsell, and M. Arzoky, "An audit of machine learning experiments on software defect prediction," 2026.
[41] Rajeev P R and K. Aravinthan, "A Novel Approach for Metrics-Based Software Defect Prediction Using Genetic Algorithm," The Scientific Temper, vol. 15, no. 3, pp. 2709–2718, 2024, doi: 10.58414/SCIENTIFICTEMPER.2024.15.3.39.
[42] Rajeev P R and K. Aravinthan, "A Generic Approach for Software Metrics Based Software Defect Prediction," Indian Journal of Natural Sciences, vol. 15, no. 86, 2024.