Received 9 November 2023, accepted 3 December 2023, date of publication 14 December 2023, date of current version 29 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3343249

RESEARCH ARTICLE

Analysis of Feature Selection Methods in Software Defect Prediction Models

MISBAH ALI[1], TEHSEEN MAZHAR[1], TARIQ SHAHZAD[2], YAZEED YASIN GHADI[3], SYED MUHAMMAD MOHSIN[4,5], SYED MUHAMMAD ABRAR AKBER[6], AND MOHAMMED ALI[7]

[1]Department of Computer Science, Virtual University of Pakistan, Lahore 55150, Pakistan
[2]Department of Computer Sciences, COMSATS University Islamabad, Sahiwal Campus, Sahiwal 57000, Pakistan
[3]Department of Computer Science and Software Engineering, Al Ain University, Abu Dhabi, United Arab Emirates
[4]Department of Computer Science, COMSATS University Islamabad, Islamabad 45550, Pakistan
[5]College of Intellectual Novitiates (COIN), Virtual University of Pakistan, Lahore 55150, Pakistan
[6]Department of Computer Graphics, Vision and Digital Systems, Faculty of Automatic Control, Electronics and Computer Science, Silesian University of Technology, 44-100 Gliwice, Poland
[7]Department of Computer Science, Applied College, King Khalid University, Abha 61421, Saudi Arabia

Corresponding authors: Tehseen Mazhar (tehseenmazhar719@gmail.com), Syed Muhammad Mohsin (syedmmohsin9@gmail.com), and Syed Muhammad Abrar Akber (abrar.akber@polsl.pl)

This work was supported by the Deanship of Scientific Research (DSR), King Khalid University, Abha, Saudi Arabia, under Grant RGP.1/380/43.

ABSTRACT Improving software quality by proactively detecting potential defects during development is a major goal of software engineering. Software defect prediction plays a central role in achieving this goal. The power of data analytics and machine learning allows us to focus our efforts where they are needed most. A key factor in the success of software fault prediction is selecting relevant features and reducing data dimensionality. Feature selection methods contribute by filtering out the most critical attributes from a plethora of potential features. These methods have the potential to significantly improve the accuracy and efficiency of fault prediction models. However, the field of feature selection in the context of software fault prediction is vast and constantly evolving, with a variety of techniques and tools available. Based on these considerations, our systematic literature review conducts a comprehensive investigation of feature selection methods used in the context of software fault prediction. The research uses a refined search strategy involving four reputable digital libraries, including IEEE Explore, Science Direct, ACM Digital Library, and Springer Link, to provide a comprehensive and exhaustive review through a rigorous analysis of 49 selected primary studies from 2014. The results highlight several important issues. First, there is a prevalence of filtering and hybrid feature selection methods. Second, single classifiers such as Naïve Bayes, Support Vector Machine, and Decision Tree, as well as ensemble classifiers such as Random Forest, Bagging, and AdaBoost are commonly used. Third, evaluation metrics such as area under the curve, accuracy, and F-measure are commonly used for performance evaluation. Finally, there is a clear preference for tools such as WEKA, MATLAB, and Python. By providing insights into current trends and practices in the field, this study offers valuable guidance to researchers and practitioners to make informed decisions to improve software fault prediction models and contribute to the overall improvement of software quality.

INDEX TERMS Machine learning, feature selection, software defect prediction, artificial intelligence.

I. INTRODUCTION

Feature selection (FS) methods are used to identify and retain the most relevant and informative data features for building accurate prediction models [1], [2], [3], [4], [5]. In literature, various terms such as attributes, metrics, and dimensions have been used for software features [6], [7]. The data contains numerous features, giving rise to noisy, irrelevant, and redundant information. As a result, the learning algorithm experiences an increase in both computation time and error rate [8], [9]. As the volume of data grows exponentially, the quality of data required for processing decreases gradually. This restriction is commonly referred to as the "Curse of Dimensionality", a concept introduced by Richard Bellman

The associate editor coordinating the review of this manuscript and approving it for publication was Walter Didimo.

in the 1960s and remains a topic of ongoing research [10], [11], [12]. Feature selection is a crucial step in data pre-processing, addressing the challenge of high-dimensional datasets [13]. This challenge involves two aspects: irrelevant variables, where certain features (independent variables) do not impact the target features (dependent variables), and redundant variables, where independent variables exhibit high correlation and can be removed [14].

In recent years, feature selection methods have gained more prominence due to their capability to enhance prediction accuracy and reduce model creation time [15], [16], [17], [18], [19]. These methods find applications in various domains, including healthcare and medicine, fraud detection, sentiment analysis, etc. [20], [21], [22], [23]. Four main categories of feature selection methods exist filter, wrapper [24], [25], embedded, and hybrid [10], [26]. Brief description of each fatire selection method is gioven in the following.

Filter Method: The Filter method, an open-loop approach, is one of the earliest feature selection methods. It assesses features based on their intrinsic characteristics before the learning tasks. This method relies on four measurement criteria: information, dependency, consistency, and distance. It performs feature selection independent of the machine-learning algorithm and utilizes statistical standards to rank the feature subsets [27], [28], [29]. Examples include Information Gain, Chi-square test, and Correlation Coefficient etc.

Wrapper Method: The Wrapper method, also known as a closed-loop approach, encapsulates the feature selection process around the learning algorithm and utilizes the accuracy or error rate of the classification process to evaluate features. Its goal is to select the most discriminating feature subset by reducing a specific classifier's estimation error. The wrapper method performs feature selection based on the learning algorithm's performance, ultimately choosing the most optimal features for the prediction process. Due to their multivariate nature, most wrapper methods require extensive computation times to achieve convergence, making them impractical for large datasets. Examples include forward selection, backward elimination, and recursive elimination [30], [31].

Embedded Method: The Embedded method is a built-in feature selection mechanism that integrates feature selection directly into the learning algorithm, leveraging its properties to guide feature evaluation. Compared to the wrapper method, the embedded approach is more efficient computationally while maintaining similar performance levels [32]. By combining the qualities of filter and wrapper methods, the embedded method selects features during the implementation of the algorithm, resulting in reduced computational complexity [33]. This is achieved by avoiding repeated executions of the classifier and the examination of every feature subset. Examples include LASSO and regularization tree-based methods.

Hybrid Method: Recent advancements in feature selection have led to the development of a hybrid method. This approach can be formulated by combining two distinct methods, for example, integrating both wrapper and filter techniques with identical criteria or feature selection methodologies. By adopting a hybrid method, the benefits of both methods can be achieved, leveraging their complementary strengths to enhance the feature selection process [34]. Figure 1 illustrates the hierarchical structure of feature selection methods.

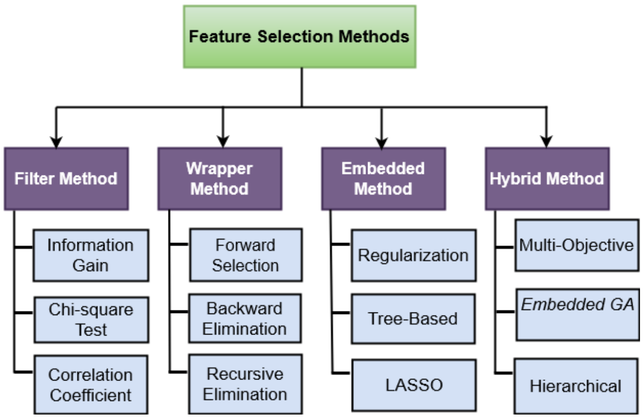


FIGURE 1. Feature selection methods.

Software defect prediction (SDP) uses historical data and machine learning techniques to predict and identify potential defects in software systems during the development phase. The main objective of software defect prediction is to proactively identify areas of code that are likely to contain flaws, allowing the software quality assurance team to focus their testing and debugging efforts on these high-risk areas [35], [36], [37]. Over the past decade, several researchers have presented empirical evidence that incorporating the feature selection method in the pre-processing step leads to improved classification accuracy in the context of software defect prediction [38], [39], [40]. A systematic literature review (SLR) is a structured research process to systematically identify, evaluate, and synthesize existing literature on a specific research topic or question. It aims to provide a comprehensive and unbiased overview of the available evidence related to the chosen topic.

Considering the importance of feature selection in predicting software failures, it's worth noting that a previous systematic review has provided valuable insights from the last decade of research [41]. This earlier review, which examined 15 primary studies from 2007 to 2017, laid the groundwork for exploring the role of feature selection methods, but left room for further investigation. This SLR attempts to extend this research to a broader and more up-to-date spectrum. By considering a selection of 49 primary studies from reputable repositories from 2014 to 2023, including IEEE Xplore, Science Direct, ACM Digital Library and Springer Link, this SLR takes a comprehensive look at the latest advances in the field. It aims to review and update the current state of feature selection techniques in software bug prediction, provide new insights into their application and relative performance, and thus expand the body of knowledge.

A. OBJECTIVE OF THE STUDY

The main objective of this SLR is to conduct a thorough and comprehensive investigation and analysis of the use of feature selection methods in the context of software failure prediction. The study aims to identify the predominant types of feature selection techniques implemented, the classification approaches used, the performance evaluation measures used, the datasets used, and the commonly used tools for executing feature selection methods. A thorough review of primary studies will provide a detailed and up-to-date understanding of the current landscape of software fault prediction methods.

B. MOTIVATION OF THE STUDY

This study is motivated by the central role of predicting software defects in improving software quality. Early detection of potential defects during development can significantly reduce the cost and effort of fixing software problems after deployment. Feature selection methods are recognized as key to the accuracy and efficiency of fault prediction models as they identify relevant features and reduce the dimensionality of the data. The motivation behind this SLR is to provide researchers and practitioners with insights into the latest trends and best practices in feature selection methods for predicting software defects. By answering key research questions, this study aims to enable stakeholders to make informed decisions to improve the effectiveness of software fault prediction models, ultimately leading to higher software quality and reliability. The study focuses on papers published since 2014 and extracts relevant literature from four well-known online search libraries: ACM, IEEE Xplore, Science Direct and Springer Link. Initially, 13,123 articles were found, and through a rigorous systematic research process, 49 relevant articles were selected as primary studies (PS).

C. CONTRIBUTION OF THE STUDY

We have identified the role of Feature Selection methods for software defect prediction and the role of the classification approach (i.e. individual, ensemble) for software defect prediction through feature selection. Identified performance evaluation of software defect prediction models employing feature selection methods. Identified datasets which have been selected for the implementation of feature selection methods.

D. ORGANIZATION OF THE STUDY

The remaining sections of the paper follow this structure: Section II delineates the research methodology, Section III presents the review's findings, and lastly, Section IV concludes the article while providing suggestions for future work. The step-by-step process to conduct the SLR is shown in Figure 2. It comprises three primary phases, with each step further subdivided into multiple sub-phases. List of abbreviations is given in Table 1.

II. RESEARCH PROTOCOL

A research protocol refers to a comprehensive plan and set of guidelines that outline the entire process of conducting the review. It provides a detailed framework for conducting the review, including the research questions to be addressed, the search strategy for identifying relevant studies, the inclusion and exclusion criteria for study selection, the methods for data extraction and synthesis, and the criteria for assessing the quality of the selected studies [42], [43], [44].

A. PHASE 1: PLANNING THE REVIEW

This Phase lays the foundation for the systematic and structured approach of the SLR. During this Phase, comprehensive guidelines are established to select the primary studies. By carefully planning the review, researchers can ensure that the review process is well-structured, systematic, and methodologically sound, leading to reliable and valuable insights [33]. The key aspects of this Phase are defined below.

1) IDENTIFICATION OF THE RESEARCH QUESTIONS

The main objective of this study is to conduct a systematic literature review (SLR) that identifies, analyzes, and summarizes empirical evidence related to the usage of FS methods for software defect prediction. The study focuses on datasets employed, FS methods, classification approaches, evaluation measures, and machine learning tools. To achieve this goal, the research questions and the motivation behind each question have been formulated to guide the review process, as shown in Table 2.

2) SELECTION OF DATA SOURCES

Data sources are the libraries or repositories from where the research studies should be retrieved. Four digital libraries have been chosen to extract the primary analyses, namely IEEE Xplore, Science Direct, ACM Digital Library, and Springer Link. The full text of the documents is searched to identify the primary studies. There are various options available to search each digital library for pertinent information. To find the most relevant literature, the search strategy is modified to satisfy the needs of the respective data source. The chosen data sources and the count of studies yielded by the search queries are shown in Table 3.

3) FORMULATION OF THE SEARCH STRING

A search string is a carefully crafted combination of keywords and search operators used to identify relevant studies that address the research question or topic of the review. This step focuses on specific keywords and their synonyms, drawn from the identified research questions in Table 3, to construct the search string. These keywords are combined using the 'AND' and 'OR' conditions in the specified sequence to formulate the ensuing search string:

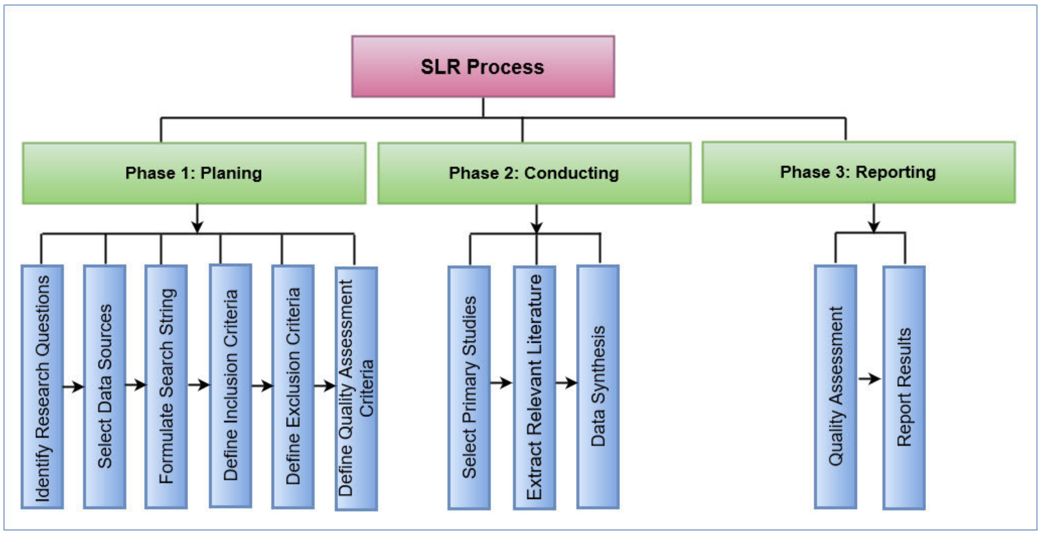


FIGURE 2. Systematic literature review process.

