Intelligent Automation & Soft Computing
DOI:10.32604/iasc.2021.017562
Review

Tech Science Press

Software Defect Prediction Using Supervised Machine Learning Techniques: A Systematic Literature Review

Faseeha Matloob[1], Shabib Aftab[1,2], Munir Ahmad[2], Muhammad Adnan Khan[3,*], Areej Fatima[4], Muhammad Iqbal[2], Wesam Mohsen Alruwaili[5] and Nouh Sabri Elmitwally[5,6]

[1]Department of Computer Science, Virtual University of Pakistan, Lahore, 54000, Pakistan
[2]School of Computer Science, National College of Business Administration and Economics, Lahore, 54000, Pakistan
[3]Riphah School of Computing and Innovation, Riphah International University, Lahore Campus, Lahore, 54000, Pakistan
[4]Department of Computer Science, Lahore Garrison university, Lahore, 54000, Pakistan
[5]College of Computer and Information Sciences, Jouf University, Sakaka, 72341, Saudi Arabia
[6]Department of Computer Science, Faculty of Computers and Artificial Intelligence, Cairo University, 12613, Egypt
[*]Corresponding Author: Muhammad Adnan Khan. Email: madnankhan@ncbae.edu.pk
Received: 03 February 2021; Accepted: 07 April 2021

Abstract: Software defect prediction (SDP) is the process of detecting defect-prone software modules before the testing stage. The testing stage in the software development life cycle is expensive and consumes the most resources of all the stages. SDP can minimize the cost of the testing stage, which can ultimately lead to the development of higher-quality software at a lower cost. With this approach, only those modules classified as defective are tested. Over the past two decades, many researchers have proposed methods and frameworks to improve the performance of the SDP process. The main research topics are association, estimation, clustering, classification, and dataset analysis. This study provides a systematic literature review that highlights the latest research trends in the area of SDP by providing a critical review of papers published between 2016 and 2019. Initially, 1012 papers were shortlisted from three online libraries (IEEE Xplore, ACM, and ScienceDirect); following a systematic research protocol, 22 of these papers were selected for detailed critical review. This review will serve researchers by providing the most current picture of the published work on software defect classification.

Keywords: Software defect prediction; systematic literature review; machine learning

1 Introduction

The development of higher-quality software at lower cost has always been a key objective of the software industry, as well as an area of focus by many researchers in the software engineering domain [1–7]. According to a study in 2018, the market for business software was $3.7 trillion [8], 23% of which was related to quality assurance (QA) and testing [9]. It is important to remove defects from software before delivery. However, as software grows in size and complexity, it becomes increasingly difficult to identify all the bugs [10]. A small bug in a critical system can lead to disaster. A notorious example

highlighting the importance of QA, testing, and removing defects was the 1999 loss of NASA's $125 million Mars Climate Orbiter due to a small data conversion issue [11]. Defects in software can be categorized as syntax errors, spelling errors, wrong program statements, and design or specification errors [12]. Testing plays a key role in the software development lifecycle (SDLC), and eliminates bugs while maintaining quality [1–7]. However, it has also been observed that software testing consumes more resources in SDLC than any other activity [1–7]. The process of software defect prediction (SDP) can be used as a QA activity to identify those modules that are more likely to be defective so that only those are tested. With this approach, higher-quality software can be developed at a lower cost [1–7].

Many researchers have focused on the domain of SDP. According to Wahono [13], the five areas receiving the most focus by researchers are: dataset analysis, association, estimation, clustering, and classification. Dataset analysis aims to investigate the issues that are used for the prediction of software defects. Many benchmark datasets are available, such as the NASA, MDP, and other data repositories. Data analysis also focuses on preprocessing techniques to make the data more suitable for prediction models, in order to achieve maximum accuracy. Association research uses association rule mining algorithms to detect the association among software defects in the software system. Estimation research employs statistical techniques, capture-recapture models, and neural networks to estimate the number of defects in a software system. Clustering techniques use clustering algorithms to detect groups or clusters of defects. This unsupervised technique from the machine learning domain is particularly used in situations where labels are unknown. Classification research is centered on determining whether a particular software module is defective by using historical data of software metrics. This process uses supervised classification algorithms [14–20]. Many classification frameworks have been proposed [2–6], and consist of various additional stages, such as data cleansing, feature selection, and ensemble creation (integration of multiple classifiers).

