A Comparative Analysis of Techniques, Datasets, Feature Selection Methods, and Evaluation Metrics in Software Fault Prediction

Rajinder Kumar, Kamaljit Kaur

***Abstract:** This study presents a systematic literature review (SLR) that investigates recent advancements in Software Fault Prediction (SFP) methodologies. The review focuses on key dimensions including techniques, datasets, feature selection methods, software metrics, and evaluation criteria. By analyzing significant studies from renowned digital libraries such as ACM, IEEE, Springer Link, and Science Direct, five research questions were defined to guide the assessment of current trends in SFP research. Findings reveal that machine learning approaches—particularly neural networks, deep learning, and ensemble methods—are increasingly employed due to their capability to manage the complexity of software fault data. Public datasets, notably those from the PROMISE and NASA MDP repositories, are widely utilized, underlining the importance of dataset diversity for enhancing model performance. Feature selection methods, particularly wrapper techniques, are often employed to improve predictive accuracy. Evaluation of models predominantly relies on confusion matrix-based metrics such as Accuracy, Precision, Recall, and F1-Score. Despite these advances, challenges remain in addressing class imbalance, adapting to rapidly evolving software environments, and achieving real-time fault prediction. The study highlights the need for greater classifier diversity and ongoing methodological improvements to enhance the robustness and generalizability of SFP models.*

***Keywords:** Software Fault Prediction; Feature Selection Techniques; Software Metrics; Public Datasets; Confusion Matrix-Based; Class Imbalance.*

Abbreviations:
SDP: Software Defect Prediction
SBP: Software Bug Prediction
SFP: Software Fault Prediction
SQA: Software Quality Assurance
PS: Primary Studies
SLR: Systematic Literature Review

I. INTRODUCTION

In the field of software engineering, software defect prediction (SDP) in early stages is vital for software reliability and quality. A software defect is a bug, fault, or error in a program that causes improper outcomes. Software defects are programming errors that can occur due to errors in the source code, requirements, or design. Defects negatively affect software quality and reliability. Various terms, including hybrid, combined, integrated, and aggregated classification, are employed in ensemble learning. The ensemble learning model is built by combining multiple machine learning classifiers to improve prediction performance [1]. The main objective of a software project is to deliver the expected functionality while meeting the required level of quality on time and within a defined budget. From the perspective of software projects developed in recent years, the complexity in software development has increased due to the increased number of customer requirements [2]. The primary objective of software bug prediction (SBP) techniques is to classify fault-prone and fault-free modules, allowing developers to assign reasonable testing sources and allocate testing preferences for various software modules, thereby enhancing the software's quality [3]. Software fault prediction (SFP) is the area of interest for many researchers and software developers. Predicting such faults at an early stage of development can reduce the maintenance cost and effort. Fault prediction models aid in various software-related activities, such as quality assurance, to enhance the understanding of software quality. This prediction is done using different software metrics. The commonly used software metrics are McCabe metrics, Halstead metrics and CK metrics. Fault prediction performed during early development will reduce maintenance costs and improve software quality. Current software systems are becoming increasingly complex and large; therefore, ensuring their reliability and quality is paramount, which depends on identifying and mitigating software faults. Software fault prediction (SFP) actively assists in detecting faults by highlighting potential faulty areas of code within the software system [4]. Reducing defects and failures in a software product is a crucial goal for software engineers. This is done to achieve maximum performance, build user trust, and enhance the overall quality of the product. During the life cycle of a product, a software goes through several feature changes, quality iterations and reassembling. Software quality assurance (SQA) consists of monitoring and controlling the software development process to ensure the desired software quality at a lower cost. It may include the application of formal code inspections, code walkthroughs, software testing, and software fault prediction Software fault prediction aims to facilitate the allocation of limited SQA resources optimally and economically by prior prediction of the fault-proneness of software modules The potential of software fault prediction to identify faulty software modules early in the

Manuscript received on 12 June 2025 | First Revised Manuscript received on 05 July 2025 | Second Revised Manuscript received on 10 July 2025 | Manuscript Accepted on 15 July 2025 | Manuscript published on 30 July 2025.

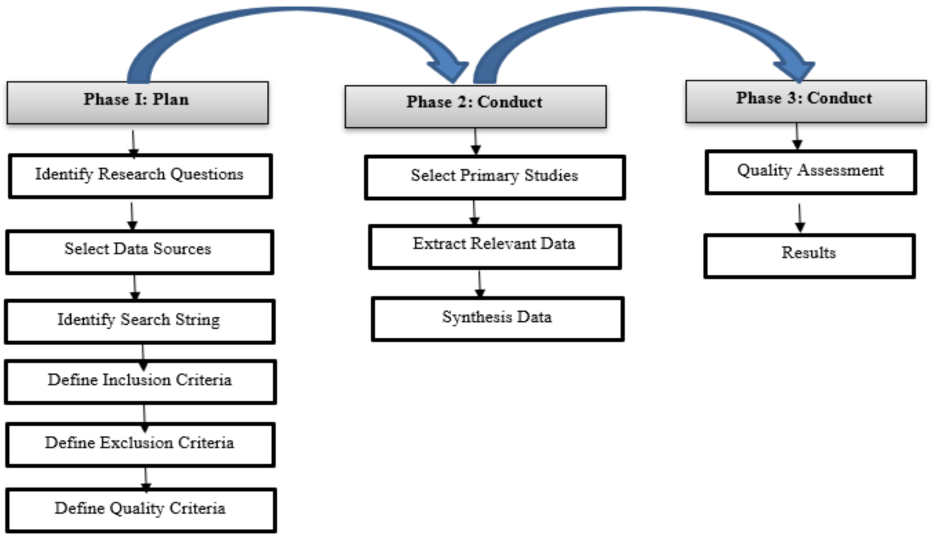
*Correspondence Author(s)
Rajinder Kumar*, Research Scholar, Department of Computer Science and Engineering, Sri Guru Granth Sahib World University, Fatehgarh Sahib. (Punjab), India and Assistant Professor, Department of Computer Applications, Chandigarh Business School of Administration, Landran, Mohali (Punjab), India. Email: Rajinderkumar.cse@gmail.com, ORCID ID: 0009-0007-3095-3872
Dr. Kamaljit Kaur, Assistant Professor, Department of Computer Science, Sri Guru Granth Sahib World University, Fatehgarh Sahib (Punjab), India. Email ID: drkamaljit2024@gmail.com

development life cycle has gained considerable attention over the last two decades. From a software development perspective, dealing with software faults is a vital and foremost important task. The presence of faults not only deteriorates the quality of the software but also increases its development and maintenance costs. Therefore, identifying which software module is likely to be fault-prone during the early phases of software development may help improve the quality of the software system. By predicting the number of faults in software modules, we can guide software testers to focus on faulty modules first. The objective of software fault prediction is to detect faulty software modules before the testing phase by using some structural characteristics of the software system. A software fault prediction model is generally constructed using fault datasets from previous releases of similar software projects, and it is later applied to predict faults in the currently under-development software system. In conclusion, this review aims to provide a comprehensive overview of SFP techniques, tracing their evolution from simple code complexity metrics to sophisticated AI-driven models. This paper provides a review of the recent research conducted on the use of software defect prediction. The latest papers published since 2025 are considered for this study. Four renowned and widely used online search libraries are selected for extracting relevant literature, including IEEE Xplore, ACM Digital Library, ScienceDirect, and SpringerLink. Initially, 1180 papers are extracted, and then the 23 most relevant papers are selected as Primary Studies (PS) after following a thorough systematic research process. The remainder of the paper is organized as follows: Section 2 presents the research protocol. Section 3 presents the findings of this review. Finally, section 4 concludes the paper with suggestions for future work for many researchers and software developers. Predicting such faults at an early stage of development can reduce the maintenance cost and effort. Fault prediction models aid in various software-related activities, such as quality assurance, to enhance the understanding of software quality. This prediction is done using different software metrics. The commonly used software metrics are McCabe metrics, Halstead metrics and CK metrics. Fault prediction performed during early development will reduce maintenance costs and improve software quality. Current software systems are becoming increasingly complex and large; therefore, ensuring their reliability and quality is of paramount importance, which depends on identifying and mitigating software faults. Software fault prediction (SFP) actively assists in detecting faults by highlighting potential faulty areas of code within the software system. Reducing defects and failures in a software product is a crucial goal for software engineers. This is done to achieve maximum performance, build user trust, and enhance the overall quality of the product. During the life cycle of a product, a software goes through several feature changes, quality iterations and reassembling. Ideally, all these changes are perfectly merged, should cause no defect and are free of error [6]. Software quality assurance (SQA) consists of monitoring and controlling the software development process to ensure the desired software quality at a lower cost. It may include the application of formal code inspections, code walkthroughs, software testing, and software fault prediction Software fault prediction aims to facilitate the allocation of limited SQA resources optimally and economically by prior prediction of the fault-proneness of software modules The potential of software fault prediction to identify faulty software modules early in the development life cycle has gained considerable attention over the last two decades [5]. From a software development perspective, dealing with software faults is a vital and foremost important task. The presence of faults not only deteriorates the quality of the software but also increases its development and maintenance costs. Therefore, identifying which software module is likely to be fault-prone during the early phases of software development may help improve the quality of the software system. By predicting the number of faults in software modules, we can guide software testers to focus on faulty modules first. The objective of software fault prediction is to detect faulty software modules before the testing phase by using some structural characteristics of the software system.

A software fault prediction model is generally constructed using fault datasets from previous releases of similar software projects, and it is later applied to predict faults in the currently under-development software system. In conclusion, this review aims to provide a comprehensive overview of SFP techniques, tracing their evolution from simple code complexity metrics to sophisticated AI-driven models. This paper provides a review of the recent research conducted on the use of software defect prediction. The latest papers published since 2025 are considered for this study. Four renowned and widely used online search libraries are selected for extracting relevant literature, including IEEE Xplore, ACM Digital Library, ScienceDirect, and SpringerLink. Initially, 1180 papers are extracted, and then the 23 most relevant papers are selected as Primary Studies (PS) after following a thorough systematic research process. The remainder of the paper is organized as follows: Section 2 presents the research protocol. Section 3 presents the findings of this review. Finally, section 4 concludes the paper with suggestions for future work.

II. REVIEW GUIDELINES

Our review methodology, grounded in A systematic literature review (SLR) framework as delineated by Kitchenham [7], is meticulously adapted to explore the realm of software fault prediction techniques. This structured approach begins by formulating research questions to define the scope, objectives, and depth of our review, with a particular focus on the evolution, effectiveness, and comparative analysis of various fault prediction methods. Inclusion and exclusion criteria are then rigorously established to delineate the boundaries of our systematic literature review. The SLR process is divided into three phases: planning the review, conducting the review, and reporting the review. Each phase consists of sub-phases, as illustrated in Figure 1. The systematic literature review (SLR) process is described in detail for software fault prediction research.