TABLE 1. List of abbreviations.

Description	Abbreviations	Description	Abbreviations	Description	Abbreviations
Software Defect Prediction	SDP	Logistic Regression	LR	Local Density of Features	LDF
Feature Selection	FS	Maximal Information Coefficient	MIC	Feature-Class Relevance	FCR
Systematic Literature Review	SLR	Support Vector Machine	SVM	Locally Weighted Learning	LWL
Primary Studies	PS	Histogram Intersection	HI	Similarity Measure	SM
Correlation-based FS	CFS	ANalysis Of Variance	ANOVA	Area Under the Curve	AUC
Average Probability Ensemble	APE	Hierarchical Agglomerative Clustering	HAC	Maximum Relevance	MR
Greedy Forward Selection	GFS	Cross-Project Defect Prediction	CPDP	Artificial Neural Network	ANN
Random Forest	RF	Similarity of Feature Distributions	SFD	Particle Swarm Optimization	PSO
Layered Recurrent Neural Network	L-RNN	Naïve Bayes	NB	Filter-based Feature Selection	FFS
Deep Neural Network	DNN	Best First Search	BFS	Whale Optimization Algorithm	WOA
Firefly Algorithm	FA	Greedy Step-wise Search	GSS	Binary Whale Optimization Algorithm	BWOA
Bernoulli Naive Bayes	BNB	Logistic Regression	LR	Convolutional Neural Network	CNN
Gradient Boosting	GB	Multinomial Naïve Bayes	MNB	Matthews Correlation Coefficient	MCC
Stochastic Gradient Descent	SGD	Least Square Support Vector Machine	LSSVM	National Aeronautics and Space Administration	NASA

TABLE 2. Research questions and motivation.

Research Questions	Motivation
RQ1: Which feature selection methods are implemented for software defect prediction?	Identify the range of feature selection methods and understand the current practices and trends.
RQ2: Which classification approach (i.e. individual, ensemble) is implemented for software defect prediction through feature selection?	Investigate the prevailing patterns in employing the classification approach, either used individually or in the ensemble.
RQ3: Which measures are implemented for the performance evaluation of software defect prediction models employing feature selection methods?	Examine the evaluation measures for assessing the real-world reliability of software defect prediction models.
RQ4: Which software defect prediction datasets have been selected to implement feature selection methods?	Explore the datasets chosen for applying feature selection methods regarding software defect prediction.

((“software”' OR “program” OR “system”) AND (“bug” OR “defect” OR “error” OR “fault”) AND (“prediction” OR “estimation” OR “interpretation” OR “classification”) AND (“features” OR “attributes” OR “dimensions” OR “metrics”) AND (“selection”, “reduction”, “engineering”, “extraction”, “elimination”)

TABLE 3. Query results from data sources.

Data source	Date searched	Results obtained	Final selection
IEEE Xplore	31/07/2023	1199	23
Science Direct	31/07/2023	3303	4
ACM Digital Library	31/07/2023	5076	4
Springer Link	31/07/2023	3545	18

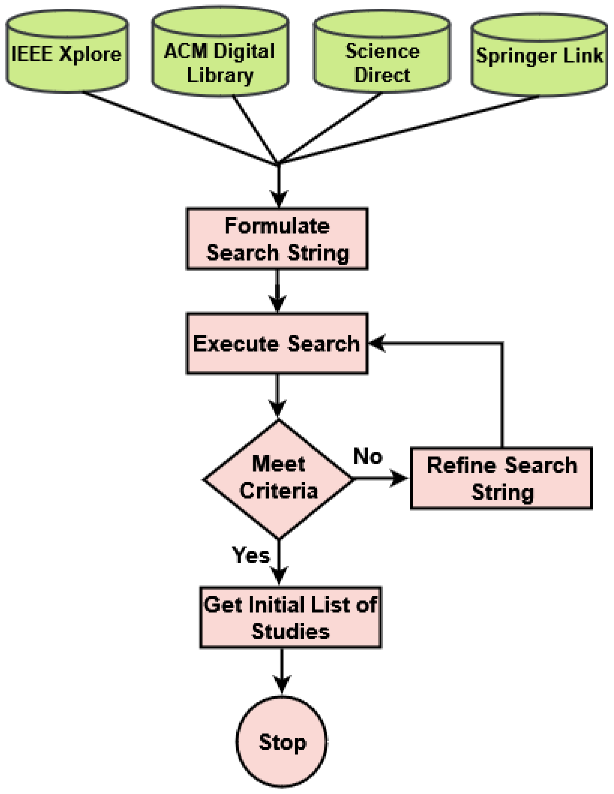


FIGURE 3. Search string formulation process.

TABLE 4. Search string formulation.

Keyword	Synonym/Alternative word
Software	("program" OR "system")
Defect	("bug" OR "error" OR "fault")
Prediction	("estimation" OR "interpretation" OR "classification")
Features	("attributes" OR "dimensions" OR "metrics")
Selection	("reduction", "engineering", "extraction", "elimination")
Technique	("algorithm" OR "classifier" OR "method' OR "model" OR "framework" OR "approach")

AND ("Technique" OR "algorithm" OR "classifier" OR "method' OR "model" OR "framework" OR "approach")). The process of formulating the search string is presented in Figure 3 whereas, search string formulation keyword and relevant alternative words are shown in Table 4.

4) DEFINING THE INCLUSION CRITERIA

Inclusion criteria in an SLR refer to the predefined rules used to determine which studies will be included in the review. In this review, the following inclusion criteria will be considered:

- Studies must have been published in the English language from 2014 to 2023
- The subject of the study should be focused on feature selection methods utilized in the domain of software defect prediction
- Selected studies must involve empirical research, conducting practical experiments on specific datasets
- The experiments conducted within the study should pertain to the classification of software defects using feature selection methods
- Each chosen study must comprehensively evaluate the performance of the applied feature selection methods
- The scope of chosen articles should be confined to publications in reputable journals, conferences, or books

TABLE 5. Quality assessment criteria.

Sr. No	Quality assessment questions
C1	Are the feature selection methods clearly described in the selected study?
C2	Does the study specify the classification approach (individual, ensemble)?
C3	Does the study include the measures used for performance evaluation?
C4	Are the datasets used for applying feature selection methods in software defect prediction identified?
C5	Does the study specify the details of the tool used?

5) DEFINING THE EXCLUSION CRITERIA

Exclusion criteria in an SLR refer to pre-designed conditions to determine which studies will be excluded from the review. The following categories of studies have been designated for exclusion:

- Those published before 2014
- Those whose primary focus is not on feature selection methods and their application in SDP
- Studies that lack empirical analysis results
- Studies that fail to evaluate the performance of executed feature selection methods

6) DEFINING THE QUALITY ASSESSMENT CRITERIA

Quality assessment (QA) criteria in an SLR refer to the predefined standards or guidelines used to assess the included studies' quality, reliability, and validity. Defining quality assessment criteria aims to ensure that the selected primary studies offer sufficient details to analyze the identified research question effectively. In this step, a standard is defined against each research question. Each quality assessment criterion is denoted by C and its respective number, as shown in Table 5.

B. PHASE 2: CONDUCTING THE REVIEW

Comprehensive state-of-the-art literature review is carried i.e. detailed in the following.

TABLE 6. List of selected primary studies.

Reference	Year	Library	Type	Journal/Conference/Book Name
[1]	2014	IEEE Xplore	Conference	International Conference on Informatics, Electronics & Vision (ICIEV)
[34]	2014	IEEE Xplore	Conference	IEEE 38th Annual Computer Software and Applications Conference
[46]	2014	Science Direct	Journal	Information and Software Technology
[56]	2014	IEEE Xplore	Conference	International Conference on Progress in Informatics and Computing (PIC)
[35]	2015	IEEE Xplore	Conference	5th International Conference on IT Convergence and Security (ICITCS)
[36]	2015	IEEE Xplore	Conference	IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)
[38]	2015	ACM	Conference	ISEC '15: 8th India Software Engineering Conference
[4]	2016	IEEE Xplore	Conference	IEEE 23rd International Conference on Software Analysis, Evolution and Re-engineering (SANER)
[5]	2016	Springer Link	Conference	Intelligent Systems Technologies and Applications
[37]	2016	IEEE Xplore	Conference	IEEE Second International Conference on Multimedia Big Data (BigMM)
[39]	2016	ACM	Conference	ICC '16: International Conference on Internet of Things and Cloud Computing
[53]	2016	IEEE Xplore	Conference	6th International Conference - Cloud System and Big Data Engineering (Confluence)
[8]	2017	IEEE Xplore	Journal	IEEE Access
[27]	2017	Springer Link	Journal	Cluster Computing
[28]	2017	Springer Link	Journal	Frontiers of Information Technology & Electronic Engineering
[29]	2017	IEEE Xplore	Conference	IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)
[40]	2017	IEEE Xplore	Conference	5th International Conference on Cyber and IT Service Management (CITSM)
[41]	2017	IEEE Xplore	Conference	IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)
[42]	2017	IEEE Xplore	Conference	International Conference on New Trends in Computing Sciences (ICTCS)
[60]	2017	Springer Link	Book Section	Proceedings of the 6th International Conference on Soft Computing for Problem Solving
[44]	2018	ACM	Conference	ICAIP '18: the 2nd International Conference on Advances in Image Processing
[45]	2018	Science Direct	Journal	Expert Systems with Applications
[22]	2019	IEEE Xplore	Conference	IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)
[43]	2019	IEEE Xplore	Conference	11th International Conference on Knowledge and Systems Engineering (KSE)
[47]	2019	Science Direct	Journal	Journal of Systems and Software
[48]	2019	IEEE Xplore	Journal	IEEE Access
[49]	2019	Springer Link	Journal	Cluster Computing
[51]	2019	IEEE Xplore	Conference	5th International Conference on Science and Technology (ICST)
[62]	2019	Springer Link	Book Section	Information Systems Design and Intelligent Applications
[12]	2020	Springer Link	Book Section	Intelligent Algorithms in Software Engineering
[52]	2020	IEEE Xplore	Conference	7th International Conference on Dependable Systems and Their Applications (DSA)
[54]	2020	IEEE Xplore	Conference	5th International Conference on Communication and Electronics Systems (ICCES)
[9]	2021	Springer Link	Book Section	Advances in Cyber Security
[21]	2021	Springer Link	Journal	Neural Computing and Applications
[23]	2021	IEEE Xplore	Journal	Journal of Systems Engineering and Electronics
[50]	2021	ACM	Journal	Computational Intelligence and Neuroscience
[55]	2021	Springer Link	Journal	Neural Computing and Applications
[57]	2021	IEEE Xplore	Conference	11th International Conference on Cloud Computing, Data Science & Engineering
[58]	2021	Springer Link	Journal	Neural Computing and Applications
[59]	2021	Springer Link	Journal	Automated Software Engineering
[61]	2021	Science Direct	Journal	Journal of Systems and Software
[63]	2022	IEEE Xplore	Conference	9th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)
[64]	2022	Springer Link	Journal	Complex & Intelligent Systems
[10]	2023	Springer Link	Journal	Applied Intelligence
[19]	2023	IEEE Xplore	Journal	IEEE Access
[65]	2023	Springer Link	Journal	Applied Intelligence

TABLE 7. Quality assessment criteria.

Criteria	Responding grade	Distribution of grades	Criteria satisfaction rate
C1	{1,0.5,1} (Yes, Nominally, No)	49 studies = 1	100%
C2	{1,0.5,1} (Yes, Nominally, No)	46 studies = 1, 3 studies = 0	93%
C3	{1,0.5,1} (Yes, Nominally, No)	49 studies = 1	100%
C4	{1,0.5,1} (Yes, Nominally, No)	49 studies = 1	100%
C5	{1,0.5,1} (Yes, Nominally, No)	31 studies = 1, 18 studies = 0	63%

1) SELECTION OF PRIMARY STUDIES

Primary studies (PS) refer to the individual articles or book sections that directly address the research questions or topic of the review. In this review, the primary studies were selected using the Tollgate approach, a structured methodology consisting of five phases [47]. This approach was helpful for the careful selection of primary studies, taking into account the established quality criteria. The individual phases of the tollgate approach are outlined in Figure 4, with 'N' denoting the total number of primary studies in each Phase. The selected primary studies are presented in Table 6.

Out of the selected primary studies, 21 are journal articles, 23 are part of conference proceedings, and 5 belong to book chapters. Figure 5 illustrates the count of primary studies according to the publication type.