To train the prediction models, a defect dataset can be obtained from earlier releases of the same project [21] or from other projects [22]. After training, the test data is fed into these models for classification. This process is shown in Fig. 1. By using these prediction models, software engineers can use the data of previously developed and tested modules to predict whether the newly developed module is defective so that appropriate testing resources can be allocated to those modules only.

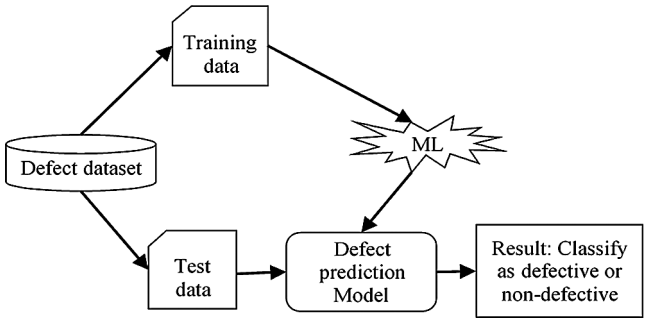

Figure 1: Software defect prediction process

In the past few years, there has been significant progress in defect prediction, with many research papers being published. Scholars have published excellent literature reviews in the domain of SDP as well [23]. Catal et al. [24] reviewed and classified 74 research studies with respect to methods, metrics, and datasets. Subsequently, Catal [25] reviewed 90 research papers between 1990 and 2009. He reviewed both machine learning-based and statistical-based approaches in SDP in software engineering. Wahono [13] reviewed research trends, datasets, methods, and frameworks from 2000 to 2013. He selected

71 primary studies for the reviews. Similarly, Li et al. [26] reviewed 70 representative defect prediction studies between 2014 and 2017. They summarized the research trends in ML algorithms, data manipulation, effort-aware prediction, and empirical studies.

This systematic literature review aims to reflect the progress made in the latest research on detecting defect-prone software modules. To extract the relevant research papers, a systematic research process was followed. Initially, 1012 studies published between 2016 and 2019 were extracted from three well-known online libraries: IEEE Xplore, ACM, and Science Direct (Tab. 1). Following a thorough systematic research process, the 22 most relevant research papers were selected for detailed review.

Table 1: Data sources and query results

Data source	Date searched	Total number of results
IEEE Xplore	27/03/2019	463
ACM	27/03/2019	482
ScienceDirect	27/03/2019	67

2 Systematic Literature Review

A systematic literature review (SLR) is the well-defined process of conducting a review of multiple articles and studies in order to answer predefined research questions. An SLR begins by defining a research protocol that involves identifying the research questions to be addressed and defining the research method to be followed to answer those questions. The research protocol explicitly defines the search strategy, and inclusion and exclusion criteria for the selection of primary studies (PSs) [27], as well as providing guidance on how to extract the relevant information from the selected studies. To conduct this SLR, detailed guidelines were extracted from Refs. [28–32]. The step-by-step and well-defined systematic research process followed in this review was extracted from Aftab et al. [32] and is presented in Fig. 2.

2.1 Identification of Research Questions

Research questions reflect the ultimate objective of an SLR and play a key role in the selection of primary studies. The selected primary studies are then reviewed for the extraction of answers to the research questions. The purpose of this study is to analyze and summarize the empirical evidence regarding the use of machine learning techniques for SDP. The following research questions are identified and addressed in this SLR.

RQ1: Which methods/techniques are used in the proposed/used SDP models/frameworks?

RQ2: Which evaluation criteria are used to measure the performance of proposed/used prediction models/frameworks?

RQ3: Which tools are used for the implementation of prediction models/frameworks?

RQ4: Which datasets are selected for the experiments?

RQ5: What is the contribution/novelty of the works by researchers in improving the prediction performance of proposed/used frameworks/models?

RQ6: In the case of comparative analysis, which supervised machine learning algorithms performed better than others?