[Fig.1: SLR Process]

A. Phase 1:

i. Planning the Review

This section outlines the methodology employed for conducting our comprehensive bibliographic review, which is grounded in the systematic review guidelines outlined by Kitchenham [7]. We articulate our literature search strategy, including the databases and keywords used, detail our selection criteria for sourcing relevant studies, and elucidate the specific research questions guiding our inquiry. This structured approach ensures a thorough and unbiased survey of the existing literature in the field of software fault prediction.

ii. Research Questions

In the pursuit of a comprehensive understanding of software fault prediction (SFP), this review is guided by a series of targeted research questions. These questions are designed to explore various aspects of SFP, including techniques, datasets, programming languages, and evaluation criteria. The primary objective of this review is to address the following issues systematically.

- **Research questions:**

RQ1: What are the most used Methodologies/Techniques for software fault prediction?
RQ2: What types & typical sizes of datasets are predominantly used in software fault prediction studies?
RQ3: Which feature selection techniques are commonly applied in SFP?
RQ4: Which types of software metrics are utilized in SFP research?
RQ5: What evaluation criteria are commonly used to measure the performance of SFP models?

By addressing these questions, this review aims to provide a detailed and comprehensive analysis of the current state of software fault prediction techniques, offering valuable insights and identifying potential areas for future research.

iii. Literature Search Strategy

To systematically identify articles relevant to software fault prediction (SFP), we first considered key data sources in the field of software engineering and computer science. These included IEEE Xplore, ACM Digital Library, ScienceDirect, and SpringerLink. IEEE Xplore was selected as the primary source due to its extensive coverage of software engineering topics and publication types. To formulate the search string, particular keywords and their synonyms are selected from the identified research questions, as shown in **Table 1**.

Table-I. Search String

Keyboards	Alternatives/Synonyms
Software	(Program OR System)
Defect	(Software fault OR software error OR bug prediction OR fault detection OR error detection OR fault prediction)
Prediction	(Estimation OR Classification)
Ensemble	(Integrated OR hybrid)
Learner	(Machine learning" OR "artificial intelligence" OR "algorithm" OR "classifier" OR "technique" OR "method" OR "feature selection" OR "model)

The keywords are then arranged with the conditions of `AND' and `OR' in a particular sequence to form the following query:

(("software" OR "program" OR "system") *AND* ("software fault" [Title] OR "software error" [Title] OR "bug prediction" [Title] OR "fault detection" [Title] OR "error detection" [Title] OR "fault prediction" [Title]) AND (prediction" OR ``estimation" OR ``classification'') AND (ensemble OR integrated OR hybrid) AND ("learning "OR "machine learning" OR "artificial intelligence" OR "algorithm" OR "classifier" OR "technique" OR "method" OR "feature selection" OR "model")).

iv. Literature Search Criteria

To refine the focus of this systematic review on software fault prediction, we established a detailed set of inclusion and exclusion criteria. These criteria were crucial in selecting the most pertinent articles from the pool that matched our search query. The requirements are detailed in Tables 1 and 2. Following this rigorous screening process, a total of 57 articles met the criteria and were retained for in-depth analysis in our review.

v. Criteria for Data Extraction from Literature

To systematically gather information pertinent to our research questions, data extraction forms were meticulously crafted. These forms were instrumental in capturing key information from the selected studies and consisted of the components outlined in **Table 4.**

Table-II: Inclusion Criteria

No.	Criteria	Description
1	Focus on SFP Techniques	Papers specifically focusing on software fault prediction techniques.
2	Use of ML/AI	Studies that utilize machine learning or artificial intelligence in fault prediction.
3	Empirical Data and Case Studies	Research including empirical data, experiments, or case studies.
4	Published in Peer-Reviewed Sources	Articles published in peer-reviewed journals or conferences.
5	Language	Papers published in the English language.

Table-III: Exclusion Criteria

No.	Criteria	Description
1	Non-Specific to SFP	Papers not specifically addressing software fault prediction
2	Lack of Empirical Data	Studies that lack empirical data or case studies
3	Non-Peer-Reviewed Sources	Articles published in non-peer-reviewed sources
4	Publication Date	Papers published before the year 2020
5	Language	Non-English language publications.

Table-IV: Quality Assessment Criteria Used for Selection of Papers

Sr. No	QA Checklist
QA1	Does the selected study provide enough detail regarding the use of research objectives for SFP Methodologies/Techniques clearly defined, to answer **RQ1?**
QA2	Does the selected study Provide enough detail regarding the types of datasets used in the studies, correctly stated, and justified, to answer **RQ2?**
QA3	Does the selected study Provide enough detail regarding the types of feature selection techniques described and their effectiveness evaluated, to answer the **RQ3?**
QA4	Does the selected study Provide enough detail regarding the types of software metrics utilized in SFP research to answer the **RQ4?**
QA5	Does the selected study Provide enough detail regarding the evaluation criteria are commonly used to measure the performance of SFP models, to answer the **RQ5?**

B. Phase 2:

i. Conducting the Review

- **Selection of Primary Studies**

The selection of primary studies is a critical step in any systematic literature review, as it lays the foundation for understanding the current state of research, identifying gaps, and determining the direction of future research. Primary studies are chosen based on their relevance, rigour, and contribution to the research questions posed.

In this paper, Primary Studies are known as the most appropriate articles selected by following the tollgate approach to answer the identified questions. The tollgate approach, comprising five phases (P-1 to P-5), facilitates the selection of 23 Primary Studies, as shown in **Table 6.** The mentioned quality criteria (Table 3) are followed during the selection of each primary study. The filters of the tollgate phases are given as follows:

Table-V. Tollgate Approach

Selected Sources	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5
IEEE Xplore	300	70	50	24	5
ACM	250	50	40	20	6
Science direct	350	60	35	15	5
Springer Link	280	90	55	25	7
Total	**1180**	**270**	**180**	**84**	**23**

Phase 1 (P-1): Initially extracted data by using various combinations of keywords from a search query.

Phase 2 (P-2): Removed duplicates and applied inclusion/exclusion criteria by reading the title.

Phase 3 (P-3): Applied inclusion/exclusion criteria by reading the abstract.

Phase 4 (P-4): Applied inclusion/exclusion criteria by reading the introduction and conclusion.

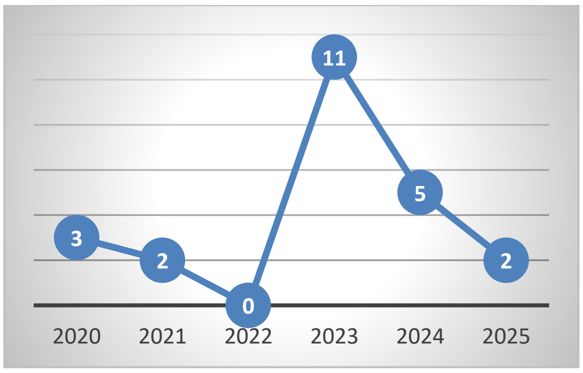
Phase 5 (P-5): Applied inclusion/exclusion criteria by reading the full text of selected studies. These articles are considered primary studies

ii. Data Extraction

The extracted data from each primary study include the following details: proposed/SFP technique, criteria for performance evaluation, the tool used for SFP implementation, datasets utilised for the experiments, and the techniques with which proposed/used SFP methods are compared. studies were chosen because they collectively address the comprehensive set of research questions that span a wide range of topics, from methodologies and datasets to tools and evaluation criteria. Each study contributes unique insights into the challenges and advancements within the SFP, offering a rich, diverse, and up-to-date perspective on the field.

iii. Data Synthesis

This stage involves the fusion of relevant extracted data, determining the amount of data
needed to address each question, and compile and present the data as shown in **Fig. 2.**



[Fig.2. Distribution of Primary Studies Over the Years]

Table-VI: Selection of Primary Studied

Reference	RQ1	RQ2	RQ3	RQ4	RQ5
Daza,2025 [6]	✓	×	✓	✓	✓
Goyal and Bhatia, 2021 [8]	×	✓	×	✓	×
Rathore and Kumar, 2021 [9]	✓	✓	×	×	✓
Qiao et al., 2020 [10]	×	✓	×	×	×
Boloori and Farhadi ,2024[11]	✓	✓	✓	×	✓
Elmishali and Kalech, 2023 [12]	×	✓	✓	✓	×
Kaur et al., 2023 [13]	✓	×	✓	×	✓
Rajput et al., 2023 [14]	✓	✓	×	×	×
Arora et al., 2023 [15] [28]	✓	×	✓	×	✓
Mehmood et al., 2023 [16]	✓	✓	×	✓	×
Wang et al., 2023 [17]	✓	✓	✓	✓	×
Khan and Nadeem, 2023 [18]	×	✓	×	×	×
Khalid and Ayub,2023[22]	×	✓	✓	✓	✓
Das and Alameen, 2023 [24]	×	✓	✓	✓	×
Yucalar et al., 2020 [25]	×	✓	×	✓	×
Pandey and Gupta,2024 [27]	×	✓	×	✓	×
Ali and Saeed,2020 [29]	×	✓	✓	✓	✓
Mafarja, Thaher and Too,2023[35]	✓	✓	✓	×	✓
Alsangari and Biricik, 2023, [55]	✓	✓	×	✓	✓
Haque, Ali and Noppen, 2024[50]	×	×	✓	✓	✓
Albattah and Alzahrani, 2024[52]	×	✓	×	✓	✓
Kaliraj and Sivakumar, 2024[55]	✓	✓	✓	×	×
Kumar and Das, 2025 [57]	✓	✓	✓	×	×

C. Phase 3:

i. Reporting the Review

- **Quality Assessment**

Each selected primary study is assessed against QA criteria (Table 4) and assigned a score between 0 and 1, as shown in Table 7. Many researchers adopt this process of quality assessment in SLRs [23]. If the article explicitly answers the QA question, the study is given a score of 1; if it partially does, the score is 0.5.

If the study answers the question, it is given a score of 0.5. A score of 0 is assigned to studies that fail to answer QA questions. The final score is calculated by summing the scores for all QA questions. After assessing the quality of the selected primary studies, it was found that the score of each primary study was greater than four against the QA criteria. This finding indicates that the selected primary studies provide sufficient information about ensemble learners.