C. EXTRACTION OF RELEVANT LITERATURE

The literature extracted from each primary study comprises the following details: proposed/used feature selection methods and classification approach. (i. e., individual/ensemble), performance evaluation measures, defect datasets for experiments, and tools used to implement feature selection

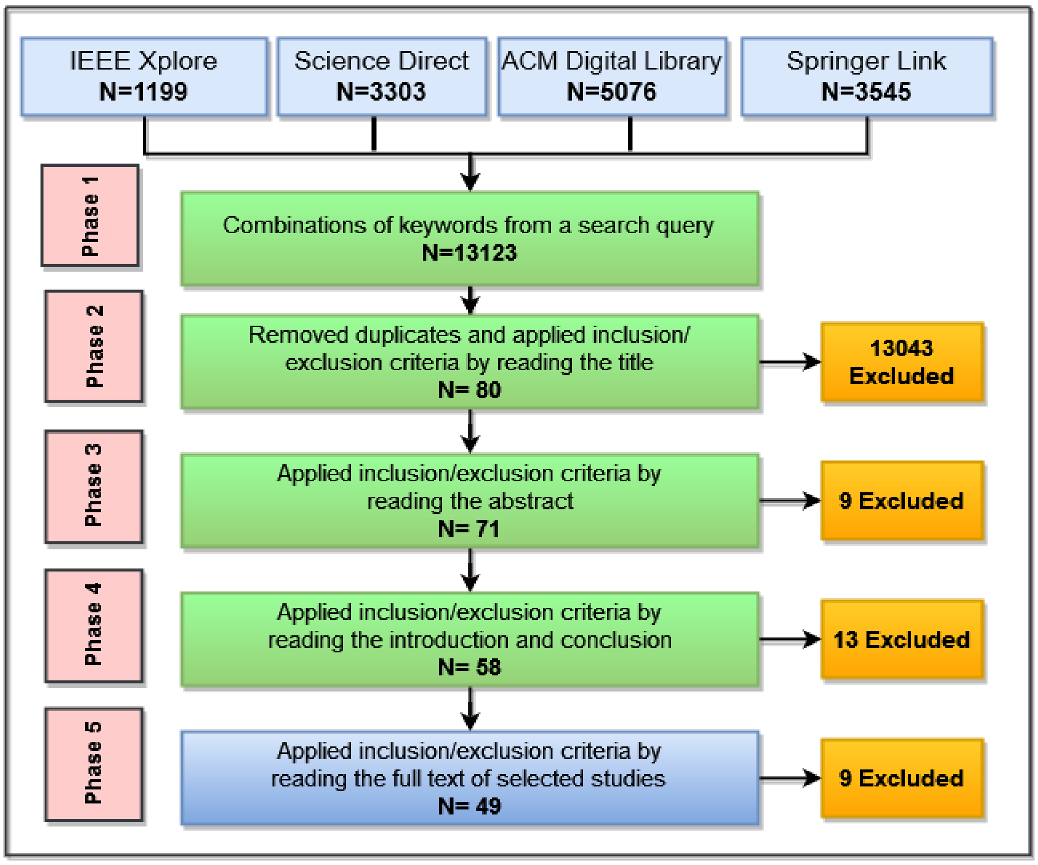

IEEE Xplore
N=1199
Science Direct
N=3303
ACM Digital Library
N=5076
Springer Link
N=3545
Phase 1
Combinations of keywords from a search query
N=13123
Phase 2
Removed duplicates and applied inclusion/ exclusion criteria by reading the title
N= 80
13043 Excluded
Phase 3
Applied inclusion/exclusion criteria by reading the abstract
N= 71
9 Excluded
Phase 4
Applied inclusion/exclusion criteria by reading the introduction and conclusion
N= 58
13 Excluded
Phase 5
Applied inclusion/exclusion criteria by reading the full text of selected studies
N= 49
9 Excluded

FIGURE 4. Phases of the tollgate approach for selection of primary studies.

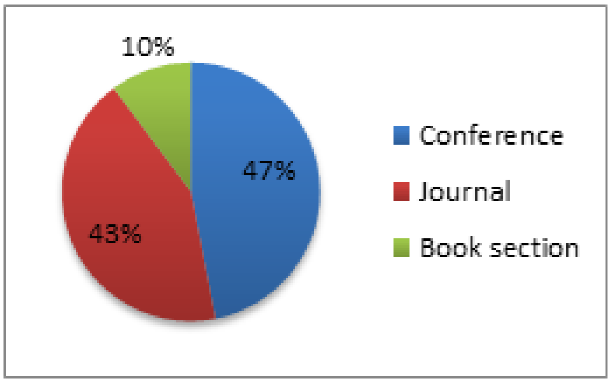


FIGURE 5. Distribution of primary studies by publication type.

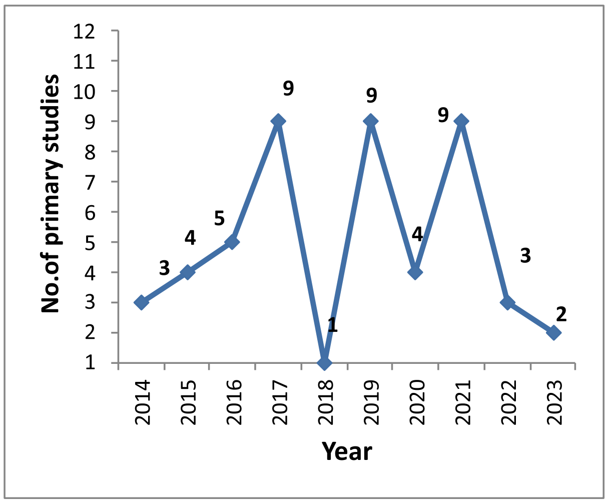


FIGURE 6. Distribution of primary studies across the years.

techniques. Figure 6 depicts the distribution of studies across years.

1) DATA SYNTHESIS

The data synthesis stage in an SLR refers to combining and analyzing the extracted data from the selected primary studies to draw meaningful conclusions and answer the research questions. During this stage, the collected data from various studies are systematically examined and synthesized to identify relationships among the findings.

D. PHASE 3: REPORTING THE REVIEW

1) QUALITY ASSESSMENT

Each primary study is assessed using the quality assessment criteria (C) shown in Table 7. Based on this assessment, each study is assigned a score between 0 and 1. In this scoring procedure, which is widely used in systematic literature reviews (SLRs), a score of 1 is assigned if the study explicitly meets the quality assessment criterion, 0.5 if it partially meets the standard, and 0 if it does not meet the criterion at all [42], [46]. To determine the final score for each study, the scores obtained for all quality assessment criteria are added together as shown in Table 7.

After assessing the quality of the chosen primary studies, it was noted that they achieved a score exceeding 90% for criteria C1, C2, C3, and C4. However, for C5, only 63% of the studies could answer the respective research question. This finding indicates that the selected primary studies offer sufficient information regarding feature selection techniques.

2) REPORT RESULTS

This is the last step of an SLR that addresses the identified research questions. The detailed answers to respective research questions from primary studies will be discussed in the next section.

III. RESULTS

This section will address the research questions outlined in Table 2.

A. RQ1: WHICH FEATURE SELECTION METHODS ARE IMPLEMENTED FOR SOFTWARE DEFECT PREDICTION?

Feature selection methods are used in machine learning and data analysis to select a subset of relevant features or variables from a larger set of available features. These methods aim to improve model performance, reduce overfitting and shorten training times by selecting the most informative and important features [9], [33]. Various FS methods have been proposed by researchers to improve the performance of the method under consideration [76]. In the following subsections, comprehensive overview of filter, wrapper, hybrid and embedded FS methods is provided, highlighting how each of these categories has been used in primary studies in the field of software fault prediction.

1) FILTERED FEATURE SELECTION METHOD

The researchers of [1], presented FS method comprising three algorithms. In the first algorithm, a specific subset of attributes was selected from the entire attribute pool with all possible pairwise combinations. After the combination, the attributes were reordered based on their frequency, with attributes with higher frequency being considered more important. The second algorithm helped to select the candidate attributes. Finally, the third algorithm determined the best attribute subset by testing all possible pairwise combinations and selecting the best one. The experimental results showed that data-aware FS performed better, even with a simple classifier. In [2] the researchers proposed a new FS framework based on feature clustering and feature ranking called FECAR. The framework consisted of two phases: Feature clustering and feature selection. In the feature clustering phase, a k-medoids clustering algorithm based on the feature-feature correlation (FF) measure was used to partition features into clusters. In the feature selection phase, relevant features from each cluster were selected based on FCR measures to create the final feature subset. In the study, symmetric uncertainty was used as FF correlation measure and information gain, chi-square and ReliefF were considered as FC relevance measures. The results showed that the FECAR framework effectively reduced the redundancy rate and improved the performance of the error predictors.

In [66], the researchers attempted to improve the accuracy of metric-based SDP by introducing the ReliefF-LC algorithm, which combined ReliefF feature selection and correlation analysis. This innovative approach effectively addressed the problem of redundant metrics and their negative impact on prediction accuracy. Experiments were conducted on two National Aeronautics and Space Administration (NASA) datasets, and the proposed algorithm was compared with two other FS methods. The experimental results showed the remarkable potential of the proposed algorithm to significantly improve the fault prediction performance. In [3], a network of software attributes demonstrating the presence of mutual information between features was presented. A clustering-based attribute selection method was proposed. The researchers used the maximum information coefficient (MIC) to determine attribute correlations within the network. This formed the basis for selecting subsets of features using methods such as spectral clustering, hierarchical clustering, k-means clustering and F-score based feature selection. Spectral clustering based feature selection showed better performance in SDP.

In [8], the authors proposed an optimized model for fault prediction using statistical process control and the FS method. In the FS method, the relevance of different attributes was analyzed using methods such as correlation and analysis of variance (ANOVA). Correlation assessed the dependency between attributes and their proximity, while ANOVA assessed attribute variance within and between classes. By applying these techniques, irrelevant or redundant attributes were identified and filtered out, resulting in an optimized data set for the SDP. In [50], the researchers proposed a new FS method using clusters of hybrid data for cross-project failure prediction (CPDP). The approach included two phases: a feature clustering phase using a density-based clustering method called DPC and a feature selection phase using three ranking strategies: local feature density (LDF), similarity of feature distributions (SFD) and feature class relevance (FCR). Experimental studies on real software projects showed that the proposed method performs better than the baseline methods and remains stable across different classifiers. Authors of [39] proposed a FS method based on a similarity measure (SM). The aim was to extract relevant features while taking into account the distributional differences of samples in different classes (defective or non-defective). The approach involved updating feature weights based on class-specific sample similarity, ranking features, and then sequentially evaluating feature subsets using the k-nearest neighbor (KNN) model with the area under the curve (AUC) metric for classification performance. The proposed method was compared with other FS approaches on eleven NASA datasets, demonstrating its effectiveness especially on large and imbalanced datasets.

Authors of [13], have proposed a novel FS method that uses clusters of hybrid data for cross-project failure prediction (CPDP). The approach included two phases: a feature clustering phase using a density-based clustering method called DPC and a feature selection phase using three ranking strategies: local feature density (LDF), feature distribution similarity (SFD) and feature class relevance (FCR). Experimental studies on real-world software projects have shown that the proposed method outperforms the baseline methods and remains stable across different classifiers. In [70], the researchers investigated different FS and extraction methods. The methods used correlation-based feature selection (CFS), principal component analysis (PCA) and kernel principal component analysis (KPCA). The results showed that SVM with CFS feature selection method provided the best prediction accuracy compared to PCA + SVM and KPCA + SVM. In [53], the researchers proposed a method that combined a genetic algorithm (GA) for feature optimization with a deep neural network (DNN) for classification. The GA was improved by a new chromosome design and a new technique to calculate the fitness function. The experimental results showed that the proposed method outperformed the existing techniques and achieved a remarkable classification accuracy. In [58], the researchers investigated feature subset selection and feature ranking approaches with the aim of improving the performance of CPD. Two FS methods were investigated: first, feature subset selection, in which a subset of the most relevant features was selected using methods such as Correlation-Based Feature Selection. The second method is feature ranking, in which the features are ranked according to their importance using techniques such as SM, correlation and gain ratio. These FS methods were applied to the initial projects to ensure consistent feature sets. The study found that both FS approaches improved CPDP accuracy, suggesting that selecting relevant features or ranking them can improve prediction across different projects.

In [61], researchers proposed a FS method using association rule mining (ARM). The process involved analyzing historical software data that included metrics related to software complexity, code size, and other pertinent attributes. The ARM method was then used to uncover patterns or rules that indicated correlations between certain metrics and the occurrence of errors. The results of the empirical evaluations showed that the proposed method combining NB with ARM-based feature selection yielded higher precision, recall, F-measure, and accuracy scores compared to conventional methods. In [73], the researchers proposed a FS method based on random selection. The aim was to identify a compact set of features that would allow effective prediction of defects and thus reduce the effort and cost of data collection for defect prediction models. The study found that the Random Subset Feature Selection (RSFS) algorithm outperformed other algorithms in terms of performance. In [5], the researchers presented a framework that enables supervised filter feature selection methods to incorporate feature set information from external knowledge sources. The framework was applied to improve the Minimum Redundancy Maximum Relevance (MRMR) algorithm, resulting in the MRMR grouping algorithm, which was designed to achieve higher classification accuracy by promoting the selection of features from different feature groups. The proposed framework showed significant accuracy improvements over the traditional MRMR algorithm and other commonly used filtering methods. In [62], the researchers investigated the interaction between FS methods and sampling methods and analyzed the impact on the accuracy of prediction models for SDP. They used three specific methods: chi-square, information gain and relief. These methods evaluated the relevance of each feature in the dataset. The study found that when using chi-square and information gain for FS, it's advantageous to first perform sampling for the AUC and recall methods. When using relief together with NB, LR and SVM, it's advantageous to perform FS first for AUC and recall improvements. The sequence of these three feature methods and sampling techniques also led to different results in terms of accuracy, precision and F1.

In [72], the researchers introduced the concept of multifilter FS based on rank aggregation. They used three different FS methods, including Chi-square, ReliefF and information gain, to improve the accuracy of SDP models. In this approach, individual ranked lists were created using different filtering methods and then aggregated using an aggregation function for the mean rank. The resulting aggregated list was further refined by a geometric mean function to select optimal features. The experimental results showed that the proposed method has improved performance. In [69], the researchers employed a FS method based on ANOVA F-value. This method was used to identify the best discriminative features for predicting faults in software modules. The proposed method was compared with current methods using precision, AUC and recall on 11 datasets. The application of this method helped to overcome the challenges of dealing with highly skewed data and to achieve more accurate and efficient automatic prediction of software faults. In [68], an improved CNN model called metric-based CNN (MB-CNN) was proposed. In particular, the chi-square FS method was implemented. The chi-square method evaluates the statistical dependency between each metric and the target variable, which in this case was the number of bugs in a software module. Metrics that showed a higher degree of dependency were considered more relevant and were selected for training the MB-CNN model. The results indicate that the proposed MB-CNN model is promising for fault prediction.