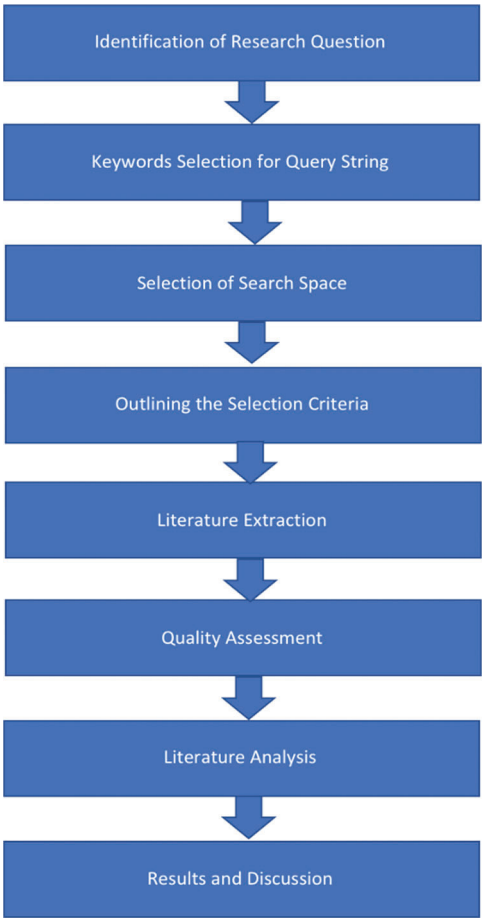

Figure 2: Systematic research process [32]

2.2 Keyword Selection and Query String

This step deals with the selection of particular keywords/words along with their synonyms while keeping in mind the research questions (see Tab. 2). A search string is created by combining keywords and their synonyms using 'AND' and 'OR' operators as shown below:

Table 2: Search string

Keyword	Alternatives/synonyms
Machine learning	('ML', 'machine learner')
Technique	('algorithm' OR 'classifier' OR 'method' OR 'model' OR 'framework' OR 'approach')
Software	('software' OR 'program' OR 'system')
Defect	('fault' OR 'SFP' OR 'SDP' OR 'bug' OR 'error')
Prediction	'determine' OR 'analysis' OR 'estimate*' OR 'explore' OR 'classify'
Evaluation	('accuracy' OR 'efficiency' OR 'performance' OR 'improvement')
Dataset	('data' OR 'defect dataset' OR 'metric dataset')
Tool	('data mining tool' OR 'ML tool' OR 'machine learning tool')

((‘Performance’ OR ‘Accuracy’ OR ‘Efficiency’ OR ‘Improvement’) AND (‘Assessment’ OR ‘Evaluation’) AND (‘Data mining’ OR ‘Machine Learning’) AND (‘Technique’ OR ‘Algorithm’ OR ‘Classifier’ OR ‘Method’ OR ‘Model’ OR ‘Framework’ OR ‘Approach’) AND (‘Software’ OR ‘Program’ OR ‘System’) AND (‘Defect’ OR ‘Fault’ OR ‘Bug’ OR ‘Error’) AND (‘Prediction’ OR ‘Determination’ OR ‘Analysis’ OR ‘Estimation’ OR ‘Exploration’ OR ‘Classification’ OR ‘Forecasting’) AND (‘Dataset’ OR ‘Defect Dataset’ OR ‘Metric Dataset’) AND (‘Tool’ AND ‘Data Mining Tool’ OR ‘Machine Learning Tool’)).

2.3 Selection of Search Space

In this step of the review, online libraries were selected for the extraction of research papers. This SLR focuses on three well-known online libraries: IEEE Xplore, ACM, and ScienceDirect. Because these three online libraries contain different options for searching the relevant material, in order to extract the most appropriate and relevant papers, a query string was searched multiple times in each library with different combinations of keywords. The total number of results from each library is listed in Tab. 1.

2.4 Outlining the Selection Criteria

This step identifies the scope of the study by explicitly defining the selection criteria for the shortlisting of extracted research papers. The purpose of this step is to select the most appropriate research papers for review. This step can be broken down into two steps: defining the inclusion criteria and defining the exclusion criteria.

2.4.1 Inclusion Criteria

a. Research papers published from 2016 until 2019.

b. Research papers presented/published in journals, conferences, conferences proceedings, workshops, or published as book chapters.

c. Research papers that performed SDP with binary classification using supervised machine learning techniques.

d. Research papers with focus on (where training and test data belong to the same project) project defect prediction.

e. Research papers that presented empirical experiments, case studies, comparative studies, or a novel method/technique/framework for software defect prediction.

f. Research papers that presented results of a used/proposed/modified machine learning algorithm/technique/framework on a dataset.