Table-VII: Quality Assessment for Primary Studies Selection

Criteria	RQ1:	RQ2:	RQ3	RQ4	RQ5:	Total Score (out of 5)	Include (Yes/No)
QA1	✓ (1)	✓ (1)	✓ (1)	✓ (1)	✓ (1)	5	Yes
QA2	✓ (1)	− (0.5)	✓ (1)	− (0.5)	✗ (0)	3	Yes
QA3	✗ (0)	✗ (0)	− (0.5)	✗ (0)	✗ (0)	0.5	No
QA4	✓ (1)	✓ (1)	✓ (1)	✓ (1)	− (0.5)	4.5	Yes
QA5	✓ (1)	✓ (1)	✓ (1)	− (0.5)	✓ (1)	4.5	Yes
	✓ = Yes (1 point)		− = Partial (0.5 points)			✗ = No (0 points)	

ii. Results

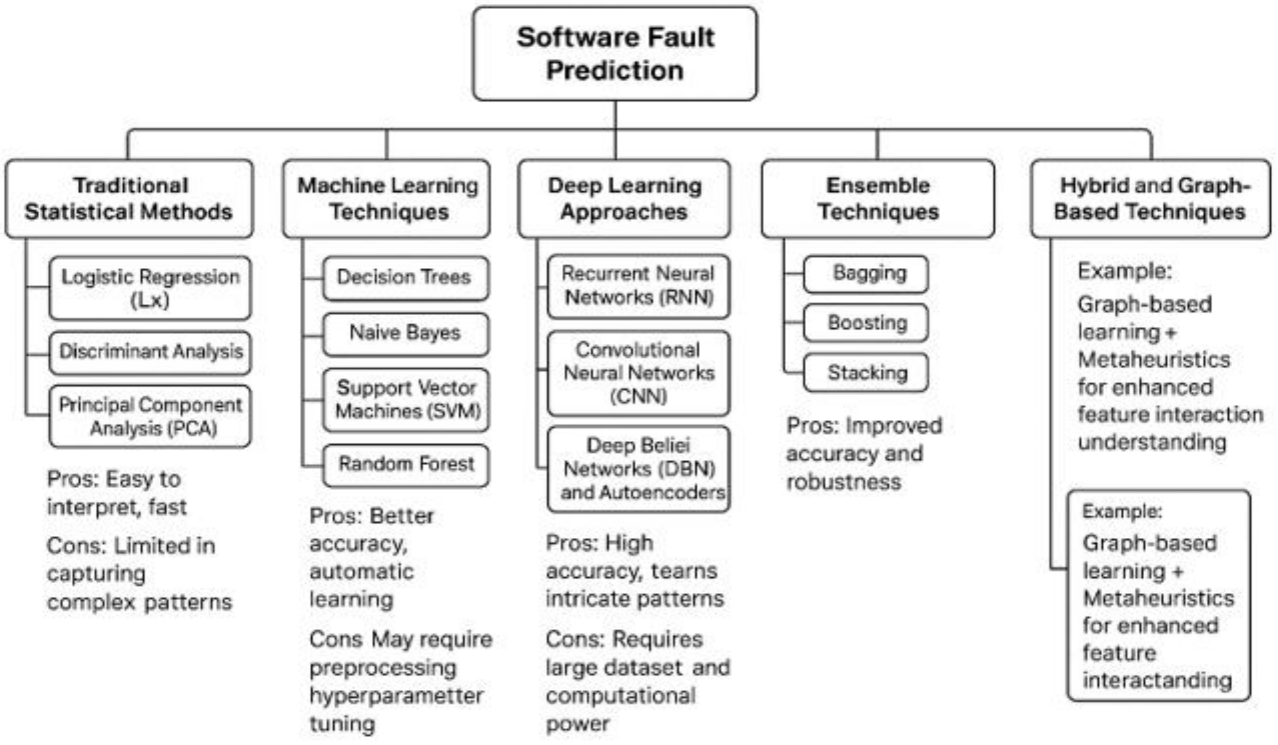
The final stage of the systematic research process involves evaluating the answers to identified research questions following a critical review. The detailed extracted answers from each primary study are discussed in the given section.

iii. Software Fault Prediction Methodologies/ Techniques and Associated Challenges.

Software quality assurance (SQA), which includes formal code inspections, code walkthroughs, software testing, validation, verification, and software fault prediction, ensures the desired software quality at a lower cost by monitoring and controlling the Software Development Life Cycle (SDLC) [10]. However, complete testing of a software system is practically not possible as it consumes an enormous amount of time and resources. SFP techniques can be broadly categorized into traditional statistical, machine learning, deep learning, and metaheuristic-based hybrid approaches: The taxonomy of soft computing techniques is given in Fig. 3.

iv. RQ1: What are the Most Commonly Used Methodologies/ Techniques for Software Fault Prediction?

This section presents a synthesis of the findings pertinent to Research Question 1 (RQ1) - "Which kind of Techniques are most used for software fault prediction?". For RQ1, we observed a diverse range of techniques employed across the selected studies. The techniques were often tied to specific periods where methodologies were at the forefront of research due to advancements in machine learning and computational capabilities. In this section, a comparative analysis of the work of numerous researchers in the field of Software fault prediction methodologies and techniques is discussed. The comparative analysis encompasses the work of various researchers in the field of methods and techniques used in software fault prediction from 2020 to 2025. The work of researchers is compared in terms of techniques, Objective, Result, and challenge of their research work and is shown in **Table 8** The figure 3. illustrates a taxonomy of Software Fault Prediction (SFP) techniques, categorizing them into five major groups: Traditional Statistical Methods, Machine Learning Techniques, Deep Learning Approaches, Ensemble Techniques and Hybrid and Graph-Based Techniques. The figure provides a comprehensive overview of how software fault prediction techniques have evolved—from simple statistical models to complex hybrid models involving deep learning and optimization algorithms [40]. Each category has distinct advantages and trade-offs, depending on the data complexity and prediction goals. This figure captures the evolution and diversification of techniques in software fault prediction: Early approaches relied on simple, interpretable models (e.g., Logistic Regression). Then, machine learning brought automation and better generalization. Deep learning pushed the frontier with highly complex, data-hungry models. Ensemble methods focused on combining models for robustness [50]. Most recently, hybrid approaches like graph-based learning with metaheuristics aim to maximize feature interaction understanding and optimize prediction accuracy. Each category has its trade-offs in terms of interpretability, accuracy, data requirements, and computational cost. The best choice often depends on the dataset characteristics and project constraints.


Software Fault Prediction
Traditional Statistical Methods
Logistic Regression (Lx)
Discriminant Analysis
Principal Component Analysis (PCA)
Pros: Easy to interpret, fast
Cons: Limited in capturing complex patterns
Machine Learning Techniques
Decision Trees
Naive Bayes
Support Vector Machines (SVM)
Random Forest
Pros: Better accuracy, automatic learning
Cons May require preprocessing hyperparametter tuning
Deep Learning Approaches
Recurrent Neural Networks (RNN)
Convolutional Neural Networks (CNN)
Deep Beliei Networks (DBN) and Autoencoders
Pros: High accuracy, tearns intricate patterns
Cons: Requires large dataset and computational power
Ensemble Techniques
Bagging
Boosting
Stacking
Pros: Improved accuracy and robustness
Hybrid and Graph-Based Techniques
Example: Graph-based learning + Metaheuristics for enhanced feature interaction understanding
Example: Graph-based learning + Metaheuristics for enhanced feature interactanding

[Fig.3: The Taxonomy of Software Fault Prediction Methodologies /Techniques]

v. Software Fault Prediction Feature Selection Methodology /Techniques and Associated Challenges

Software Fault Prediction (SFP), feature selection plays a vital role in improving model performance, interpretability, and generalization by identifying the most relevant software metrics. Feature selection is the process of identifying and selecting a subset of pertinent and significant features (attributes/metrics) from a larger set of data. In Software Engineering, this is particularly important in areas such as software fault prediction, Effort Estimation, Code Smell Detection, Software Maintenance and Evolution, and Defect Localisation. Feature selection in software engineering helps focus on the most important factors influencing quality, productivity, or maintenance. It enhances model performance, supports better decisions, and can guide improvements in software design, testing, and development. Choosing the proper feature selection method depends on the Dataset size and nature, as well as the Prediction goals (e.g., fault vs. effort). Desired balance between accuracy, interpretability, and scalability. [57] Below is a summary of the most applied feature selection techniques categorized by methodology

Table-VIII: Representation of the Feature Selection Techniques in SFP

Category	Technique Name
Filter Methods	- Information Gain (IG) - Chi-Square - ReliefF
Wrapper Methods	- Forward Selection - Backwards Elimination - Recursive Feature Elimination (RFE)
Embedded Methods	- LASSO (L1 Regularization) - Ridge (L2) - Tree-Based (e.g., Random Forest)
Metaheuristic / Hybrid Methods	- Genetic Algorithm (GA) - Particle Swarm Optimization (PSO) - Whale Optimization Algorithm (WOA) - Grey Wolf Optimizer (GWO) - Graph-based Hybrid Approaches

Table 9 Representation of the Feature Selection Techniques in SFP. The Filter methods are fast, but may ignore feature interactions. Wrapper methods give better performance, but are slow. Embedded methods balance speed and accuracy. Metaheuristics are particularly effective for large and complex datasets and have gained popularity in recent SFP research.

Table-IX: Comprehensive Overview of Software Fault Prediction Methodologies/Techniques