In [30], the researchers introduced a transformation and FS method called TFSCPDP. In the transformation phase, the features of the source project were used to transform the target project. In the feature selection phase, relevant features were selected based on distances to the transformed source. The results showed that TFSCPDP effectively reduced distributional differences and high-dimensional feature problems in CPDP and achieved a mean F-score of 0.8, the highest among existing approaches. In [65], the researchers proposed a model based on LASSO–SVM. In the preprocessing step, a FS method, compression and minimum absolute value selection, was implemented to reduce the dimensionality of the original dataset and eliminate data that were not relevant to the SDP. This technique contributed to a more focused and accurate analysis of software errors, ultimately leading to a more efficient and effective prediction process. In [74], the researchers employed the least square support vector machine (LSSVM) algorithm with ReliefF feature selection and the SMOTE method to solve problems with imbalanced classes in SDP datasets. The ReliefF FS method was implemented to address the challenge of identifying relevant attributes. The core principle of ReliefF was to evaluate whether a feature is meaningful by analyzing how its values differ within and between classes. If the values are similar within the same class but differ between classes, the attribute was considered valuable. The research showed the positive impact of the ReliefF method on classification performance.

2) HYBRID METHOD

In [56], the researchers investigated how FS and ensemble learning affect the accuracy of error prediction. They introduced the average probability ensemble (APE) method to classify errors. In the pre-processing phase, features were selected using Greedy Forward Selection (GFS) and Correlation-based FS (CFS). Seven base classifiers were used to create the APE model: random forest (RF), Gradient Boosting (GB), Stochastic Gradient Descent (SGD), Weighted Support Vector Machine, Logistic Regression (LR), Multinomial Naïve Bayes (MNB) and Bernoulli Naïve Bayes (BNB). The model was tested on six data sets and the results showed that GFS performed better than CFS. In [7], the researchers analyzed the performance of SDP models using FS methods, specifically focusing on whether a subset of features improves prediction accuracy. They conducted a comprehensive investigation of sixteen open-source projects with ten popular FS algorithms, including the Forward Greedy Algorithm, Backward Greedy Algorithm, Gini Index, Information Gain, Relief-based Algorithm, CFS, Logistic Regression-based Feature Selection (BLogReg), Ranker Algorithm, OneR Algorithm, and Wrapper Subset Algorithm. The experimental results showed that the FS methods, especially the forward and backward greedy algorithms, consistently delivered the best results across projects.

In [4], the researchers introduced a comprehensive attribute selection process consisting of five consecutive steps. First, they calculated the balance of each attribute using a base classifier and then ranked the attributes in a list based on their respective balance values. Then, pairwise combinations of all attributes were formed and the balance was calculated for each combination. These combinations were accurately filtered, selecting only the combinations with a high attribute balance to create a catalog of potential combinations after selecting the candidate attributes. The final step involved determining the best attribute set. This involved determining the set of attributes that provided the most favorable balance result for the SDP model. The proposed approach showed a performance improvement of up to 54% in predicting software faults. In [9], the researchers proposed a novel method using a feature space transformation process in conjunction with a normalization technique. The feature selection method involved transforming feature spaces and then classifying instances using a Support Vector Machine (SVM) with a Histogram Intersection (HI) kernel. The proposed multi-step process began by calculating the balance of each attribute using a base classifier and sorting accordingly. Potential attribute pairs were then generated, their balance calculated and those with a high balance selected. The eligible attributes were then selected based on their frequency and weighting, and the best attribute set was determined by iteratively optimizing the balance. With this approach, the most relevant attributes were systematically identified and retained, improving the accuracy of fault prediction. In [48], the researchers proposed a novel FS framework called MICHAC (Maximal Information Coefficient with Hierarchical Agglomerative Clustering), which consists of two main steps: using MIC to rank and eliminate irrelevant features and applying Hierarchical Agglomerative Clustering (HAC) to group and remove redundant features. Experiments on NASA and AEEEM datasets have shown the effectiveness of MICHAC in improving defect prediction across different classifiers and performance metrics.

In [63], the researchers developed a framework using FS based on five classifiers including IBk, KStar, Locally Weighted Learning (LWL), Random Tree (RT) and RF. The FS method used chi-square attribute evaluation and correlation attribute evaluation along with Best First, Greedy Stepwise and HAC to optimize attribute subsets and remove redundancy. The novel addition of MIC-valued attributes resulted in improved accuracy of the system. In [51], the researchers proposed a method that combines the Bat-based search algorithm (BA) for feature selection and the RF algorithm for prediction. The FS over BA aimed to eliminate uncorrelated features and retain meaningful features. The CFS algorithm was integrated into the feature selection process. CFS evaluates subsets of features based on their correlation with the class labels and their intercorrelation with each other. The selected subset of features was the result of the combination of BA and CFS. The proposed approach led to a more accurate identification of faulty software modules. In [49], the researchers combined an integrated sampling method with FS that effectively addresses the problems of class imbalance and attribute relevance in SDP. Three attribute selection algorithms were used: chi-square

(CS), information gain (IG), and relief (RLF). CS evaluated the relevance of attributes; IG evaluated the importance of attributes based on information gain, while RLF identified relevant attributes by comparing nearest neighbors of the same and different classes. The integration of the Synthetic Minority Oversampling Technique (SMOTE) with the Relief method was applied to the Naïve Bayes (NB) classifier. The results of the study showed that the integrated approach outperformed the other methods and achieved an improved prediction accuracy of 82%.

In [38], the researchers designed a parallel hybrid framework to efficiently process large software metrics datasets in the cloud. The FS method used was a hybrid approach that combined both filtering and wrapper methods. The Fisher filter and the Maximum Relevance (MR) filter were used to assign weights to the software metrics based on their relevance to fault prediction. These filters evaluated the discriminative power and correlation of each metric with the class labels. The metrics selected from these filtering methods were then integrated into an artificial neural network (ANN)-based wrapper that evaluated different subsets of metrics by training and testing an ANN model for fault prediction. This hybrid approach improved the accuracy of fault prediction by selecting important software metrics. In [14], researchers presented a hybrid method for selecting important software features to improve automated fault prediction. The proposed approach combined filter and wrapper methods. The filtering method evaluated the relevance of features without performance evaluation, while the wrapper method incorporated induction algorithms to evaluate the performance of selected subsets of features. The hybrid framework used two wrapper approaches ANNIGMA and SVM with MR filters. This fusion streamlined FS and training and reduced measurement effort.

In [54], the researchers presented a novel FS algorithm that operates in a two-step process within a hybrid framework of wrapper and filter. In the first step, features were grouped based on redundancy using a fast correlation-based filter grouping algorithm. In the second step, a subset of features was selected from each group using the particle swarm optimization (PSO) algorithm to ensure that at least one feature was selected from each group. The experiments performed with different classifiers on NASA and PROMISE datasets showed that the proposed method achieved up to 90% better results compared to using all features. In [33], the researchers presented a cluster-based hybrid feature selection (CHIFS) where relevant and non-redundant features are selected. They defined a spectral cluster-based feature quality coefficient (FQ) to evaluate the relevance and redundancy of the features. The method iteratively selected the final feature subset from the evaluated features based on the FQ. Experimental results showed that CHIFS outperformed other methods in terms of accuracy and efficiency, and it was particularly effective when based on the Pearson correlation coefficient. In [64], the researchers used a combination of FS methods to improve the process of predicting software faults. In the study, the FS method Simple Majority Voting was used, which incorporated the results of three different FS methods: Recursive Feature Elimination with Cross-Validation (RFECV), CFS and Select-k-Best FS. The final feature subset was determined by a majority vote between the three methods. The proposed approach contributed significantly to the prediction of software errors and thus improved the performance of the prediction models. In [60], the researchers proposed a new method, Rank Aggregation-Based Hybrid Multi-Filter Wrapper Feature Selection (RAHMFWFS), to overcome the challenges posed by the high dimensionality of software features in SDP models. The RAHMFWFS method was divided into two stages. The first stage involved a rank aggregation based multi-filter feature selection (RMFFS) that aggregated individual rankings from different filtering methods using a unique rank aggregation mechanism. In the second stage, an Enhanced Wrapper Feature Selection (EWFS) method was used, which was guided by a dynamic reordering strategy based on the aggregated ranked list. The proposed RAHMFWFS method was evaluated using NB and decision tree (DT) classifiers on benchmarking datasets for defects. The results showed the effectiveness of the proposed approach in solving problems related to filter rank selection and local stagnation within hybrid feature selection, leading to an improvement in the performance of SDP models.

In [34], the researchers proposed a FS algorithm known as ReliefF-based clustering (RFC). The RFC algorithm involved several steps, including initial relevance ranking, clustering based on symmetric uncertainty, and representative FS. The proposed algorithm was compared with classical FS methods on NASA SDP datasets using metrics such as AUC and F-score. The results showed that RFC improved the performance of SDP by effectively handling irrelevant features.

3) WRAPPER METHOD

In [6], the researchers integrated code metrics and process metrics as indicators. They investigated the SDP process by applying RF, Neural Networks (NN) and SVM algorithms using the Eclipse environment for binary classification. The selection of features was done by general linear model regression (GLM) to evaluate the importance of variables (VI) for each feature. Of the 61 features, the 20 most important features were captured based on the VI, which highlight the most important features for predicting errors and lead to improved accuracy. In [40], the researchers proposed a novel approach called MOFES (Multi-Objective Feature Selection) based on multi-objective optimization principles to strike a balance between reducing the number of selected features and optimizing the performance of the defect prediction model. The researchers concluded that MOFES effectively selects fewer features while achieving better prediction performance compared to the baseline methods. In [55], a feature selection approach was proposed by the researchers to improve the performance of a layered recurrent neural network (L-RNN) for SDP. The proposed method used three wrapper FS algorithms, including Binary Genetic Algorithm (BGA), Binary Particle Swarm Optimization (BPSO), and Binary Ant Colony Optimization (BACO). The approach was evaluated on 19 real software projects and compared with other methods such as NB, ANN, LR using AUC as a performance measure. The proposed approach showed better performance compared to existing methods. The proposed approach showed better performance compared to the current methods. In [52], the researchers proposed a model called GFsSDAEsTSE, which specifically deals with feature redundancy and class imbalance. The method integrates FS, deep learning through a stacked denoising autoencoder (SDAE) and ensemble learning. The JRC method was used for iterative FS, where features were added to the subset. The algorithm was stopped when adding more features did not significantly improve performance or when a tolerance threshold for variability was reached, resulting in improved performance of the SDP.

In [57], the researchers introduced a new method called MOFES (Multi-Objective Feature Selection), which selects relevant features. Two optimization goals were considered: Minimizing the number of selected features (in terms of cost) and maximizing the performance of the prediction models (in terms of utility). Pareto-based algorithms for multi-criteria optimization were used to solve this problem. The study analyzed different algorithms and compared MOFES with existing FS methods, demonstrating the effectiveness of the proposed method using real data sets. In [59] the researchers proposed a FS method based on the firefly algorithm (FA) to improve the accuracy of SDP. The natural blinking behavior of fireflies inspired this optimization technique, in which fireflies communicate through bioluminescence signals to efficiently search the cost function space. The approach considers both classification accuracy and size reduction as fitness objectives. The experiments showed that the potential of FA for FS in predicting software errors led to improved prediction accuracy and thus quality. In [25], researchers investigated two primary types of FS methods, including filter-based feature selection (FFS) and wrapper feature selection (WFS). Different search methods were explored, including Best First Search (BFS) and Greedy Step-wise Search (GSS).

The specific FS method used in the study was WFS, in which the classifier was used to evaluate the performance of each feature by measuring its interaction with the underlying classifier. This approach was compared with FFS methods in which the features derived from the features in the dataset were evaluated and ranked without the direct involvement of a classifier. The experimental results showed that metaheuristic methods within WFS outperformed the conventional BFS and GSS search methods. In [67], the researchers investigated the combination of SMOTE for data equalization and PSO for FS. The FS process involved representing each potential feature subset as a "particle" in a high-dimensional search space. The position of each particle was linked to a specific feature combination; its movement was based on its own historical best position and the global best position found by the entire swarm. This study contributed to the improvement of SDP models by enabling effective handling of imbalanced data and the selection of optimal features. In [32], the researchers investigated FS methods such as recursive feature elimination (RFE), correlation-based feature selection, Lasso, Ridge, ElasticNet, and Boruta. The proposed model combined Partial Least Square (PLS) regression and RFE for FS, which was additionally combined with SMOTE due to the unbalanced nature of the datasets used. The proposed approach achieved significant improvements in error prediction. In [71], the researchers presented an extended metaheuristic search-based FS algorithm called EMWS. This algorithm uses the whale optimization algorithm (WOA) and simulated annealing (SA) to effectively select a subset of relevant features. A unified defect prediction algorithm called WSHCKE was developed, which combined a hybrid DNN consisting of a convolutional neural network (CNN) and a kernel extreme learning machine (KELM), which integrated selected features into deep semantic features by CNN and improved the prediction by the classification capacity of KELM. The proposed algorithm was evaluated through extensive experiments in 20 software projects with different evaluation indicators. The results confirmed the effectiveness of EMWS and WSHCKE compared to existing methods.

In [75], the researchers proposed a new framework that integrates nested stacking and heterogeneous FS to achieve more accurate prediction of software defects. In the heterogeneous FS approach, different base models, including AdaBoost and RF, were used to perform FS independently. Each base model identified the most important features for its prediction. The features selected by these models were then considered for stacking in the nested-stacking classifier. The results of the experiments showed that the proposed nested-stacking system outperformed the baseline models. In [15], the researchers proposed a framework based on the Binary Whale Optimization Algorithm (BWOA) called SBEWOA. The BWOA was used as an FS method that utilized transfer functions to convert the original continuous WOA into a binary version suitable for FS. In addition, other optimization algorithms, in particular Gray Wolf Optimizer (GWO) and Harris Hawks Optimization (HHO), were integrated into the BWOA framework to improve its efficiency in navigating the feature space. This improved method, referred to as "SBEWOA"," showed promising results for SDP by effectively selecting informative features and minimizing irrelevant features.

4) EMBEDDED METHOD

In [16], researchers proposed a framework that investigated the joint effects of FS and data resampling on unbalanced two-class classification. The research compared two contrasting approaches, i.e., one in which FS was applied before resampling the data (FS+DS) and another in which FS was applied after resampling the data (DS+FS). The FS methods were individually selected from filter, wrapper and embedded methods. In the study, a comprehensive empirical analysis was conducted on fifty-two benchmark datasets with unbalanced distributions using different FS methods. The study concluded that FS is a mandatory step to improve the classification of unbalanced two-class datasets and should be integrated either before or after resampling the data.