2.4.2 Exclusion Criteria

a. Research papers published before 2016.

b. Research papers that are not in the English language.

c. Research papers that do not perform SDP.

d. Research papers that do not employ a supervised machine learning technique in the method/technique/framework used for SDP.

e. Research papers that do not evaluate the defect prediction method/technique/framework used on any dataset.

2.5 Literature Extraction

This step deals with the extraction of the most relevant and appropriate research papers from the selected research material. The complete process followed in this stage is shown in Fig. 3. The ultimate objective of

this stage is to select the primary studies (most relevant research papers) for the review so that the answers to the research questions can be extracted.

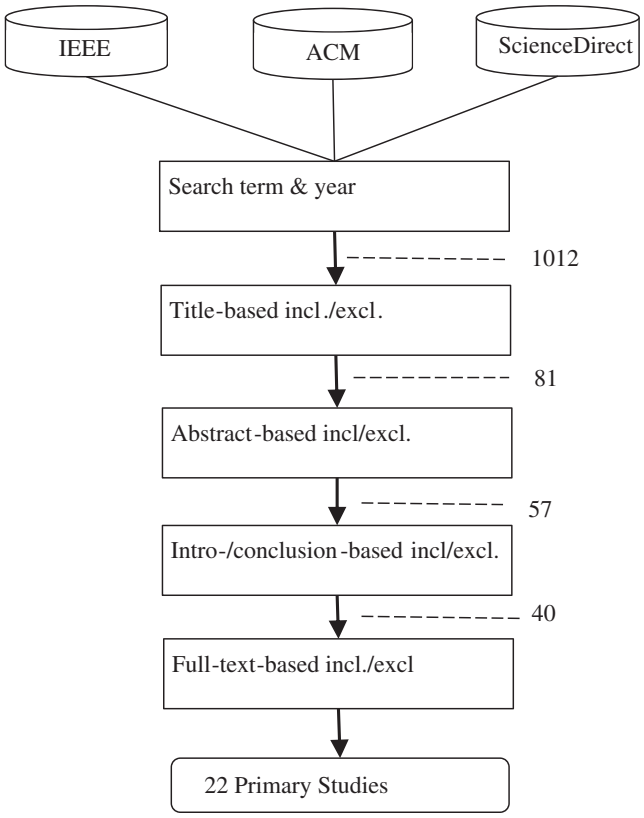

Figure 3: Extraction of primary studies

The tollgate approach [33] was used to shortlist articles for critical review. The tollgate approach consists of five phases P-1 to P-5, and leads to the selection of 22 PSs [34–55], as seen in Tab. 3.

Table 3: Tollgate approach

Selected Sources	P-1	P-2	P-3	P-4	P-5
IEEE Xplore	463	45	35	27	15
ACM	482	22	13	6	4
ScienceDirect	67	14	9	7	3
Total	1012	81	57	40	22

Phase 1 (P-1). Basic search using the search terms and year.

Phase 2 (P-2). Title-based inclusion/exclusion, duplicate articles are removed.

Phase 3 (P-3). Abstract-based inclusion/exclusion.

Phase 4 (P-4). Introduction and conclusion-based inclusion/exclusion.

Phase 5 (P-5). Full text-based inclusion/exclusion.

2.6 Quality Criteria for Study Selection

The purpose of establishing QA criteria is to make sure that selected primary studies provide enough details to answer the identified research questions. The QA and data extraction processes are carried out concurrently. A QA checklist was devised to evaluate the eminence of selected primary studies, as shown in Tab. 4.

Table 4: QA criteria for PSs

S.no.	QA checklist
QA1	Does the study provide enough details of an ML classifier or framework to answer RQ1?
QA2	Are suitable performance measures reported to answer RQ2?
QA3	Does the study include details of the tool used to answer RQ3?
QA4	Does the study include details of the dataset used to answer RQ4?
QA5	Does the study propose a technique to improve the performance of an ML classifier to answer RQ5?
QA6	Does the study provide a suitable analysis on performance improvements to answer RQ6?

Each selected PS is assessed against QA criteria (Tab. 4) and assigned a score of 0 to 1. If the article explicitly answers each of the QA questions, the study is given score of 1; and if it partially answers the question, the study is given a score of 0.5. A score of 0 is assigned to studies that fail to answer the QA question. The final score is calculated by adding up the scores for all the QA questions.