Ref. No	Aim of the Study	Methodologies/Techniques	Objectives	Result	Challenges
[2]	Software Fault Prediction Using an RNN-Based Deep Learning Approach and Ensemble Machine Learning Techniques	RNN-based deep learning approach (RNNBDL)	Adoption of advanced machine learning and deep learning techniques in the software development lifecycle	The RNNBDL model demonstrated the highest accuracy (ACC) on large datasets, such as Apache ActiveMQ and SFP XP-TDD.	Indicates a direction towards creating a novel hybrid technology that combines various methodologies for software fault prediction
[3]	BPDET: An Effective Software Bug Prediction Model using Deep Representation and Ensemble Learning Techniques	Bug Prediction Model using Deep Representation and Ensemble Learning Techniques	proposes a classification framework called Bug Prediction using Deep Representation and Ensemble Learning (BPDET) for SBP. The research focuses on improving the performance of SBP models using deep learning techniques	The proposed BPDET model outperformed baseline methods in eight out of twelve datasets	The class imbalance problem, which affects the accuracy of predicting faulty and non-faulty modules
[5]	A new Ensemble approach for Software Fault Prediction	Model averaging ensemble method (combinations of classifiers and resampling techniques)	The study aims to investigate the impact of different classifiers and resampling techniques on software fault prediction (SFP) performance.	Significant results were found for all classifiers except Naive Bayes in the Friedman test.	The research also highlights the scarcity of studies that report the impact of resampling techniques on classifier performance.
[4]	An empirical study of ensemble techniques for software fault Prediction	Ensemble techniques such as Dagging, Decorate, Grading, Multi BoostAB, Real AdaBoost, Rotation Forest, Ensemble Selection and classification algorithms are naive Bayes, logistic regression, and J48 (decision tree)	It aims to evaluate the cost-effectiveness of SFP models based on the ensemble techniques used.	Cost-benefit analysis indicates that SFP models can save testing costs for 20 out of 28 datasets.	It notes that many new ensemble techniques have not been explored for software fault prediction.
[7]	A Deep Ensemble Learning Method for Effort-Aware Just-In-Time Defect Prediction	Fusion-based method that combines deep learning techniques with Random Forest and XGBoost classifiers.	The research also introduces a reinforcement learning technique to minimize false alarms in real-time predictions.	The study highlighted the importance of handling unbalanced data for effective model performance.	The paper discusses the challenges associated with the unbalanced properties of datasets, which can impact the performance of defect prediction models.
[9]	Software fault prediction based on the dynamic selection of learning technique: findings from the Eclipse project study	It evaluates learning techniques: Naive Bayes (NB), Logistic Regression (LR), K-Nearest Neighbour (KNN), Support Vector Machine (SVM), Multilayer Perceptron (MLP), and Decision Tree (J48)	It focuses on selecting the most appropriate learning techniques for fault prediction modelling.	The approach effectively predicted software faults, enhancing reliability and quality.	These challenges underline the need for comprehensive experimental analysis and cost-benefit evaluation in SFP approaches.
[23]	Nature-Inspired Approaches in Software Fault Prediction	Nature-inspired algorithms, Ant Colony, Particle, Swarm Optimization, Firefly, Bat, Harris Hawks, and Genetic Algorithm,	The research aims to investigate the performance of various nature-inspired optimization algorithms for software fault prediction	The Firefly algorithm performed best on the CM1 dataset, achieving an accuracy of 79.38% with only 13 features.	The paper does not specify the evaluation metrics used to assess the performance of the algorithms
[27]	Software fault prediction using the Whale algorithm with genetics algorithm	Whale algorithms with Genetic algorithms and SVM classifiers	The paper aims to develop a software fault prediction model integrating a genetic algorithm, Whale optimization algorithm, and an SVM classifier	The integration of SVM with optimization algorithms improved prediction performance in terms of accuracy, precision, recall, and F-measure	The paper discusses the challenge of identifying definitive domain requirements during the test case generation process, which can lead to inefficiencies
[38]	Software defect prediction based on kernel PCA and weighted extreme learning machine	KPWE, combining Kernel Principal Component Analysis (KPCA) and Weighted Extreme Learning Machine (WELM)	The research evaluates the performance of KPWE against 41 baseline methods across multiple software projects.	The KPWE method outperforms 41 baseline methods in defect prediction across 44 software projects.	The complex structures of software defect data make it challenging to extract suitable features and learn effective models.

Table-X: Comprehensive Overview of Software Fault Prediction Methodologies/Techniques

Ref. No	Aim of the Study	Methodologies/Techniques	Objectives	Result	Challenges
[10]	Deep Learning Based Software Defect Prediction	support vector regression (SVR), Fuzzy support vector regression (FSVR) is also utilized in the proposed approach.	It aims to evaluate the proposed model's performance using the mean squared error (MSE) and the coefficient of determination.	The paper evaluates the proposed approach using performance metrics, such as MSE and R², through 10-fold cross-validation on two datasets.	The challenge lies in selecting the dataset, which affects defect prediction performance across different datasets.
[11]	Enhancing software defect prediction models using metaheuristics with a learning to rank approach.	TR (Learning To Rank) and metaheuristics optimize.	The study focuses on improving defect density prediction through a hybrid machine learning model.	Hyperparameter tuning with metaheuristics enhanced the overall model.	The complex nature of datasets makes building accurate machine learning models a challenging task.
[15]	Nature-Inspired Approaches in Software Fault Prediction.	Nature-inspired algorithms, Ant Colony, Particle Swarm Optimization, Firefly, Bat, Harris Hawks, and Genetic Algorithm.	The research aims to investigate the performance of various nature-inspired optimization algorithms for software fault prediction.	The Firefly algorithm performed best on the CM1 dataset, achieving an accuracy of 79.38% with only 13 features.	The paper does not specify the evaluation metrics used to assess the performance of the algorithms.
[26]	Software fault prediction using the Whale algorithm with the genetics algorithm.	Whale algorithms with Genetic algorithms and SVM classifiers.	The paper aims to develop a software fault prediction model that integrates a genetic algorithm, a Whale optimisation algorithm, and an SVM classifier.	The integration of SVM with optimization algorithms improved prediction performance in terms of accuracy, precision, recall, and F-measure	The paper discusses the challenge of identifying definitive domain requirements during the test case generation process, which can lead to inefficiencies
[27]	Software defect prediction based on kernel PCA and weighted extreme learning machine	KPWE, combining Kernel Principal Component Analysis (KPCA) and Weighted Extreme Learning Machine (WELM)	The research evaluates the performance of KPWE against 41 baseline methods across multiple software projects.	The KPWE method outperforms 41 baseline methods in defect prediction across 44 software projects.	The complex structures of software defect data make it challenging to extract suitable features and learn effective models.
[44]	A hybrid model of wavelet neural network and metaheuristic An algorithm for software development effort estimation	hybrid model: - neural network (WNN) and metaheuristic algorithms: firefly algorithm and the bat algorithm	The research evaluates the proposed techniques on PROMISE SDEE repositories to assess their effectiveness	The study indicates that the WBG technique performs best on the COCOMO and NASA93 datasets	The paper highlights the challenge of estimating software development effort (SDEE) due to the unknown characteristics of the software at the time of estimation,
[47]	Transfer Learning Code Vectorizer-based Machine Learning Models for Software Defect Prediction.	Use of transfer learning with the Universal Language Model Fine Tuning (ULMFiT) for defect prediction	The paper aims to utilize transfer learning for defect prediction by deriving features from software source code text.	It discusses the correlation between data quality and the performance of machine learning algorithms.	The high computational cost associated with machine learning methods is also mentioned as a challenge.
[54]	A Software Defect Prediction Method Based on Program Semantic Feature Mining	Semantic feature mining (PSFM method)	It focuses on extracting semantic information from source code to enhance defect prediction accuracy.	The paper demonstrates improved performance in software defect prediction compared to other deep learning methods.	The paper highlights that current methods lack features to mine defect manifestations at the semantic level of code.

RQ3: Which Feature Selection Techniques are Commonly Applied in SFP?

Research Question 3 (RQ3) - "What types of feature selection techniques are described and their effectiveness evaluated, to answer the RQ3? " For RQ3, the feature selection techniques used in the studies were analyzed. The feature selection techniques described, along with their effectiveness in determining the generalizability and applicability of the SFP techniques, are presented in **Table 10**. It explores types of feature selection techniques, specific algorithms that are most frequently applied, and certain techniques that are preferred over others. **RQ3**, we can guide future researchers to choose the most effective feature selection techniques, understand the trade-offs among various methods and design more accurate and efficient fault prediction models.

vi. *Software Fault Prediction Datasets, Sizes, Software Metrics and Evaluation Criteria*

Software Fault Prediction (SFP), datasets play a crucial role in training and evaluating models. These datasets typically consist of software metrics (features) and fault labels (target values) for each module (e.g., class, file, function). The goal is to predict whether a software module is faulty or non-faulty. [19]. Commonly Used Datasets in Software Fault Prediction are: -

Datasets: - (*1*) NASA MDP Datasets Provided by the NASA Metrics Data Program (MDP), these are the most widely used datasets in SFP research. Dataset CM1 (Spacecraft instrument software), PC1 (Flight software), KC1 (Storage management for ground support), KC2 (Storage management system), KC3 (Scientific instrument controller), MC1 (Space shuttle software), MC2 (Flight software), JM1 (Real-time predictive ground system), and MW1(Missile warning system) Each dataset typically includes: Software metrics: e.g., LOC, cyclomatic complexity, Halstead metrics, coupling, cohesion. Defect labels: Binary labels indicating fault-prone (1) or fault-free (0) modules [21].

(2*) **PROMISE Repository*** includes NASA datasets and others like JEdit, Ant, and Eclipse. Contains both static code metrics and process metrics (e.g., number of revisions, bug reports) [30]

(*3) **Eclipse Datasets*** (Bug Prediction) Collected from the Eclipse open-source IDE. Data from multiple releases (e.g., 2.0, 2.1, 3.0). Contains metrics from Eclipse's CVS, Bugzilla, and static analysis tools. Often used for temporal and just-in-time defect prediction.

Features (Software Metrics) in SFP Datasets. These datasets generally include the following types of features: Size Metrics - LOC (Lines of Code), Number of Functions/Classes. Complexity Metrics: - Cyclomatic Complexity and Halstead Metrics (volume, effort, bugs). Coupling & Cohesion Metrics and Coupling Between Objects (CBO), Lack of Cohesion in Methods (LCOM). Change Metrics: - (AEEEM, Eclipse) Number of revisions, Number of bug-fixes, Recent changes.

Data Characteristics and Challenges: Imbalanced data: - Most modules are non-faulty; faulty ones are fewer (class imbalance). Noisy data: -Fault labels may be inaccurate or missing. High dimensionality: -Some datasets contain many features, requiring feature selection. Cross-project generalization: - A model trained on one project may not perform well on another [34].

Software Metrics: Common software metrics used in fault prediction or maintenance include Size Metrics, such as lines of code (LOC) and the Number of Functions. Complexity Metrics: - Cyclomatic Complexity and Halstead Metrics. Coupling & Cohesion Metrics: -Coupling Between Objects (CBO) and LCOM (Lack of Cohesion in Methods). Testing Metrics: - Code coverage and Number of test cases. Change/Process Metrics: - Number of revisions, Bug-fix count and Code churn. Object-Oriented Metrics: - DIT (Depth of Inheritance Tree), NOC (Number of Children) [33]

Evaluation Criteria: - Software Fault Prediction (SFP), evaluation criteria refer to the quantitative metrics used to assess the performance of a classifier or prediction model. These metrics indicate how effectively the model can distinguish between faulty and non-faulty software modules [39]. Since SFP is typically applied to imbalanced datasets (containing a small number of faulty instances), selecting the right evaluation metrics is crucial for an accurate and meaningful performance assessment. The most common Key Evaluation Metrics in SFP are Accuracy, which is the overall proportion of correct predictions. Precision: - Proportion of predicted faulty modules that are genuinely faulty. Recall (Sensitivity or True Positive Rate): Proportion of actual faulty modules correctly predicted. F1-Score Meaning: Harmonic mean of Precision and Recall. Specificity (True Negative Rate): Proportion of actual non-faulty modules correctly predicted. [20]

RQ2: What types & typical sizes of datasets are predominantly used in software fault prediction
And
RQ4: Which types of software metrics are utilized in SFP research
And
RQ5: What evaluation criteria are commonly used to measure the performance of SFP models?

Research Question 2 (RQ2) - "What kind of dataset is most used for software fault prediction?" For RQ2, the datasets used in the studies were analyzed. The size of these datasets and the types of programming languages they encompass are critical factors in determining the generalizability and applicability of the SFP techniques. Research Question 4 (RQ4): “What types of software metrics are utilized in SFP research?” For RQ4, the types of software metrics used in SFP research are diverse and designed to capture different facets of code quality and evolution. Selecting the right combination of metrics has a significant impact on the performance of fault prediction models. Modern research often employs hybrid feature sets that combine static, object-oriented, and process metrics to achieve the best results [45]. Research Question 5 (RQ5) “What evaluation criteria are commonly used to measure the performance of SFP models?” Describe through a set of evaluation criteria — especially Accuracy, Precision, Recall, F1-score, AUC-ROC, and AUC-PR, with selection depending on data balance, domain needs, and risk level [46] and described in Table 11.