Year wise distribution of each FS method is depicted in Figure 7 whereas, Figure 8 shows the FS method and the corresponding number of year-wise primary studies. Table 8 summarizes the feature selection methods deliberated in the chosen primary studies.

The choice of feature selection method depends on several factors, such as the size of the dataset, the computational resources available and the desired level of prediction accuracy. In this context, an analysis was conducted to highlight the strengths and limitations of different feature selection methods based on their prevalence in the selected primary studies. Table 9 gives an insight into the strengths and limitations of filter, hybrid and wrapper methods and provides a comprehensive overview of their utility in predicting software faults.

From 2014 to 2023, numerous feature selection methods were proposed. Researchers proposed several frameworks, including FECAR, MICHAC, FeSCH, MOFES, EMWS, CNN, and TFSCPDP, using filter, wrapper, hybrid, and embedded feature selection methods, leading to improved classification performance. In the primary studies, filter and hybrid FS methods were used most frequently, while wrapper and embedded methods were used less frequently.

B. RQ2: WHICH CLASSIFICATION APPROACH (I.E. INDIVIDUAL, ENSEMBLE) IS IMPLEMENTED FOR SOFTWARE DEFECT PREDICTION THROUGH FEATURE SELECTION?

Classifiers are used to categorize data into different classes. The main goal of a classifier is to learn a mapping from input features to predefined output classes, which enables the algorithm to predict the class of new unseen instances based on their features [78], [79]. Many researchers have implemented various single and ensemble classifiers to improve the prediction accuracy of the proposed models for SDP [31], [80], [81], [82]. In [2], researchers implemented two individual classifiers, NB and C4.5, to predict the probability of errors in new data. In [66], researchers selected three individual classifiers belonging to different families, including NB, Multilayer Perceptron (MLP), and SVM, to apply to reduced datasets with subgroup features. In [56], the researchers combined the predictive abilities of seven base classifiers consisting of individual and ensemble types. The approach included ensemble classifiers such as RF and GB as well as individual classifiers such as SGD, weighted SVMs (W-SVMs), LR, MNB and BNB. This integration was achieved using a voting ensemble approach, where the final decision was determined by averaging the probabilities generated by the underlying classifiers. In [3], [67], and [70], the researchers used SVM as an individual classifier to effectively train and develop the proposed model, thus utilizing the capabilities of SVM in the context of their study. In [7], the researchers implemented a mixture of individual and ensemble classifiers by considering NB and LR as individual classifiers and RF as an ensemble classifier to improve the predictive capabilities of the proposed model. In [4], [25], [40], and [61], researchers employed a conditional probability based classifier called NB to analyze the impact of the proposed FS method. In [9], researchers employed SVM with its HI kernel to evaluate the effectiveness of the proposed model. In [8], the researchers used two individual classifiers, DT and NB, to evaluate the proposed model.

In [48], the researchers used a combination of classifiers, including NB. They used Repeated Incremental Pruning to Produce Error Reduction (RIPPER) as an individual classifier and RF as an ensemble classifier to evaluate the performance of the proposed MIC with the Hierarchical Agglomerative Clustering (MICHAC) framework. Similarly, in [63], researchers employed a mixture of individual and base classifiers using Instance-Based k-nearest Neighbors (IBK), Kstar, LWL and RT as individual classifiers and RF as ensemble classifiers. In [6], researchers also employed a combination of classifiers that included RF (ensemble) along with the different capabilities of NB and neural network classifiers (individual) to investigate the impact of individual and ensemble algorithms on prediction accuracy. In [13] and [50], researchers used LR as an individual classifier to improve the prediction accuracy of CPDP models. In [51], researchers implemented an ensemble approach with an RF classifier to increase the performance of the proposed framework. In [39], [40], and [73], researchers employed KNN as an individual classifier to formulate the prediction model. In [14], researchers employed the prediction capabilities of two individual classifiers, including SVM and ANN, to improve the accuracy of the proposed framework. In [55], the researchers employed five individual classifiers including NB, ANN, LR, KNN, and C4.5 decision trees to compare the results obtained with L-RNN. In [52], the researchers used a two-stage ensemble learning with heterogeneous base classifiers, including Bagging, AdaBoost and RF. These base classifiers were combined with a voting ensemble method that considers weighted average probabilities as a combination rule. In [53], the researchers implemented DNN as an individual classifier to evaluate the effectiveness of the proposed hybrid approach. In this study a combination of classifiers was used, with KNN and DT as individual classifiers and RF as an ensemble classifier. In [57], four classifiers including J48, KNN, LR and NB were implemented to build the proposed model. In [58], two individual classifiers, namely KNN and NB, were implemented to improve the prediction capabilities of CPDP. In [59], three individual classifiers, including SVM, NB and KNN, were implemented to investigate the prediction

TABLE 8. Summary of FS methods.

year	Reference	Feature Selection Method
2014	[1]	Proposed an FS algorithm based on feature combinations, sorting, and selecting the best feature set
	[2]	Proposed an FS framework based on feature clustering and feature ranking called FECAR
	[38]	Proposed an FS method named relief-LC algorithm by seamlessly combining relief FS and Linear Correlation analysis
2015	[24]	Proposed an average probability ensemble (APE) method using greedy forward selection (GFS) and correlation-based FS methods
	[3]	Proposed a clustering-based FS method using maximal information coefficient
	[7]	Explored ten filter/wrapper-based FS methods on sixteen open-source projects.
	[4]	Proposed a comprehensive FS method calculating the balance, frequency and weight of each attribute
2016	[10]	Propose a sequential FS method by calculating and iteratively optimizing the balance of each attribute
	[5]	Proposed an optimized FS method using Correlation and ANOVA
	[4]	Proposed MICHAC (Maximal Information Coefficient with Hierarchical Agglomerative Clustering) framework, using MIC and Hierarchical Agglomerative Clustering (HAC) methods for FS
	[53]	Proposed a framework using chisquaredattributeeval and correlation attribute eval along with best first, greedy stepwise, and Hierarchical Agglomerative Clustering (HAC)
	[3]	Proposed FS method through General Linear Model (GLM) regression to assess variable importance (VI) for each feature
2017	[41]	Proposed an FS method using density-based clustering of hybrid data and ranking strategies
	[42]	Proposed a method for FS using bat-based search along with Correlation-based Feature Selection (CFS)
	[40]	Integrated the SMOTE sampling technique with the three feature selection algorithms, including Chi-Square (CS), Information Gain (IG), and Relief (RLF)
	[29]	Proposed a multi-objective feature selection method based on optimization principles
	[28]	Proposed a FS method based on a similarity measure (SM)
	[27]	Designed a parallel hybrid framework for FS that combined both filter and wrapper methods
	[8]	Proposed a hybrid method for FS using filter and wrapper methods
	[37]	Proposed FESCH (Feature Selection Using Clusters of Hybrid-Data) for Cross-Project Defect Prediction (CPDP)
	[60]	Proposed correlation-based FS method
2018	[44]	Proposed a hybrid FS method using wrapper and filter methods by implementing a fast correlation-based filter method
2019	[45]	Proposed an FS method using three wrapper algorithms, including Binary Genetic Algorithm (BGA), Binary Particle Swarm Optimization (BPSO), and Binary Ant Colony Optimization (BACO)
	[43]	Proposed greedy forward selection-based FS method
	[73]	Proposed genetic algorithm-based FS method
	[22]	Proposed a Cluster-based Hybrid FS method using spectral cluster-based Feature Quality coefficient
	[47]	Introduced a method called MOFES using Pareto-based multi-objective optimization algorithms
	[48]	Proposed an FS method using feature subset selection and feature ranking
	[51]	Proposed an FS method using association rule mining
	[49]	Proposed an FS method using the Firefly algorithm
	[62]	Proposed an FS method based on random subset selection
2020	[5]	Proposed a FS framework using Minimum Redundancy Maximum Relevance (MRMR) algorithm
	[12]	Proposed wrapper FS method based on meta-heuristic methods
	[2]	Proposed a majority voting-based FS method
	[75]	Analyzed the integration of various filter-based FS methods and sampling techniques
2021	[9]	Proposed a rank aggregation-based multi-filter FS method
	[50]	Proposed a Rank Aggregation-Based Hybrid Multi-Filter Wrapper FS (RAHMFWFS) method
	[41]	She proposed an FS method based on variance (ANOVA) F-value analysis.
	[61]	Proposed a meta-heuristic search-based FS algorithm called EMWS
	[58]	Proposed a Convolutional Neural Network (CNN) model named metrics-based CNN (MB-CNN) using the Chi-square FS method
	[57]	Analyzed the combination of SMOTE for balancing data and Particle Swarm Optimization algorithm for feature selection
	[21]	Proposed a FS model by combining Partial Least Square (PLS) Regression and Recursive Feature Elimination
	[55]	Proposed an FS method based on minimum absolute value compression and selection
	[23]	Proposed a hybrid FS algorithm relief-based clustering (RFC)
2022	63	Proposed a Least Square Support Vector Machine (LSSVM) algorithm with the ReliefF FS method
	[64]	Proposed a framework by integrating Nested-Stacking and heterogeneous FS methods using AdaBoost and random forest
	[19]	Proposed a hybrid technique using transformation and FS method named TFSCPDP
2023	[65]	Proposed a framework based on binary whale optimization FS algorithm
	[10]	Proposed a framework by integrating three feature selection methods, i.e. filter, wrapper, and embedded, along with sampling techniques

accuracy of the proposed method. In [64], several classifiers were used, including LR and SVM as individual classifiers and ensemble classifiers such as RF and XGBoost. In [34], [60], and [72], two individual classifiers, including DT and NB, were used to evaluate the effectiveness of the proposed method.

In [69], two classifiers were utilized. One was LR, employed as an individual classifier while other was a DT-based bagging classifier, used as an ensemble classifier, with the aim of enhancing the predictive capability of the proposed method. In [71], researchers employed two individual classifiers including DNN and kernel extreme learning machine to boost the predicting capabilities of proposed mode. In [68], researchers implemented artificial neural network-based classifier named CNN as an individual classifier to enhance the performance of propped method.

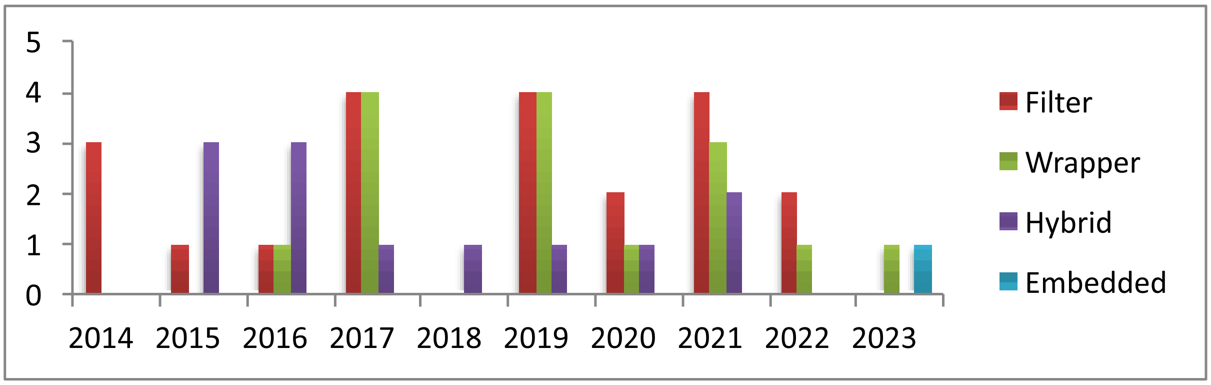


FIGURE 7. Year-wise distribution of FS methods.

TABLE 9. Strengths and limitations of FS methods.

FS methods	Strengths	Limitations
Filter Methods	Computational Efficiency: Filter methods are computationally efficient since they don't involve training a learning algorithm. Interpretability: They provide results that are easy to interpret and understand Scalability: Filter methods are suitable for large datasets and handle a large number of features.	Limited to Univariate Analysis: They only consider individual features without accounting for feature interactions Lower Predictive Accuracy: Filter methods may not yield the highest predictive accuracy due to their simplicity. Limited Control: Filter methods offer limited control over the feature selection process, as they use predefined criteria that may not capture domain-specific nuances.
Hybrid Methods	Scalability: Filter methods are suitable for large datasets and handle a large number of features. Reduced Overfitting: They can mitigate overfitting concerns associated with wrapper methods.	Complex Implementation: Implementing hybrid methods can be more complex compared to filter methods and may require fine-tuning. Computational Cost: They may still involve computational cost due to the wrapper component.
Wrapper Methods	Better Predictive Accuracy: Wrapper methods consider feature subsets and aim for better predictive accuracy. Feature Interaction: They capture complex feature interactions. Tailored to Learning Algorithm: Wrapper methods can optimize feature selection for the chosen learning algorithm.	Learning algorithm overhead: Wrapper methods require running the learning algorithm multiple times, making them computationally expensive, especially for large datasets. Risk of Overfitting: Due to their close ties to the learning algorithm, there is a risk of overfitting. Algorithm Dependency: Wrapper methods are highly dependent on the chosen learning algorithm, which can limit their versatility. Some algorithms may not work well with certain wrapper techniques, restricting their applicability.
Embedded Methods	Integrated Feature Selection: Embedded methods are an integral part of the model-building process, ensuring that relevant features are selected. Better Control: They provide more control over feature selection compared to filter methods.	Learning Algorithm Specific: Embedded methods are closely tied to specific learning algorithms, limiting their versatility. Algorithm Dependent: Their availability depends on the chosen algorithm, which may constrain their use.

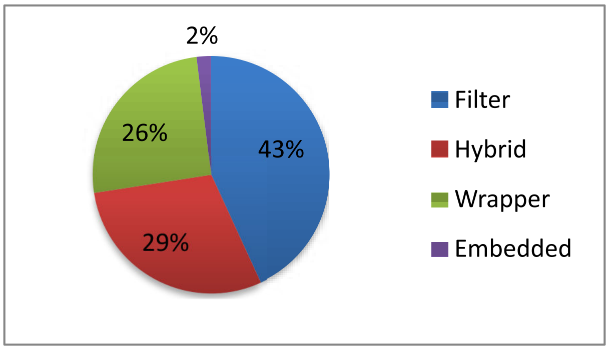


FIGURE 8. Distribution of FS methods across primary studies.