Table-XI: Comprehensive Overview of Feature Selection Techniques in Software Fault Prediction

Ref. No	Aim of the study	Feature Selection Techniques / Methodologies	Objectives	Result	Challenges
[8]	Software fault prediction using lion optimization algorithm	Lion Optimisation-based Feature Selection (LiOpFS)	The research focuses on selecting optimal feature subsets from high-dimensional defect datasets	LiOpFS outperforms baseline techniques, achieving an AUC of 90.1% and an accuracy of 94.2%	The paper discusses the challenge of the Curse of Dimensionality, which threatens the performance of classifiers in predicting fault-prone software modules
[12]	Issue-Driven Features for Software Fault Prediction	Issue-Driven features	The research aims to evaluate the performance of fault prediction models using issue-driven features compared to traditional feature sets.	Issue-driven features showed a 6% to 13% improvement in AUC across 86 open-source projects.	The paper discusses the challenge of accurately predicting software faults, which is exacerbated by the complexity of modern software systems.
[16]	A Novel Approach to Improve Software Defect Prediction Accuracy Using Machine Learning	Random Forest, Logistic Regression, Multilayer Perceptron, Bayesian Net, Rule ZeroR, J48, Lazy IBK, Support Vector Machine, Neural Networks, and Decision Stump	The objective is to enhance defect prediction accuracy by applying feature selection techniques to five NASA datasets: CM1, JM1, KC2, KC1, and PC1.	On average, feature selection enhances defect prediction accuracy by 5%	The proposed method heavily relies on feature selection techniques to enhance prediction accuracy
[24]	Feature Selection Using Golden Jackal Optimization for Software Fault Prediction	Golden Jackal Optimization (GJO) algorithm, (which is inspired by the hunting tactics of golden jackals.)	The paper aims to apply effective feature selection methods to identify a precise and interpretable model.	FSGJO outperformed other feature selection techniques, achieving higher accuracy in most datasets tested.	The performance of the GJO algorithm varies significantly depending on the characteristics of the dataset used.
[26]	Boosted Whale Optimization Algorithm With Natural Selection Operators for Software Fault Prediction	Whale Optimization Algorithm (WOA)	The paper aims to propose new variants of the Whale Optimization Algorithm (WOA) for feature selection in software fault prediction (SFP) applications	new variants of the Whale Optimization Algorithm (WOA) as wrapper algorithms specifically designed to address feature selection challenges in SFP applications	Class imbalance in software fault prediction datasets poses a significant challenge
[29]	Software Defect Prediction Using Variant-based Ensemble Learning and Feature Selection Techniques	Variant-based ensemble learning and feature selection techniques	The research aims to propose a classification framework for predicting defect-prone software modules, thereby reducing testing costs in software development.	The proposed framework achieved F-Measure, Accuracy, and MCC scores of 0.507, 84.974, and 0.488 on the JM1 dataset.	The paper discusses the challenge of high costs associated with the testing process in software development, particularly when fixing defects during testing, which can lead to increased project completion time.
[30]	Performance Analysis of Feature Selection Methods in Software Defect Prediction: A Search Method Approach	selection (FSS) methods: Correlation-based Feature Subset Selection (CFS) and Consistency Feature Subset Selection (CNS)	The paper aims to analyze the performance of various feature selection (FS) methods in software defect prediction (SDP) models	It highlights that the performance of FS methods varies across datasets and classifiers	The selection of performance metrics, such as accuracy and stability measures, poses challenges in evaluating prediction models

Table-XII: Comprehensive Overview of Feature Selection Techniques in Software Fault Prediction

Ref. No	Aim of the Study	Feature Selection Techniques / Methodologies	Objectives	Result	Challenges
[31]	Feature selection using the firefly algorithm in software defect prediction	firefly algorithm (FA)	The study proposes a new technique inspired by the behaviour of fireflies for effective feature selection.	Support Vector Machine (SVM) with feature selection (FS) achieved a classification accuracy that was 4.53% higher than SVM without FS.	The need for effective feature selection to manage software quality and predict defects is emphasized as a challenge
[32]	A Classification Framework for Software Defect Prediction Using Multi-Filter Feature Selection Technique and MLP	Multi-Filter Feature Selection Technique and Multi-Layer Perceptron (MLP)	The framework is designed to enhance software quality by identifying modules that require thorough testing	MLP-FS-ROS outperformed in F-Measure and MCC, while MLP-FS excelled in Accuracy	The paper discusses the challenge of class imbalance in datasets, which can significantly affect the performance of classification techniques.
[34]	A Framework for Software Defect Prediction Using Feature Selection and Ensemble Learning Techniques	Wrapper approach with Artificial Neural Network (MLP) AND search methods: Best First, Greedy Stepwise, Genetic Algorithm, Particle Swarm Optimization, Rank Search, and Linear Forward Selection	The research aims to develop a framework for software defect prediction using feature selection and ensemble learning techniques.	The results indicate that no single search method consistently outperformed base classifiers across all datasets.	No classification technique achieved 100% accuracy, indicating inherent challenges in software defect prediction.
[35]	Classification framework for faulty software using an enhanced exploratory whale optimiser-based feature selection scheme and random forest ensemble learning	Binary Whale Optimization Algorithm (BWOA)	The objective is to enhance classification performance by selecting the most informative features and eliminating those that are irrelevant.	The study highlights the high accuracy achieved despite the complexity and computation cost of the proposed methodology.	The paper highlights the need for efficient methods to improve algorithm performance, given the No Free Lunch argument.
[36]	An effective feature selection-based cross-project defect prediction model for software quality improvement	cross-project defect prediction (CPDP): MIC_SM_FS and BPSO_FS. (Binary particle swarm optimization algorithm)	The paper aims to propose a novel CPDP approach with two distinct feature selection strategies, one non-iterative and one iterative	BPSO_FS achieved comparable performance to baseline ALL with a 65% reduction in features	The paper discusses the challenge of distribution dissimilarity between source and target project data, which limits the capability of cross-project defect prediction (CPDP) models
[38]	3PcGE: 3-parent child-based genetic evolution for software defect prediction	3PcGE (three-parent child-based genetic evolution)	The research aims to demonstrate that 3PcGE enhances the performance of SDP classifiers as a feature selector	The proposed 3PcGE technique outperforms filter-based FS techniques by 18.98% in the AUC measure	The challenge of this work will involve expanding datasets and exploring additional multi-objective algorithms
[41]	Optimal Feature Selection through Search-Based Optimizer in Cross-Project	Search-based optimizer	The research aims to select optimal features from multi-class data for cross-project defect prediction (CPDP) using a search-based optimizer.	The research demonstrated that feature selection enhances prediction accuracy in cross-project defect prediction scenarios.	The paper discusses the challenge of heterogeneous data in cross-project defect prediction (CPDP), affecting model performance.
[42]	A two-stage transformer fault diagnosis [43] method based on multi-filter interactive feature selection, integrated adaptive Sparrow, and an algorithm-optimised support vector machine	multi-filter interactive feature selection method (MIFS) and adaptive sparrow algorithm (ASSA) optimized support vector machine (SVM). (ASSA-SVM)	It proposes a two-stage integration model, MIFS-ASSA-SVM, for improved feature selection and parameter optimization	The optimal feature subset selected under ReliefF-mRMR showed the highest mean accuracy and the least dimension	Redundant high-dimensional feature sets can waste computing power and complicate fault identification

Table-XIII: Comprehensive Overview of Feature Selection Techniques in Software Fault Prediction

Ref. No	Aim of the Study	Feature Selection Techniques / Methodologies	Objectives	Result	Challenges
[51]	A Study on Software Defect Prediction using Feature Extraction Techniques	feature extraction techniques: Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Kernel-based Principal Component Analysis (K-PCA), and Autoencoders. And Support Vector Machine (SVM) classifier	It explores the impact of different feature extraction methods on software defect prediction	The study found that Autoencoders achieved the highest performance based on ROC-AUC with a p-value of 0.009	The findings are based on experimental results without real-world validation.
[48]	A software defect prediction method with metric compensation based on feature selection and transfer learning	peUpMeCom, which integrates metric compensation based on transfer learning and Pearson feature selection	The research introduces a metric compensation technique to handle significant distribution differences between source and target projects	The proposed method improves defect prediction accuracy using metric compensation based on feature selection and transfer learning	The main external threat to validity identified in the paper is dataset bias, which raises concerns about the generalizability of the proposed defect prediction model
[57]	Cost-Effective Prediction Model for Optimal Selection of Software Faults Using Coati Optimization Algorithm	A novel feature selection (FS) method called FS using Coati Optimization Algorithm (FSCOA) and classifiers: K-Nearest Neighbors (KNN), Quadratic Discriminant Analysis (QDA), Decision Trees (DT), and Naive Bayes (NB)	The algorithm aims to select relevant and optimal subsets of features from large datasets to enhance the performance of the machine learning model.	The FSCOA model outperformed other feature selection algorithms in over 90% of test cases.	Data Quality Issues, Curse of Dimensionality and Optimization Algorithm Challenges of this Paper

Table-XIV: Comprehensive Overview of Software Fault Prediction Datasets, Sizes, Software Metrics and Evaluation Criteria

Ref. No	Aim of the Study	Datasets Types / Sizes	Software Metrics	Evaluation Criteria	Result	Challenges
[2]	Software Fault Prediction Using an RNN-Based Deep Learning Approach and Ensemble Machine Learning Techniques	Eclipse (Java-based open-source), Apache Active MQ (JIRA bug repository) and large sample sizes.	Chidamber and Kemer (CK) metrics, Object-Oriented (OO) metrics, and entropy metrics	Accuracy (ACC), Area Under Curve (AUC), F-measure (FM), Cohen's Kappa (KE), Precision, Recall, True Negative Rate	ensemble ML Accuracy: 94.38% (Random Tree with RF) on Apache Active MQ and 79.93% (SVM with RF) on the Eclipse dataset	Development of a new dataset using JavaDoc documents and integrating transfer learning into the newly developed dataset
[3]	BPDET: An Effective Software Bug Prediction Model using Deep Representation and Ensemble Learning Techniques	12 data sets from NASA's PROMISE	Basic Halsted, Derived Halsted, and McCabe	ROC (receiver operating characteristic curve), F-measure, Matthew's correlation coefficient (MCC) and precision-recall area (PRC)	The MCC values of BPDET is highest for CM1 (0.420),	It highlights issues such as missing values, data redundancy, and irrelevant features that hinder the effective detection of faulty modules.
[4]	An empirical study of ensemble techniques for software fault prediction	28 benchmarked software fault datasets from the PROMISE data repository	object-oriented software metrics	precision, recall, AUC (area under the ROC curve), specificity, and G-means	Ensemble techniques produced mean values greater than 0.7 for most performance measures.	The study indicates that previous analyses were restricted to a few fault datasets and ensemble techniques
[7]	An ANN-Based Approach for Software Fault Prediction Using Object-Oriented Metrics	18 public datasets from the PROMISE repository	object-oriented metrics	ROC-AUC (receiver operating characteristics area under the curve), accuracy, and mean squared error (MSE	The accuracy of the proposed model ranges from 92% to 93%, demonstrating high prediction reliability.	The selection of appropriate metrics for fault prediction is identified as a critical challenge.
[9]	Software fault prediction based on the dynamic selection of learning technique: findings from the Eclipse project study	5 Eclipse project datasets: JDT core, PDF UI, Equinox framework, Lucene, and Mylyn	Object-oriented	accuracy, AUC, sensitivity, and specificity	Accuracy for Eclipse datasets was at least 0.70, peaking at 0.877	The paper discusses the challenge of selecting suitable learning techniques for SFP due to variations in prediction performance across different software systems.
[18]	Evaluating the effectiveness of decomposed Halstead Metrics in software fault prediction	5 public datasets	Halstead base metrics, McCabe metrics, and Lines of Code (LoC).	Accuracy, F-measure, and Area Under Curve (AUC)	Accuracy improved from 0.82 to 0.97, F-measure from 0.81 to 0.99, and AUC from 0.79 to 0.99	The research indicates that different datasets may exhibit distinct characteristics affecting performance.

Table-XV: Comprehensive Overview of Software Fault Prediction Datasets, Sizes, Software Metrics and Evaluation Criteria

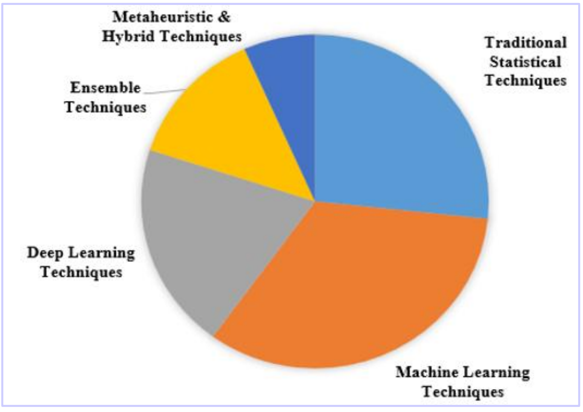
Ref. No	Aim of the Study	Datasets Types / Sizes	Software Metrics	Evaluation Criteria	Result	Challenges
[27]	Software Metrics Selection for Fault Prediction: A Review	limited number of datasets	Lines of Code (LOC).	Not specified	variation in metric performance across different programming languages, suggesting further investigation is needed.	Achieving high software quality while meeting predetermined goals is a significant challenge due to traditional testing limitations
[42]	Performance Evaluation of various ML techniques for Software Fault Prediction using the NASA dataset	JM1 provided by NASA	21 features	Accuracy (ACC), recall, precision, and F1-score	RF classifier showed improved accuracy of 92.28140% with random sampling	Software developers face significant challenges in creating high-quality software, requiring adherence to a series of actions and restrictions
[47]	Software Fault Prediction Using LSSVM with Different Kernel Functions	PROMISE repository and 15 fault prediction datasets are used	object-oriented	Accuracy, F−Measure, True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN)	The study analysed the results using box plots, which showed high median values for the LSSVM models.	The paper does not address the number of faults present in a module, which is a significant challenge in fault prediction.
[53]	Software Quality Prediction: An Investigation Based on Artificial Intelligence Techniques for Object-Oriented Applications	PROMISE data repository and (8) eight open-source real-world software projects such as Tomcat, Velocity, Ivy, Jedit, Workflow, Poi, Forrest, Ant	object-oriented software metrics.	Accuracy, Precision and Recall	Bagging and Random Forest techniques achieved the highest AUC values, indicating strong predictive performance.	Software reliability is challenged by the complexity of software and its associated defect rates.

III. DISCUSSION AND ANALYSIS

In this section, an analysis and discussion of the considered research papers are presented in the form of answering the proposed research questions (RQ). The answers to the research questions are entirely based on the comparative analysis. This comprehensive review aimed to unravel the intricacies and current trends in Software Fault Prediction (SFP), addressing several pertinent research questions as detailed in the earlier sections of this paper. A total of 71 relevant studies were carefully selected and analysed to support the summarised findings.

A. RQ1: What are the Most Commonly used Methodologies/Techniques for Software Fault Prediction?

During the analysis of Table 8, it has been observed that among all Methodologies/techniques used in software fault prediction, Machine Learning Techniques are the most commonly used. The analysis of the most widely used methodologies and techniques for software fault prediction is shown in Fig. 4.



[Fig.4: Analyses of Software Fault Prediction Methodologies/Techniques]

B. RQ2: What types & typical sizes of datasets are predominantly used in software fault prediction studies?

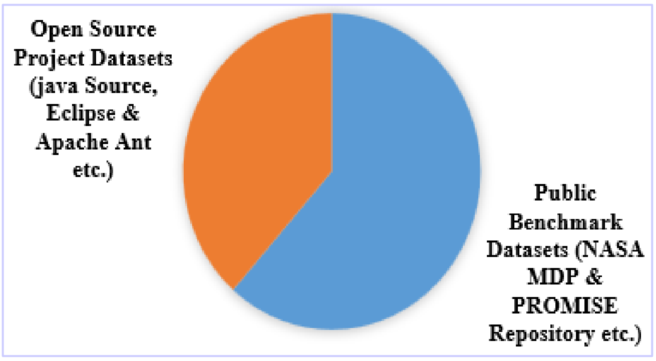
Software Fault Prediction studies predominantly rely on small to medium-sized benchmark datasets, such as those from NASA and PROMISE, due to their accessibility and standardization. However, there is a growing trend toward using open-source project data and industrial datasets for enhanced realism and scalability [56]. The typical sizes of datasets & Types used in software fault prediction studies by different researchers are shown in Table 11. and analysis, as shown in Fig. 5

C. RQ3: Which Feature Selection Techniques are Commonly Applied in SFP?

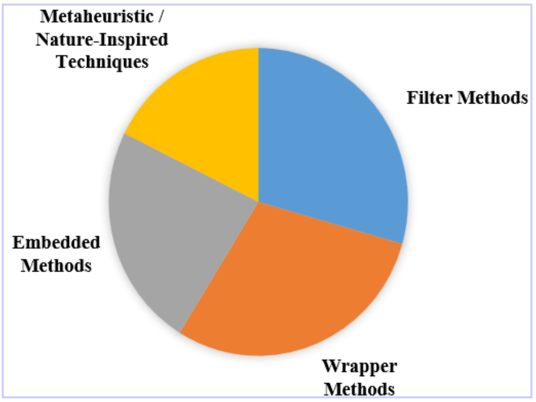
The selection techniques used by most researchers for software fault prediction are Filter Methods, Wrapper Methods, and Embedded Methods. Studies of feature selection techniques are commonly applied in SFP, shown in Table 10, and the analysis is shown in Fig. 6

D. RQ4: Which types of software metrics are utilized in SFP research?

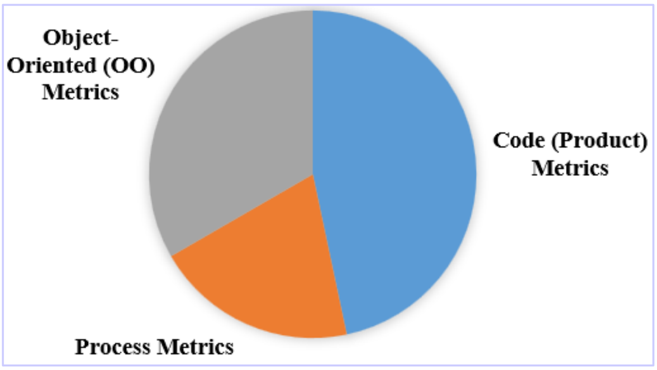
In Software Fault Prediction (SFP) research, code metrics are the most widely used due to their availability and ease of extraction from source code. However, process metrics and object-oriented design metrics are increasingly being adopted to capture development history and structural design features [37]. Studies of software metrics are utilized in SFP research, shown in Table 11, and the analysis is shown in Fig. 7



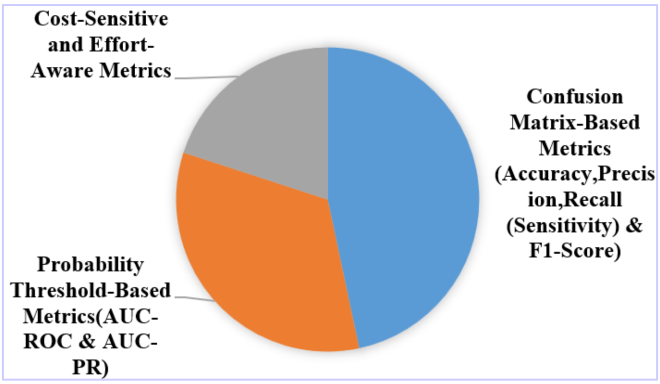
[Fig.5: Analyses of Types Datasets are Used in Software Fault Prediction Studies]



[Fig.6: Analyses of Feature Selection Techniques Software Fault Prediction Studies]



[Fig.7: Analyses of Types of Software Metrics are Utilized in SFP]



[Fig.8: Analyses of Evaluation Criteria are Used to Measure the Performance of SFP]

E. RQ4: Which Types of Software Metrics are Utilised in SFP Research?

In Software Fault Prediction (SFP) research, code metrics are the most widely used due to their availability and ease of extraction from source code. However, process metrics and object-oriented design metrics are increasingly being adopted to capture development history and structural design features. Studies of software metrics are utilised in SFP research, as shown in Table 11, and the analysis is presented in Fig. **7.**

F. RQ5: What Evaluation Criteria are Commonly used to Measure the Performance of SFP Models?

In SFP research, a range of evaluation criteria is used to assess prediction models comprehensively. While accuracy is simple and commonly reported, it can be misleading in imbalanced datasets. Therefore, precision, recall, F1-score, and AUC-ROC/AUC-PR are more reliable for measuring performance, especially for detecting faulty modules. Moreover, cost-sensitive and effort-aware metrics are increasingly emphasized to ensure the practical utility of SFP models in real-world settings [49]. Studies of evaluation criteria are commonly used to measure the performance of SFP models, as shown in **Table 11** and the analysis shown in Fig. 8

IV. CONCLUSION

This review has systematically explored methodologies, datasets, feature selection techniques, and evaluation criteria in Software Fault Prediction (SFP). The study highlights the significant role machine learning models, particularly neural networks, deep learning, and ensemble methods, play in advancing SFP by effectively handling the complexities of software fault datasets. The widespread use of repositories like PROMISE and NASA emphasises the importance of diverse datasets in enhancing model accuracy. However, persistent challenges such as data imbalance, the fast-evolving nature of software development, and the demand for real-time prediction remain key hurdles.