In [32], researchers analyzed the result of proposed model by combining several individual and ensemble classifiers. Individual classifies were LR and SVM while ensemble classifiers include AdaBoost, XGBoost, RF, GB, Stacking, and Extra Trees. In [65], LASSO-SVM classifier was implemented to access the performance of the proposed model. In [74], LSSVM was implemented to assess the predictive capabilities of a relief FS approach. In [75], The researchers utilized a range of classifiers to enhance classification performance through the Nested-Stacking Classifier approach. In the first layer, boosting algorithms including LightGBM, CatBoost, and AdaBoost were integrated, alongside a simple stacking model featuring MLP and RF. A Gradient Boosting DT served as the meta-classifier for this layer. The final layer employed a LR meta-classifier. In [30], a combination of classifiers was adopted including KNN, SVM, and LR as individual classifiers while employing RF as the ensemble classifier. In [15], seven classifiers were assessed, specifically

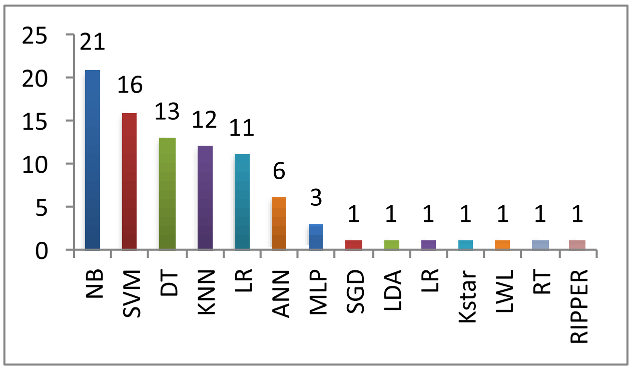


FIGURE 9. Distribution of primary studies over individual classifiers.

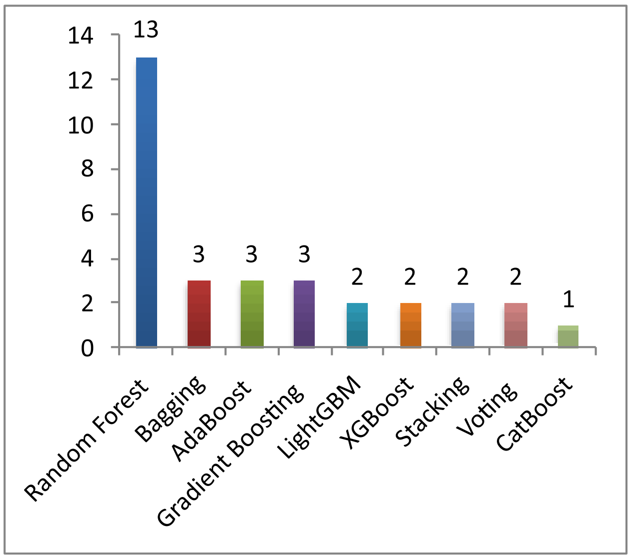


FIGURE 10. Distribution of primary studies over ensemble classifiers.

KNN, NB, Linear Discriminant Analysis (LDA), Linear Regression, DT, and SVM as individual classifiers while RF was employed as an ensemble classifier. In [16], three distinct classifiers from diverse families were utilized; including C4.5, SVM, and MLP aiming to evaluate the effectiveness of the proposed framework. A distribution of primary studies over individual classifiers is shown in Figure 9 and distribution of primary studies over ensemble classifiers is presented in Figure 10.

From 2014 to 2023, a range of classifiers were implemented in the selected primary studies, encompassing individual and ensemble classifiers. Individual classifiers that were widely implemented include NB, SVM, DT, KNN, and LR. On the other hand, the list of less commonly utilized individual classifiers encompassed ANN, MLP, SGD, LDA, LR, Kstar, LWL, RT, and RIPPER. In terms of ensemble classifiers, the more frequently employed ones consisted of RF, Bagging, AdaBoost, and GB. In contrast, the ensemble classifier techniques that saw less frequent use encompassed LightGBM, XGBoost, Stacking, Voting, and CatBoost.

C. RQ3: WHICH MEASURES ARE IMPLEMENTED FOR THE PERFORMANCE EVALUATION OF SOFTWARE DEFECT PREDICTION MODELS USING FEATURE SELECTION METHODS?

Many researchers have used several performance measures to evaluate how FS methods affect the predictive ability of SDP models [29]. These measures provide a quantitative way to assess how well a particular technique accomplishes its intended task. Using appropriate performance metrics, researchers and practitioners can make informed decisions about the feasibility and suitability of different techniques for specific tasks. In [3], [8], [14], [38], [70], [73], researchers evaluated the effectiveness of their proposed methods using accuracy measures. In [1], [4], [9], both the probability of detection (pd) and the likelihood of false alarm (pf) were used to evaluate performance. The fusion of these two metrics led to a further comprehensive performance measure, the so-called balance. In [2], [7], [33], [39], [40], [49], [54], [55], [56], [58], the AUC measure was included in the power analysis. In [5], the researchers used the Macro-F1 performance measure, which calculates the average of the F1 measure. In [6], the researchers used sensitivity, specificity and AUC to empirically investigate the performance of the classifiers using the Eclipse bug dataset.

In [13], [30], [48], and [66], the researchers investigated the results obtained by the classifiers using precision, recall, f-measure and AUC. In [50], the performance of the proposed FeSCH method was evaluated by using precision, recall and f-measure. In [25], [51], and [63], researchers employed precision and AUC measures to evaluate the performance of classifiers when applied to datasets from the TERA-PROMISE repository. In [52], researchers employed the F1 score, AUC, and Matthews correlation coefficient (MCC) to determine the performance of the proposed GFsSDAEsTSE method on twelve NASA datasets. In [53], the researchers investigated the effects of the hybrid approach for FS using precision, recall, specificity, F-score, and accuracy measures. In [57] and [75], the researchers employed the F1 score and AUC to indicate the performance of the proposed methods. In [59], the researchers accessed the interpretation of FA using accuracy, probability of detection, precision, false alarm probability, and effort. Effort was measured using the Lines of Code (LOC) metric. In [60], the researchers used accuracy, F-measure and AUC measures to evaluate the performance of the rank aggregation based multi-filter wrapper FS (RAHMFWFS) method. In [32], [61], and [65], the researchers employed accuracy, precision, recall, and F-measure to analyze the performance of the proposed FS method. In [62] and [64], the effectiveness of the proposed FS and sampling methods was evaluated using precision, recall, f-measure, accuracy, and AUC.

In [34], the predictive ability of the proposed FS algorithm was analyzed using precision, TPR, F-measure, FPR, and AUC. In [67], the researchers utilized recall, F1-measure, TPR, specificity, and FPR to evaluate the performance

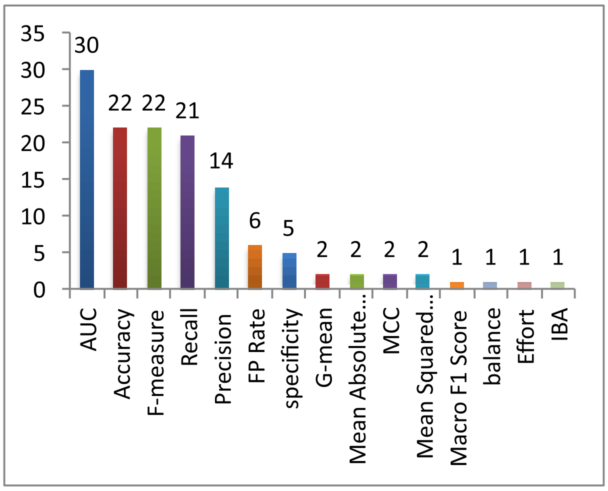


FIGURE 11. Distribution of primary studies across performance measures.

of the proposed method when implemented on datasets collected from Apache applications. In [67], the researchers used mean absolute error (MAE) and mean squared error (MSE) to evaluate the performance of the proposed CNN. In [69], the researchers used precision, AUC, recall, mean absolute error (MAE), root mean square error (RMSE), precision, and F-measure to analyze the performance of the proposed cost-sensitive approach. This evaluation was performed on eleven datasets from the PROMISE repository. In [71], the performance of the proposed hybrid DNN was investigated using F-1, AUC, G-measure and MCC. In [72], the researchers implemented accuracy, f-measure, recall and AUC for evaluating the proposed multi-filer feature selection method using NASA datasets. In [74], accuracy, f-measure, recall, and AUC were used to investigate the predictive ability of the Least Square Support Vector Machine (LSSVM) classifier. In [15], the researchers used TPR, TNR and AUC to investigate the efficiency of an ensemble learning based classification system. In [10], accuracy, TPR, TNR, G-mean, precision, F1 and Index of Balanced Accuracy (IBA) were used to investigate the impact of the Whale Optimizer based FS method on the proposed system. Figure 11 presents the performance measures alongside the corresponding count of primary studies.

Many performance measures were implemented during 2014-2023 to evaluate the performance of proposed techniques. Among the selected primary studies, AUC, accuracy, F-measure, recall, and precision are the most frequently used measures. In contrast, less frequently used measures include FP rate, specificity, G-mean, MAE, MCC, MSE, macro F1-score, balance, effort and IBA.

D. RQ4: WHICH SOFTWARE DEFECT PREDICTION DATASETS HAVE BEEN SELECTED FOR APPLYING FEATURE SELECTION TECHNIQUES?

Machine learning datasets are collections of data used to train, test and evaluate machine learning models. These datasets contain a variety of examples that are used to teach a machine learning classifier how to make predictions based on patterns and relationships within the data [1], [2], [3], [4], [5]. Researchers have used multiple datasets in their experiments and have found that different accuracy is achieved depending on the model proposed for each dataset. Therefore, most researchers have included multiple data sets in their studies [13], [54], [60].

The SDP datasets provided by NASA have gained much importance in research. These datasets include a set of static software metrics called features to determine the presence of bugs in modules. Each dataset that comes from the NASA repository corresponds to a NASA software system or subsystem and includes static code metrics that capture corresponding defect data for individual modules. Over the past decade, these datasets have been used extensively in [1], [4], [7], [9], [14], [34], [38], [39], [48], [49], [52], [56], [61], [63], [65], [70], [72], and [74]. Publicly available datasets from the PROMISE repository are used by the authors of [3], [8], [15], [33], [40], [51], [53], [55], [59], [66], [68], [69], and [73]. This repository contains datasets on object-oriented metrics, Halstead/McCabe metrics for procedural code, and some other static code metrics. In [2], the researchers implemented their proposed framework using real projects from the NASA and Eclipse repositories. In [5], the GRV dataset was used, which contains code quality metrics as features and comes from a JIRA software defect dataset. In [6], the researchers used the Eclipse dataset to investigate the process empirically. Eclipse datasets refer to data collections related to the integrated development environment (IDE) Eclipse, which supports various programming languages and is widely used for software development. It supports various programming languages and is widely used in software development.

In [13], the researchers used two datasets, which were from ReLink and AEEEM. The ReLink dataset was collected using the Understand toolkit. The AEEEM dataset, short for Appraisal, Estimation, and Emotion in Software Engineering and Multimedia datasets, comprises a curated and diverse collection of data specifically tailored for research and analysis in the fields of software engineering and multimedia studies. It included five different projects, namely EQ, JDT, LC, ML, and PDE, and contained 61 metrics. In [32], [54], [58], and [64], the researchers selected open-source datasets from NASA and PROMISE for empirical analysis. In [57], the researchers conducted experiments with the RELINK and PROMISE datasets, which are from real open-source projects. In [60], experiments were conducted with four publicly available datasets that originated from PROMISE, NASA, AEEEM, and ReLink. A collection of twenty-five datasets with different levels of granularity was selected for analysis.

In [62], two datasets from NASA and AEEEM were selected for empirical analysis. The NASA datasets consisted of five projects, including CM1, PC1, PC3, PC4, and MC2. The AEEEM datasets consisted of four projects, including

TABLE 10. Summary of commonly used software defect prediction datasets.

Dataset Name	Source	Characteristics	Domain	Max. Features
NASA	MDP Repository [1]	Wide range of software metrics and defect data	Space Exploration	38 [48]
ReLink	PROMISE repository [57]	Source code metrics, change history, defect info	Software Development	26 [13]
AEEEM	AEEEM corpora. [25]	Eclipse-specific metrics and defect information	Software Engineering	61 [50]
PROMISE	PROMISE repository [3]	Diverse software metrics and defect data	Empirical Software Engineering	20 [71]
Eclipse	PROMISE repository [2]	Metrics and defect data relevant to Eclipse	Open-Source Development	155 [2]

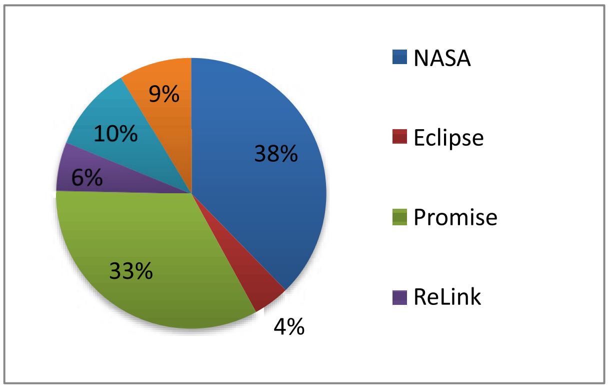


FIGURE 12. Distribution of datasets across primary studies.

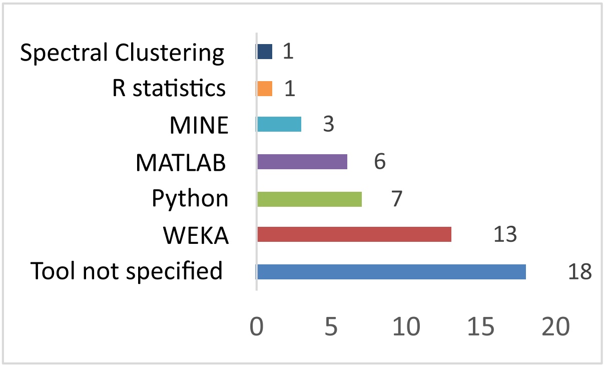


FIGURE 13. Distribution of primary studies across ML tools.

EQ, JDT, LC and PDE. In [25], datasets were collected from three different repositories: NASA, PROMISE and AEEEM. In [30] and [50], researchers used these datasets to prove the effectiveness of the proposed models. In [67], four datasets written in JAVA were used, which were obtained from Apache applications, including Apache Beam, Apache Hive, Apache Geode, and Apache Flink from Apache applications. Three datasets were used in [71], which are from PROMISE, ReLink and AEEEM. These datasets are easily accessible to the public and are often used as benchmark datasets. In [75], experiments were conducted using two software bug datasets, namely Kamei and PROMISE. In [16], the researchers used a collection of 52 datasets derived from real-world challenges, all of which are publicly available through the UCI and KEEL repositories. A distribution of datasets across primary studies is shown in Figure 12.

In order to provide a comprehensive overview of the data sets used to predict software failures, a summary table (Table 10) has been created containing essential details of the main data sets commonly used in this area. For each dataset, information is provided on the source, the features, the associated range and the maximum number of features available. This tabular information can be useful for researchers and practitioners conducting software failure prediction studies, as it provides a clear overview of the datasets under consideration.

E. RQ5: WHICH TOOLS ARE FREQUENTLY EMPLOYED TO EXECUTE FEATURE SELECTION IN THE CONTEXT OF SOFTWARE DEFECT PREDICTION?

Machine learning tools are essential resources that enable researchers, data scientists and developers to explore, analyze and model complex patterns in data. These tools include various software frameworks, libraries and platforms that facilitate the entire lifecycle of machine learning, from data preprocessing to model training and deployment [40], [49], [50]. They provide efficient implementations of algorithm visualization functions and often simplify the coding process, allowing practitioners to focus on extracting meaningful insights from the data instead of dealing with complicated technical details. In selected primary studies, researchers have used various machine learning tools to perform the proposed techniques. In [2], [4], [25], [34], [39], [40], [49], [51], [54], [58], [60], [66], and [72], researchers chose Waikato Environment for Knowledge Analysis (WEKA) for classifier performance evaluation.

In [3], [7], [38], [48], [53], [55], and [65], the researchers used MATLAB to analyze the predictive ability of the proposed methods. In [15], [30], [52], [56], [69], [74], and [75], the researchers executed the proposed method using Python libraries. In [13], [48], and [50], the researchers used the Maximum Information-based Nonparametric Exploration Toolkit (MINE) for empirical analysis. In [6], the researchers used the statistical tool R to empirically investigate the proposed approach. The researchers in [3] used the spectral clustering tool to identify and isolate error-prone modules. However, the machine learning tool was not explicitly specified in the studies [1], [5], [8], [9], [14], [16], [32], [32], [33], [59], [61], [62], [63], [64], [67], [70], [71], and [73]. A graphical representation of the distribution of primary studies over machine learning tools is shown in Figure 13.

Various machine learning tools have been chosen to analyze the performance of proposed approaches during 2014-2023. Among the selected primary studies, WEKA, MATLAB, and Python were the most frequently used ML tools. The less commonly used tools include R statistics tool and MINE.

IV. CONCLUSION AND FUTURE WORK

This systematic literature review provides valuable insights into the pivotal role of feature selection methods in software defect prediction based on the 49 primary studies selected from IEEE Xplore, Science Direct, ACM Digital Library, and Springer Link. It addresses key research questions, highlighting prevalent trends such as the prominence of filter and hybrid FS methods. It also examines the utilization of individual classifiers, such as NB, SVM, and DT. It also explores the application of ensemble classifiers like RF and bagging. Additionally, the review delves into the application of diverse performance evaluation metrics. The study also underscores the preferred tools, including WEKA, MATLAB, and Python. These findings offer a comprehensive understanding of the current trends in feature selection methods and lay the foundation for future research to improve the accuracy and efficiency of software defect prediction models. In the future, the impact of various feature selection methods on the performance of ensemble models should be investigated to enhance the effectiveness of software defect prediction models.

REFERENCES

[1] J. I. Khan, A. U. Gias, M. S. Siddik, M. H. Rahman, S. M. Khaled, and M. Shoyaib, "An attribute selection process for software defect prediction," in *Proc. Int. Conf. Inform. Electron. Vis. (ICIEV)*, 2014, pp. 1–4.

[2] S. Liu, X. Chen, W. Liu, J. Chen, Q. Gu, and D. Chen, "FECAR: A feature selection framework for software defect prediction," in *Proc. IEEE 38th Annu. Comput. Softw. Appl. Conf.*, Jul. 2014, pp. 426–435.

[3] S. Y. Kim, S. Gu, H.-H. Jeong, and K.-A. Sohn, "A network clustering based software attribute selection for identifying fault-prone modules," in *Proc. 5th Int. Conf. IT Converg. Secur. (ICITCS)*, Aug. 2015, pp. 1–5.

[4] P. Mandal and A. S. Ami, "Selecting best attributes for software defect prediction," in *Proc. IEEE Int. WIE Conf. Electr. Comput. Eng. (WIECON-ECE)*, Dec. 2015, pp. 110–113.

[5] K. Perera, J. Chan, and S. Karunasekera, "A framework for feature selection to exploit feature group structures," in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining*. Cham, Switzerland: Springer, 2020, pp. 792–804.

[6] W. Han, C.-H. Lung, and S. A. Ajila, "Empirical investigation of code and process metrics for defect prediction," in *Proc. IEEE 2nd Int. Conf. Multimedia Big Data (BigMM)*, Apr. 2016, pp. 436–439.

[7] K. Muthukumaran, A. Rallapalli, and N. L. B. Murthy, "Impact of feature selection techniques on bug prediction models," in *Proc. 8th India Softw. Eng. Conf.*, Feb. 2015, pp. 120–129.

[8] J. Nanditha, K. N. Sruthi, S. Ashok, and M. V. Judy, "Optimized defect prediction model using statistical process control and correlation-based feature selection method," in *Intelligent Systems Technologies and Applications*, vol. 1. Berlin, Germany: Springer, 2016, pp. 355–366.

[9] M. H. Rahman, S. Sharmin, S. M. Sarwar, and M. Shoyaib, "Software defect prediction using feature space transformation," in *Proc. Int. Conf. Internet Things Cloud Comput.*, Mar. 2016, pp. 1–6.

[10] B. Venkatesh and J. Anuradha, "A review of feature selection and its methods," *Cybern. Inf. Technol.*, vol. 19, no. 1, pp. 3–26, Mar. 2019.

[11] P. Grohs and L. Herrmann, "Deep neural network approximation for high-dimensional elliptic PDEs with boundary conditions," *IMA J. Numer. Anal.*, vol. 42, no. 3, pp. 2055–2082, Jul. 2022.

[12] R. Bellman and R. Kalaba, "Dynamic programming and statistical communication theory," *Proc. Nat. Acad. Sci. USA*, vol. 43, no. 8, pp. 749–751, Aug. 1957.

[13] C. Ni, W.-S. Liu, X. Chen, Q. Gu, D.-X. Chen, and Q.-G. Huang, "A cluster based feature selection method for cross-project software defect prediction," *J. Comput. Sci. Technol.*, vol. 32, no. 6, pp. 1090–1107, Nov. 2017.

[14] S. Huda, S. Alyahya, M. Mohsin Ali, S. Ahmad, J. Abawajy, H. Al-Dossari, and J. Yearwood, "A framework for software defect prediction and metric selection," *IEEE Access*, vol. 6, pp. 2844–2858, 2018.

[15] M. Mafarja, T. Thaher, M. A. Al-Betar, J. Too, M. A. Awadallah, I. A. Doush, and H. Turabieh, "Classification framework for faulty-software using enhanced exploratory whale optimizer-based feature selection scheme and random forest ensemble learning," *Int. J. Speech Technol.*, vol. 53, no. 15, pp. 18715–18757, Aug. 2023.

[16] C. Zhang, P. Soda, J. Bi, G. Fan, G. Almpanidis, S. García, and W. Ding, "An empirical study on the joint impact of feature selection and data resampling on imbalance classification," *Appl. Intell.*, vol. 53, no. 5, pp. 5449–5461, 2023.

[17] J. B. Awotunde, S. Misra, A. E. Adeniyi, M. K. Abiodun, M. Kaushik, and M. O. Lawrence, "A feature selection-based K-NN model for fast software defect prediction," in *Proc. Int. Conf. Comput. Sci. Appl.* Cham, Switzerland: Springer, 2022, pp. 49–61.

[18] S. C. Rathi, S. Misra, R. Colomo-Palacios, R. Adarsh, L. B. M. Neti, and L. Kumar, "Empirical evaluation of the performance of data sampling and feature selection techniques for software fault prediction," *Exp. Syst. Appl.*, vol. 223, Aug. 2023, Art. no. 119806.

[19] V. Mishra, S. Rath, and S. K. Rath, "Feature analysis for detection of breast cancer thermograms using dimensionality reduction techniques," in *Proc. Int. Health Inform. Conf.* Singapore: Springer, 2023, pp. 311–321.

[20] A. Madasu and S. Elango, "Efficient feature selection techniques for sentiment analysis," *Multimedia Tools Appl.*, vol. 79, nos. 9–10, pp. 6313–6335, Mar. 2020.

[21] K. L. Chiew, C. L. Tan, K. Wong, K. S. C. Yong, and W. K. Tiong, "A new hybrid ensemble feature selection framework for machine learning-based phishing detection system," *Inf. Sci.*, vol. 484, pp. 153–166, May 2019.

[22] P. Ghosh, S. Azam, M. Jonkman, A. Karim, F. M. J. M. Shamrat, E. Ignatious, S. Shultana, A. R. Beeravolu, and F. De Boer, "Efficient prediction of cardiovascular disease using machine learning algorithms with relief and LASSO feature selection techniques," *IEEE Access*, vol. 9, pp. 19304–19326, 2021.

[23] H. Zhao, Z. Liu, X. Yao, and Q. Yang, "A machine learning-based sentiment analysis of online product reviews with a novel term weighting and feature selection approach," *Inf. Process. Manag.*, vol. 58, no. 5, Sep. 2021, Art. no. 102656.

[24] Z. Xu, J. Liu, X. Luo, Z. Yang, Y. Zhang, P. Yuan, Y. Tang, and T. Zhang, "Software defect prediction based on kernel PCA and weighted extreme learning machine," *Inf. Softw. Technol.*, vol. 106, pp. 182–200, Feb. 2019.

[25] A. O. Balogun, S. Basri, S. A. Jadid, S. Mahamad, M. A. Al-momani, A. O. Bajeh, and A. K. Alazzawi, "Search-based wrapper feature selection methods in software defect prediction: An empirical analysis," in *Proc. Intell. Algorithms Softw. Eng., 9th Comput. Sci. Line Conf.*, vol. 19. Cham, Switzerland: Springer, 2020, pp. 492–503.

[26] R. Zebari, A. Abdulazeez, D. Zeebaree, D. Zebari, and J. Saeed, "A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction," *J. Appl. Sci. Technol. Trends*, vol. 1, no. 2, pp. 56–70, May 2020.

[27] P. Jindal and D. Kumar, "A review on dimensionality reduction techniques," *Int. J. Comput. Appl.*, vol. 173, no. 2, pp. 42–46, Sep. 2017.

[28] U. M. Khaire and R. Dhanalakshmi, "Stability of feature selection algorithm: A review," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 4, pp. 1060–1073, 2019.

[29] U. StaÅczyk and L. C. Jain, *Feature Selection for Data and Pattern Recognition: An Introduction*. Berlin, Germany: Springer, 2015, pp. 1–7.

[30] Y. Z. Bala, P. Abdul Samat, K. Y. Sharif, and N. Manshor, "Improving cross-project software defect prediction method through transformation and feature selection approach," *IEEE Access*, vol. 11, pp. 2318–2326, 2023.

[31] D. Q. Zeebaree, H. Haron, A. M. Abdulazeez, and D. A. Zebari, "Machine learning and region growing for breast cancer segmentation," in *Proc. Int. Conf. Adv. Sci. Eng. (ICOASE)*, Apr. 2019, pp. 88–93.

[32] S. Mehta and K. S. Patnaik, "Improved prediction of software defects using ensemble machine learning techniques," *Neural Comput. Appl.*, vol. 33, no. 16, pp. 10551–10562, Aug. 2021.

[33] F. Wang, J. Ai, and Z. Zou, "A cluster-based hybrid feature selection method for defect prediction," in *Proc. IEEE 19th Int. Conf. Softw. Qual., Rel. Secur. (QRS)*, Jul. 2019, pp. 1–9.

[34] X. Xiaolong, C. Wen, and W. Xinheng, "RFC: A feature selection algorithm for software defect prediction," *J. Syst. Eng. Electron.*, vol. 32, no. 2, pp. 389–398, Apr. 2021.

[35] M. J. Hernndez-Molinos, A. J. Snchez-Garca, R. E. Barrientos-Martnez, J. C. Pz-Arriaga, and J. O. Ocharn-Hernndez, "Software defect prediction with Bayesian approaches," *Mathematics*, vol. 11, no. 11, p. 2524, 2023.

[36] C. L. Prabha and N. Shivakumar, "Software defect prediction using machine learning techniques," in *Proc. 4th Int. Conf. Trends Electron. Informat. (ICOEI)*, Jun. 2020, pp. 728–733.

[37] L. Qiao, X. Li, Q. Umer, and P. Guo, "Deep learning based software defect prediction," *Neurocomputing*, vol. 385, pp. 100–110, Apr. 2020.

[38] M. M. Ali, S. Huda, J. Abawajy, S. Alyahya, H. Al-Dossari, and J. Yearwood, "A parallel framework for software defect detection and metric selection on cloud computing," *Cluster Comput.*, vol. 20, no. 3, pp. 2267–2281, Sep. 2017.

[39] Q. Yu, S.-J. Jiang, R.-C. Wang, and H.-Y. Wang, "A feature selection approach based on a similarity measure for software defect prediction," *Frontiers Inf. Technol. Electron. Eng.*, vol. 18, no. 11, pp. 1744–1753, Nov. 2017.

[40] X. Chen, Y. Shen, Z. Cui, and X. Ju, "Applying feature selection to software defect prediction using multi-objective optimization," in *Proc. IEEE 41st Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 2, Jul. 2017, pp. 54–59.