In this paper, an SLR is conducted to track the most recent research advances in techniques for software defect prediction. This review is conducted after critically analysing the most relevant research papers published in three well-known online libraries: ACM, IEEE, SpringerLink, and ScienceDirect. Five research questions regarding the different aspects of research progress on the use of SFP techniques, Dataset, feature selection, software metrics & evaluation criteria for software defect prediction are defined and addressed in this study. From the comparative analysis, it has been observed that techniques such as Machine Learning Techniques and Traditional Statistical Techniques are primarily used by Researchers, with the most common use of Feature Selection Techniques, including Wrapper Methods, and Public Benchmark

Datasets (e.g., NASA MDP & PROMISE Repository). Confusion Matrix-Based Metrics (Accuracy, Precision, Recall (Sensitivity) & F1-Score). Moreover, the diversity of classifiers used in building the SFP model should also be investigated to improve the effectiveness and quality of the software.

ACKNOWLEDGMENTS

The first author would like to thank the Computing Lab of Sri Guru Granth Sahib World University, Fatehgarh Sahib, Punjab, India, for successfully carrying out the research work.

DECLARATION STATEMENT

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

- **Conflicts of Interest/ Competing Interests:** Based on my understanding, this article has no conflicts of interest.
- **Funding Support:** This article has not been funded by any organizations or agencies. This independence ensures that the research is conducted with objectivity and without any external influence.
- **Ethical Approval and Consent to Participate:** The content of this article does not necessitate ethical approval or consent to participate with supporting documentation.
- **Data Access Statement and Material Availability:** The adequate resources of this article are publicly accessible.
- **Author's Contributions:** Rajinder Kumar: Data curation, Methodology, Investigation, writing – original draft, Writing – review and editing, Visualisation. Kamaljit Kaur: Conceptualisation, Investigation, Writing – review and editing, Supervision

REFERENCES

1. Matloob, F., Ghazal, T. M., Taleb, N., Aftab, S., Ahmad, M., Khan, M. A., Soomro, T. R. (2021). Software defect prediction using ensemble learning: A systematic literature review. IEEE Access.https://www.researchgate.net/publication/353107026Software_Defect_Prediction_Using_Ensemble_Learning_A_Systematic_Literature_Review
2. Borandag, E. (2023). Software Fault Prediction Using an RNN-Based Deep Learning Approach and Ensemble Machine Learning Techniques. Applied Sciences, 13(3), 1639. https://www.mdpi.com/2076-3417/13/3/1639
3. Pandey, S. K., Mishra, R. B., & Tripathi, A. K. (2020). BPDET: An effective software bug prediction model using deep representation and ensemble learning techniques. Expert Systems with Applications, 144, 113085. https://www.Sciencedirect.com/science/article/abs/pii/S0957417419308024?utm_source.
4. Rathore, S. S., & Kumar, S. (2021). An empirical study of ensemble techniques for software fault prediction. Applied Intelligence, 51, 3615-3644. https://link.springer.com/article/10.1007/s10489-020-01935-6?utm_source.
5. Phung, K., Ogunshile, E., & Aydin, M. (2021, October). A novel software fault prediction approach to predict error-type proneness in Java programs using stream X-machine and machine learning. In 2021, the 9th International Conference on Software Engineering Research and Innovation (CONISOFT) (pp. 168-179). IEEE.https://uwe-repository.worktribe.com/output/ 7605934/a-novel-software-fault-prediction-approach-to-predict-error-type-proneness-in-the-java-programs-using-stream-x-machine-and-machine-learning?utm_source.
6. Alfredo Daza, (2025) Software defect prediction based on a multi-classifier with hyperparameters: Future work. www.sciencedirect.com/journal/results-in-engineering. https://doi.org/10.1016/j.rineng.2025.104123
7. Barbara Wi̜eckowska, Katarzyna B. Kubiak, Paulina Jozwiak, Wacław Moryson and Barbara Stawinska-Witoszynska (2022). Cohen's Kappa Coefficient as a Measure to Assess Classification Improvement following the Addition of a New Marker to a Regression Model. International Journal of Environmental Research and Public Health. https://www.mdpi.com/1660-4601/19/16/10213?utm_source.
8. Goyal, S., & Bhatia, P. K. (2021). Software fault prediction using lion optimization algorithm. International Journal of Information Technology, 13, 2185-2190. https://ouci.dntb.gov.ua/en/works/7ABmB1a4/?utm_source.
9. Rathore, S. S., & Kumar, S. (2021). Software fault prediction based on the dynamic selection of learning technique: findings from the eclipse project study. Applied Intelligence, 1-16. https://link.springer.com/content/pdf/10.1007/s10489-021-02346-x.pdf?utm_source.
10. Qiao, L., Li, X., Umer, Q., & Guo, P. (2020). Deep learning based software defect prediction. Neurocomputing, 385, 100-110. https://colab.ws/articles/10.1016%2Fj.neucom.2019.11.067?utm_source.
11. Aryan Boloori, Azadeh Zamanifar, Amirfarhad Farhadi (2024). Enhancing software defect prediction models using metaheuristics with a learning to rank approach. https://doi.org/10.1007/s44248-024-00016-0
12. Amir Elmishali and Meir Kalech (2022). Issue-Driven Features for Software Fault Prediction, Software and Information Systems Engineering. https://dblp.org/rec/journals/infsof/ ElmishaliK23?utm_source.
13. Kaur, G., Pruthi, J., & Gandhi, P. (2023). Machine Learning-Based Software Fault Prediction Models. Karbala International Journal of Modern Science, 9(2). https://kijoms.uokerbala.edu.iq/home/vol9/iss2/9/?utm_source.
14. Rajput, P. K., Aarti, & Pal, R. (2023, February). Genetic Algorithm-Based Clustering with Neural Network Classification for Software Fault Prediction. In Proceedings of International Conference on Data Science and Applications: ICDSA 2022, Volume 1 (pp. 399-414). https://ebin.pub/proceedings-of-international-conference-on-data-science-and-applications-icdsa-2022-volume-1-9811966303-9789811966309.html?utm_source.
15. ARORA, T., SAINI, H., & GARG, S. (2023). Nature-Inspired Approaches in Software Fault Prediction. JUN 2023 | IRE Journals | Volume 6 Issue 12 | ISSN: 2456-8880. https://cse.mait.ac.in/index.php/campus-life/r-dlab/research-publications/9-computer-center/1254-details-of-paper-published-in-journal-international-national-during-2023-24?utm_source.
16. Mehmood, I., Shahid, S., Hussain, H., Khan, I., Ahmad, S., Rahman, S., & Huda, S. (2023). A Novel Approach to Improve Software Defect Prediction Accuracy Using Machine Learning. IEEE Access. https://dblp.org/pid/351/0094?utm_source.
17. Wang, Z., Tong, W., Li, P., Ye, G., Chen, H., Gong, X., & Tang, Z. (2023). BugPre: an intelligent software version-to-version bug prediction system using graph convolutional neural networks. Complex & Intelligent Systems, 9(4), 3835-3855. https://ouci.dntb.gov.ua/en/works/98oBGYjl/?utm_source.
18. Khan, B., & Nadeem, A. (2023). Evaluating the effectiveness of decomposed Halstead Metrics in software fault prediction. PeerJ Computer Science, 9, e1647. https://ouci.dntb.gov.ua/en/ works/ ldkAogk4/?utm_source.
19. Al Qasem, O., Akour, M., & Alenezi, M. (2020). The influence of deep learning algorithms on software fault prediction. IEEE Access, 8, 63945-63960. https://malenezi.github.io/malenezi/pdfs/09055422.pdf?utm_source.
20. Mohsen Hesamolhokama, Amirahmad Shafiee, Mohammadreza Ahmaditeshnizi, Mohammadamin Fazli, Jafar Habibi(2024), SDPERL: A Framework for Software Defect Prediction Using Ensemble Feature Extraction and Reinforcement Learning, arXiv:2412.07927v2. https://arxiv.org/abs/ 2412.07927 ?utm_source.
21. Khleel, N. A. A., & Nehéz, K. (2023). Software defect prediction using a bidirectional LSTM network combined with oversampling techniques. Cluster Computing, 1-24. https://link.springer.com/article/10.1007/s10586-023-04170-z?utm_source.
22. Khalid, A., Badshah, G., Ayub, N., Shiraz, M., & Ghouse, M. (2023). Software Defect Prediction Analysis Using Machine Learning Techniques. Sustainability, 15(6), 5517.