[41] H. Alsolai and M. Roper, "A systematic review of feature selection techniques in software quality prediction," in *Proc. Int. Conf. Electr. Comput. Technol. Appl. (ICECTA)*, Nov. 2019, pp. 1–5.

[42] S. Keele, "Guidelines for performing systematic literature reviews in software engineering," Software Eng. Group School Comput. Sci., Dept. Comput. Sci., Math. Keele Univ., Durham Univ., Durham, U.K., Keele, U.K., Tech. Rep. EBSE-2007-01, 2007.

[43] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering—A systematic literature review," *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 7–15, 2009.

[44] F. Matloob, T. M. Ghazal, N. Taleb, S. Aftab, M. Ahmad, M. A. Khan, S. Abbas, and T. R. Soomro, "Software defect prediction using ensemble learning: A systematic literature review," *IEEE Access*, vol. 9, pp. 98754–98771, 2021.

[45] H. Snyder, "Literature review as a research methodology: An overview and guidelines," *J. Bus. Res.*, vol. 104, pp. 333–339, Nov. 2019.

[46] C. Okoli, "A guide to conducting a standalone systematic literature review," *Commun. Assoc. Inf. Syst.*, vol. 37, pp. 879–910, 2015.

[47] W. Afzal, R. Torkar, and R. Feldt, "A systematic review of search-based testing for non-functional system properties," *Inf. Softw. Technol.*, vol. 51, no. 6, pp. 957–976, Jun. 2009.

[48] Z. Xu, J. Xuan, J. Liu, and X. Cui, "MICHAC: Defect prediction via feature selection based on maximal information coefficient with hierarchical agglomerative clustering," in *Proc. IEEE 23rd Int. Conf. Softw. Anal., Evol., Reengineering (SANER)*, vol. 1, Mar. 2016, pp. 370–381.

[49] S. A. Putri, "Combining integreted sampling technique with feature selection for software defect prediction," in *Proc. 5th Int. Conf. Cyber IT Service Manag. (CITSM)*, Aug. 2017, pp. 1–6.

[50] C. Ni, W. Liu, Q. Gu, X. Chen, and D. Chen, "FeSCH: A feature selection method using clusters of hybrid-data for cross-project defect prediction," in *Proc. IEEE 41st Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 1, Jul. 2017, pp. 51–56.

[51] D. R. Ibrahim, R. Ghnemat, and A. Hudaib, "Software defect prediction using feature selection and random forest algorithm," in *Proc. Int. Conf. New Trends Comput. Sci. (ICTCS)*, Oct. 2017, pp. 252–257.

[52] H. D. Tran, L. T. M. Hanh, and N. T. Binh, "Combining feature selection, feature learning and ensemble learning for software fault prediction," in *Proc. 11th Int. Conf. Knowl. Syst. Eng. (KSE)*, Oct. 2019, pp. 1–8.

[53] C. Manjula and L. Florence, "Deep neural network based hybrid approach for software defect prediction using software metrics," *Cluster Comput.*, vol. 22, no. S4, pp. 9847–9863, Jul. 2019.

[54] Y. Du, L. Zhang, J. Shi, J. Tang, and Y. Yin, "Feature-grouping-based two steps feature selection algorithm in software defect prediction," in *Proc. 2nd Int. Conf. Adv. Image Process.*, Jun. 2018, pp. 173–178.

[55] H. Turabieh, M. Mafarja, and X. Li, "Iterated feature selection algorithms with layered recurrent neural network for software fault prediction," *Exp. Syst. Appl.*, 122, pp. 27–42, May 2019.

[56] I. H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," *Inf. Softw. Technol.*, vol. 58, pp. 388–402, Feb. 2015.

[57] C. Ni, X. Chen, F. Wu, Y. Shen, and Q. Gu, "An empirical study on Pareto based multi-objective feature selection for software defect prediction," *J. Syst. Softw.*, vol. 152, pp. 215–238, Jun. 2019.

[58] Q. Yu, J. Qian, S. Jiang, Z. Wu, and G. Zhang, "An empirical study on the effectiveness of feature selection for cross-project defect prediction," *IEEE Access*, vol. 7, pp. 35710–35718, 2019.

[59] M. Anbu and G. S. Anandha Mala, "Feature selection using firefly algorithm in software defect prediction," *Cluster Comput.*, vol. 22, no. 5, pp. 10925–10934, Sep. 2019.

[60] A. O. Balogun, S. Basri, S. Mahamad, L. F. Capretz, A. A. Imam, M. A. Almomani, V. E. Adeyemo, and G. Kumar, "A novel rank aggregation-based hybrid multifilter wrapper feature selection method in software defect prediction," *Comput. Intell. Neurosci.*, vol. 2021, pp. 1–19, Nov. 2021.

[61] F. M. Tua and W. Danar Sunindyo, "Software defect prediction using software metrics with Naïve Bayes and rule mining association methods," in *Proc. 5th Int. Conf. Sci. Technol. (ICST)*, Jul. 2019, pp. 1–5.

[62] S. Fan, C. Liu, and Z. Li, "An empirical study on the impact of the interaction between feature selection and sampling in defect prediction," in *Proc. 7th Int. Conf. Dependable Syst. Their Appl. (DSA)*, Nov. 2020, pp. 131–140.

[63] M. Kakkar and S. Jain, "Feature selection in software defect prediction: A comparative study," in *Proc. 6th Int. Conf.-Cloud Syst. Big Data Eng.*, Jan. 2016, pp. 658–663.

[64] A. Joon, R. Kumar Tyagi, and K. Kumar, "Noise filtering and imbalance class distribution removal for optimizing software fault prediction using best software metrics suite," in *Proc. 5th Int. Conf. Commun. Electron. Syst. (ICCES)*, Jun. 2020, pp. 1381–1389.

[65] K. Wang, L. Liu, C. Yuan, and Z. Wang, "Software defect prediction model based on LASSO–SVM," *Neural Comput. Appl.*, vol. 33, no. 14, pp. 8249–8259, Jul. 2021.

[66] Y. Xia, G. Yan, X. Jiang, and Y. Yang, "A new metrics selection method for software defect prediction," in *Proc. IEEE Int. Conf. Prog. Informat. Comput.*, May 2014, pp. 433–436.

[67] R. Malhotra, N. Nishant, S. Gurha, and V. Rathi, "Application of particle swarm optimization for software defect prediction using object oriented metrics," in *Proc. 11th Int. Conf. Cloud Comput., Data Sci. Eng.*, Jan. 2021, pp. 88–93.

[68] M. Nevendra and P. Singh, "Defect count prediction via metric-based convolutional neural network," *Neural Comput. Appl.*, vol. 33, no. 22, pp. 15319–15344, 2021.

[69] A. Ali, N. Khan, M. Abu-Tair, J. Noppen, S. McClean, and I. McChesney, "Discriminating features-based cost-sensitive approach for software defect prediction," *Automated Softw. Eng.*, vol. 28, no. 2, pp. 1–18, Nov. 2021.

[70] R. Kumar and K. P. Singh, "SVM with feature selection and extraction techniques for defect-prone software module prediction," in *Proc. 6th Int. Conf. Soft Comput. Problem Solving*, vol. 2. Singapore: Springer, 2017, pp. 279–289.

[71] K. Zhu, S. Ying, N. Zhang, and D. Zhu, "Software defect prediction based on enhanced metaheuristic feature selection optimization and a hybrid deep neural network," *J. Syst. Softw.*, vol. 180, Oct. 2021, Art. no. 111026.

[72] A. O. Balogun, S. Basri, S. J. Abdulkadir, S. Mahamad, M. A. Al-momamni, A. A. Imam, and G. M. Kumar, "Rank aggregation based multi-filter feature selection method for software defect prediction," in *Advances in Cyber Security*. Singapore: Springer, 2021, pp. 371–383.

[73] G. N. V. R. Rao, V. V. S. S. Balaram, and B. Vishnuvardhan, "Attribute reduction for defect prediction using random subset feature selection method," in *Information Systems Design and Intelligent Applications*, vol. 1. Singapore: Springer, 2019, pp. 551–558.

[74] T. F. Husin and M. R. Pribadi, "Implementation of LSSVM in classification of software defect prediction data with feature selection," in *Proc. 9th Int. Conf. Electr. Eng., Comput. Sci. Informat. (EECSI)*, Oct. 2022, pp. 126–131.

[75] L.-Q. Chen, C. Wang, and S.-L. Song, "Software defect prediction based on nested-stacking and heterogeneous feature selection," *Complex Intell. Syst.*, vol. 8, no. 4, pp. 3333–3348, Aug. 2022.

[76] L. Kumar, S. K. Sripada, A. Sureka, and S. K. Rath, "Effective fault prediction model developed using least square support vector machine (LSSVM)," *J. Syst. Softw.*, vol. 137, pp. 686–712, Mar. 2018.
[77] H. Alsghaier and M. Akour, "Software fault prediction using particle swarm algorithm with genetic algorithm and support vector machine classifier," *Softw., Pract. Exper.*, vol. 50, no. 4, pp. 407–427, Apr. 2020.
[78] T. O. Olaleye, O. T. Arogundade, S. Misra, A. Abayomi-Alli, and U. Kose, "Predictive analytics and software defect severity: A systematic review and future directions," *Sci. Program.*, vol. 2023, pp. 1–18, Feb. 2023.
[79] R. Panigrahi, S. K. Kuanar, L. Kumar, N. Padhy, and S. C. Satapathy, "Software reusability metrics prediction and cost estimation by using machine learning algorithms," *Int. J. Knowl.-based Intell. Eng. Syst.*, vol. 23, no. 4, pp. 317–328, Feb. 2020.
[80] S. S. Rathore and S. Kumar, "An empirical study of some software fault prediction techniques for the number of faults prediction," *Soft Comput.*, vol. 21, no. 24, pp. 7417–7434, Dec. 2017.
[81] S. S. Rathore and S. Kumar, "A study on software fault prediction techniques," *Artif. Intell. Rev.*, vol. 51, no. 2, pp. 255–327, Feb. 2019.
[82] S. S. Rathore and S. Kumar, "Towards an ensemble based system for predicting the number of software faults," *Exp. Syst. Appl.*, vol. 82, pp. 357–382, Oct. 2017.

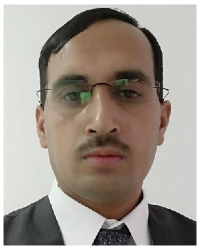
MISBAH ALI received the bachelor's degree in computer science from the Punjab University College of Information Technology (PUCIT), Lahore, Pakistan, and the master's degree in computer science from Virtual University of Pakistan. She has contributed significantly to the international software industry for three years by working with software houses, such as Systems Ltd., and TRG. She also has six years of teaching experience. Her research interests include machine learning, natural language processing, and software process improvement.

TEHSEEN MAZHAR received the B.Sc. degree in computer science from Bahaudin Zakaria University, Multan, Pakistan, the M.Sc. degree in computer science from Qauid-e-Azam University Islamabad, Pakistan, and the M.S.C.S. degree from the Virtual University of Pakistan, where he is currently pursuing the Ph.D. degree. He is also with SED and a Lecturer with GCUF. He has more than 21 publications in reputed journals, such as *Electronics*, *Health*, *Applied Sciences*, *Brain Sciences*, *Symmetry*, *Future Internet*, *PeerJ*, and *CMC*. His research interests include machine learning, the Internet of Things, and computer networks.

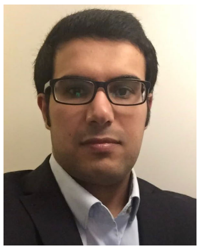
TARIQ SHAHZAD received the B.E. and M.S. degrees from COMSATS University Islamabad, Pakistan, in 2006 and 2014, respectively, and the Ph.D. degree from the University of Johannesburg, South Africa, in 2021. He is currently an Assistant Professor with COMSATS University Islamabad, Sahiwal Campus, Pakistan. His research work has published in top-tier IEEE conferences and well-reputed peer-reviewed journals. His research interests include the Internet of Things, machine learning, and AI in healthcare. He has also served as a technical program committee member and a paper reviewer for international conferences and journals.

YAZEED YASIN GHADI received the Ph.D. degree in electrical and computer engineering from Queensland University. He is currently an Assistant Professor of software engineering with Al Ain University. He was a Postdoctoral Researcher with Queensland University, before joining Al Ain University. His current research is on developing novel electro-acousto-optic neural interfaces for large scale high resolution electrophysiology and distributed optogenetic stimulation. He has published more than 80 peer-reviewed journal and conference papers and holds three pending patents. He was a recipient of several awards. His dissertation on developing novel hybrid plasmonicphotonic on chip biochemical sensors received the Sigma Xi Best Ph.D. Thesis Award.

SYED MUHAMMAD MOHSIN received the B.S. and M.S. degrees in computer science from the Virtual University of Pakistan, in 2011 and 2016, respectively. He is currently a Ph.D. Scholar with COMSATS University Islamabad, Pakistan. He has 14 years of experience in system/network administration, teaching, and research and development. He was a recipient of the Merit Scholarship from the Higher Education Commission (HEC), Pakistan, for his Ph.D. study. He has presented his research work in national and international conferences. He has published numerous research articles in conferences and journals. His research interests include smart grids, the Internet of Things, cloud computing, green energy, machine learning, cyber security, and related areas. He has been a TPC member and an invited reviewer of international conferences and journals.



SYED MUHAMMAD ABRAR AKBER received the B.S. and M.S. degrees in computer science from the Virtual University of Pakistan, in 2010 and 2015, respectively. He is currently pursuing the Ph.D. degree with the Silesian University of Technology, Gliwice, Poland. He has published numerous research articles in high level conferences and peer-reviewed journals. He has also presented his research work at prestigious national and international conferences. His research interests include artificial intelligence, big data processing, wireless networks, computer graphics, and related areas.



MOHAMMED ALI received the Ph.D. degree in computer science from Swansea University, U.K. He is currently an Assistant Professor with the Department of Computer Science, King Khalid University, Saudi Arabia. He is also the Vice Dean of the Applied College, King Khalid University. His research interests include data mining and knowledge discovery, artificial intelligence, information visualization, machine learning, and visual analytics.