https://doi.org/10.3390/su15065517
23. Sofian Kassaymeh, Salwani Abdullah, Ph.D, Mohammed Azmi Al-Betar (2021). Salp swarm optimiser for modelling the software fault prediction problem, Journal of King Saud University – Computer and Information Sciences 34 (2022) 3365–3378. https://www.sciencedirect.com/science/article/pii/S1319157821000173?utm_source.
24. Das, H., Prajapati, S., Gourisaria, M. K., Pattanayak, R. M., Alameen, A., & Kolhar, M. (2023). Feature Selection Using Golden Jackal Optimization for Software Fault Prediction. Mathematics, 11(11),2438.https://www.mdpi.com/2227-7390/11/11/2438?utm_source,
25. Feng, S., Keung, J., Yu, X., Xiao, Y., Bennin, K. E., Kabir, M. A., & Zhang, M. (2021). COSTE: Complexity-based OverSampling Technique to alleviate the class imbalance prob- lem in software defect prediction. Information and Software Technology, 129, 106432. https://bibbase.org/network/publication/feng-keung-yu-xiao-bennin-kabir-zhang-coste-complexity-based-over-sampling-technique-to-alleviate-the-class-imbalance-problem-in-software-defect-prediction-2021?utm_source.
26. Hassouneh, Y., Turabieh, H., Thaher, T., Tumar, I., Chantar, H., & Too, J. (2021). Boosted whale optimization algorithm with natural selection operators for software fault prediction. IEEE Access, 9, 14239-14258. https://jeeemi.org/index.php/jeeemi/ article/view/334?utm_source=chatgpt.com
27. Anil Kumar Pandey and Manjari Gupta (2024), Software Metrics Selection for Fault Prediction: A Review, International Journal of Management, Technology and Engineering, ISSN NO: 2249-7455. https://www.researchgate.net/publication/382888111 _Software_Metrics_Selection_for_Fault_Prediction_A_Review?utm_source.
28. Zhao, K., Xu, Z., Yan, M., Zhang, T., Xue, L., Fan, M., & Keung, J. (2023). The Impact of Class Imbalance Techniques on Crash Fault Residence Prediction Models. Empirical Software Engineering, 28(2), 49. https://yanmeng.github.io/papers/EMSE231.pdf?utm_source.
29. Ali, U., Aftab, S., Iqbal, A., Nawaz, Z., Bashir, M. S., & Saeed, M. A. (2020). Software defect prediction using variant-based ensemble learning and feature selection techniques. Int. J. Mod. Educ. Comput. Sci, 12(5), 29-40. https://www.mecs-press.org/ijmecs/ijmecs-v12-n5/v12n5-3.html?utm_source.
30. Balogun, A. O., Basri, S., Abdulkadir, S. J., & Hashim, A. S. (2019). Performance analysis of feature selection methods in software defect prediction: a search method approach. Applied Sciences, 9(13), 2764. https://www.mdpi.com/2076-3417/9/13/2764?utm_source.
31. Anbu, M., & Anandha Mala, G. S. (2019). Feature Selection Using the Firefly Algorithm in Software Defect Prediction. Cluster Computing, 22, 10925-10934. https://jisem-journal.com/index.php/journal/article/download/6277/2891/10449?utm_source.
32. Iqbal, A., & Aftab, S. (2020). A Classification Framework for Software Defect Prediction Using Multi-Filter Feature Selection Technique and MLP. International Journal of Modern Education Computer Science, 12(1). https://www.mecs-press.org/ijmecs/ijmecs-v12-n1/v12n1-3.html?utm_source.
33. Balogun, A. O., Basri, S., Mahamad, S., Abdulkadir, S. J., Almomani, M. A., Adeyemo, V. E., .& Bajeh, A. O. (2020). Impact of feature selection methods on the predictive performance of software defect prediction models: An extensive empirical study. Symmetry, 12(7), 1147. https://www.mdpi.com/2073-8994/12/7/1147?utm_source.
34. Rathi, S. C., Misra, S., Colomo-Palacios, R., Adarsh, R., Neti, L. B. M., & Kumar, L. (2023). Empirical evaluation of the performance of data sampling and feature selection techniques for software fault prediction. Expert Systems with Applications, 223, 119806. https://www.researchgate.net/publication/369306462_Empirical_evaluation_of_the_performance_of_data_sampling_and_feature_selection_techniques_for_software_fault_prediction?utm_source.
35. Mafarja, M., Thaher, T., Al-Betar, M. A., Too, J., Awadallah, M. A., Abu Doush, I., & Turabieh, H. (2023). Classification framework for faulty software using an enhanced exploratory whale optimiser-based feature selection scheme and random forest ensemble learning. Applied Intelligence, 1-43. https://link.springer.com/article/10.1007/s10489-022-04427-x?utm_source.
36. Yogita Khatri Sandeep Kumar Singh (2022), An effective feature selection-based cross-project defect prediction model for software quality improvement, Int J Syst Assur Eng Manag (March 2023) 14(Suppl. 1): S154–S172. https://ideas.repec.org/a/spr/ijsaem/v14y2023i1d10.1007_s13198-022-01831-x.html?utm_source.
37. Shiqi Tang, Song Huang, Changyou Zheng, Erhu Liu, Cheng Zong, and Yixian Ding (2022), A Novel Cross-Project Software Defect Prediction Algorithm Based on Transfer Learning, TINGHUA SCIENCE AND TECHNOLOGY, ISSN 1007- 0214, 04/18 pp. 41–57 DOI: 10.26599/TST.2020.9010040. https://www.sciopen.com/article/10.26599/TST.2020.9010040?utm_source.
38. Goyal, S. (2023). 3PcGE: 3-parent child-based genetic evolution for software defect prediction. Innovations in Systems and Software Engineering, 19(2), 197-216. https://link.springer.com/article/10.1007/s11334-021-00427-1?utm_source.
39. Aarti, A., Rajput, P. K., & Khare, A. (2023, April). Hybrid semi-supervised SOM-based clustered approach with genetic algorithm for software fault classification. In AIP Conference Proceedings (Vol. 2724, No. 1). AIP Publishing. https://www.researchgate.net/publication/370379196_Hybrid_semisupervised_SOM_based_clustered_approach_with_genetic_algorithm_for_software_fault_classification?utm_source.
40. Khatri, Y., & Singh, S. K. (2023). An effective software cross-project fault prediction model for quality improvement. Science of Computer Programming, 226, 102918. https://ideas.repec.org/a/spr/ijsaem/v14y2023i1d10.1007_s13198-022-01831-x.html?utm_source.
41. Faiz, R. B., Shaheen, S., Sharaf, M., & Rauf, H. T. (2023). Optimal Feature Selection through Search-Based Optimizer in Cross-Project. Electronics, 12(3), 514. https://doi.org/10.3390/electronics12030514.
42. Baraah Alsangari & Göksel Biricik (2023) Performance Evaluation of various ML techniques for Software Fault Prediction using NASA dataset. 5th International Congress on Human-Computer Interaction, Optimization and Robotic Applications. https://www.proceedings.com/content/069/069589webtoc.pdf?utm_source.
43. Hanyu Shi & Mingxia Chen (2022) A two-stage transformer fault diagnosis method based on multi-filter interactive feature selection, integrated adaptive sparrow algorithm, optimised support vector machine, IET Electric Power Applications. DOI: 10.1049/elp2.12270.https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/elp2.12270?utm_source.
44. Sagheer Abbas, Shabib Aftab, Muhammad Adnan Khan, Taher M. Ghazal, Hussam Al Hamadi and Chan Yeob Yeun (2023), Data and Ensemble Machine Learning Fusion-Based Intelligent Software Defect Prediction System, DOI: 10.32604/cmc. 2023.037933. https://www.techscience.com/cmc/v75n3/52611?utm_source.
45. abdullah sharaf , Amin y. noaman, and Asaad ahmed (2023), Prediction and Correction of Software Defects in Message-Passing Interfaces Using a Static Analysis Tool and Machine Learning, IEEE Access. https://sciprofiles.com/profile/3095509?utm_source.
46. Al Qasem, O., Akour, M., & Alenezi, M. (2020). The influence of deep learning algorithms is a factor in software prediction. IEEE Access, 8, 63945-6396. https://malenezi.github.io/malenezi/pdfs/09055422.pdf?utm_source.
47. Kulamala, V. K., Kumar, L., & Mohapatra, D. P. (2021). Software fault prediction using LSSVM with different kernel functions. Arabian Journal for Science and Engineering, 46, 8655-8664. https://link.springer.com/article/10.1007/s13369-021-05643-2?utm_source.
48. Jinfu CHEN, Xiaoli WANG, Saihua CAI, Jiaping XU, Jingyi CHEN, Haibo CHEN (2022), A software defect prediction method with metric compensation based on feature selection and transfer learning, Chen et al. / Front Inform Technol Electron Eng. https://link.springer.com/article/10.1631/FITEE.2100468?utm_source.
49. Anupama Kaushik & Niyati Singal (2022) A hybrid model of wavelet neural network and metaheuristic algorithm for software development effort estimation, Int. j. inf. tecnol.. 14(3):1689–1698,.https://link.springer.com/journal/41870/volumes-and-issues/14-3?page=2&utm_source.
50. HAQUE, ALI, MCCLEAN & NOPPEN (2024), Heterogeneous Cross-Project Defect Prediction Using Encoder Networks and Transfer Learning, IEEE Access, 10.1109/ACCESS.2023.3343329. https://pure.ulster.ac.uk/files/130014670/Heterogeneous_Cross-Project_Defect_Prediction_using_Encoder_and_Transfer_Learning.pdf?utm_source.
51. Malhotra, R., & Khan, K. (2020). A study on software defect prediction using feature extraction techniques. In 2020, the 8th International Conference on Reliability, Infocom Technologies and Optimization (pp. 1139-1144). IEEE. https://www.researchgate.net/ publication/ 344983707_A

_Study_on_Software_Defect_Prediction_using_Feature_Extraction_Techniques?utm_source.
52. Waleed Albattah and Musaad Alzahrani (2024), Software Defect Prediction Based on Machine Learning and Deep Learning Techniques: An Empirical Approach, https:// doi.org/10.3390/ai5040086.https://www.mdpi.com/2673-2688/5/4/ 86?utm_source.
53. İlhan, Ö., & Erçelebi Ayyıldız, T. (2021). Software Quality Prediction: An Investigation Based on Artificial Intelligence Techniques for Object-Oriented Applications. In Trends in Data Engineering Methods for Intelligent Systems: Proceedings of the International Conference on Artificial Intelligence and Applied Mathematics in Engineering. https://www.researchgate.net/publication/353005268_Sofware_Quality_Prediction_An_Investigation_Based_on_Artificial_Intelligence_Techniques_for_Object-Oriented_Applications?utm_source.
54. Wenjun Yao, Muhammad Shafiq, Xiaoxin Lin and Xiang YuA (2023), Software Defect Prediction Method Based on Program Semantic Feature Mining, https://www.mdpi.com/2079-9292/12/7/1546?utm_source.
55. S. Kaliraj, Velisetti Geetha Pavan Sahasranth, V. Sivakumar (2024), A holistic approach to software fault prediction with dynamic classification, Automated Software Engineering.
56. Ran YAN, Meichen WANG, Zhaowei XU and Kai ZHANG (2023) Research on Software Fault Feature Data Extraction Method for Software Fault Prediction Technology, Advances in Machinery, Materials Science and Engineering Application IX M. Chen et al. (Eds.). https://www.researchgate.net/publication/ 374791399_Research_on_Software_Fault_ Feature_ Data_ Extraction_ Method_ for_ Software_ Fault_ Prediction_ Technology? utm_source.
57. Hrishikesh Kumar & Himansu Das (2025), Cost-Effective Prediction Model for Optimal Selection of Software Faults Using Coati Optimization Algorithm, SN Computer Science. https://link.springer.com/article/10.1007/s42979-025-03953-y?utm_source.

AUTHOR'S PROFILE

Rajinder Kumar, a Research Scholar, is pursuing her Ph.D. in the Department of Computer Science and Engineering, Sri Guru Granth Sahib World University, Fatehgarh Sahib. (Punjab), India and an Assistant Professor at Chandigarh Business School of Administration, Landran, Mohali, in the Department of Computer Applications. He has an MTech and a BTech in Computer Science from Punjab Technical University, Kapurthala, Punjab. Her areas of interest include Software Engineering, Databases, Data Mining and Machine Learning.

Dr. Kamaljit Kaur is an Assistant Professor in the Department of Computer Science and Engineering, Sri Guru Granth Sahib World University, Fatehgarh Sahib (Punjab), India, with an overall experience of 13 years in academia & research related to Computer Science. He is an avid researcher, having guided dissertations of many M.Tech PG students & is also currently guiding 5 PhD scholars. He is also a member of the reviewer panel for many esteemed, refereed, and peer-reviewed journals. His current research interests include Software Engineering, Cloud Computing, Web Engineering, Big Data, the Internet of Things, database security, and Mobile Computing.