After assessing the quality of a selected PS, it was found that each PS score $\geq 80\%$ against QA criteria. This means the selected PS provides adequate information to address this SLR.

2.7 Literature Analysis

After going through the complete systematic process of relevant literature extraction, 22 primary studies were selected to answer the defined research questions after a detailed critical review. The step of literature analysis includes reviewing the primary studies by keeping in mind the research questions so that succinct answers to those questions can be extracted.

3 Results and Discussion

This is the last and most important step of the SLR process, and yields answers to the research questions identified in the first step of SLR.

RQ1: Which methods/techniques are used in the proposed/used SDP models/frameworks?

More than 30 techniques and algorithms were used by the researchers in the selected primary studies. The researchers used these techniques in order to compare and improve the prediction performance. These techniques included classification algorithms as well as feature selection, re-sampling, and ensemble learning techniques. All of the techniques used are shown in Fig. 4, grouped into 10 classes. During the review, it was observed that Naïve Bayes (NB), K-nearest neighbors (KNN), multilayer perceptron (MLP), logistic regression (LR), decision tree (DT), random forest (RF), and support vector machine (SVM) are the most widely used classifiers in SDP. It can be seen from Fig. 5 that the DT, Bayesian, neural network, kernel, and ensemble classifiers make up 73% of the techniques used. Researchers have also proposed techniques and methods to improve the performance of ML classifiers on software defect predictions. Researchers in Refs. [40,43,44,52] compared the performance of individual ML classifiers on selected datasets, and identified the best performing classifiers. Data preprocessing

techniques, such as feature selection (FS) [34,37,39,45,55] and data balancing [38,42,47,51], are also used to improve the efficiency of models. Researchers have also proposed hybrid, meta learning, and network-based frameworks to detect defects with higher accuracy and precision [35,36,41,45,47–50,53,54].

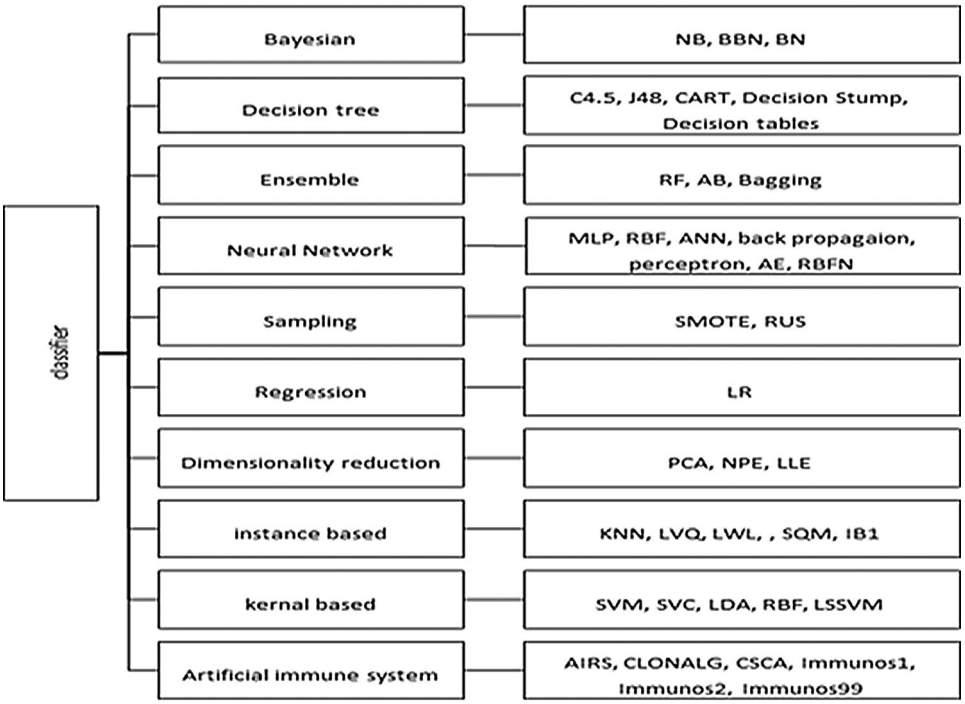

Figure 4: Techniques used in primary studies

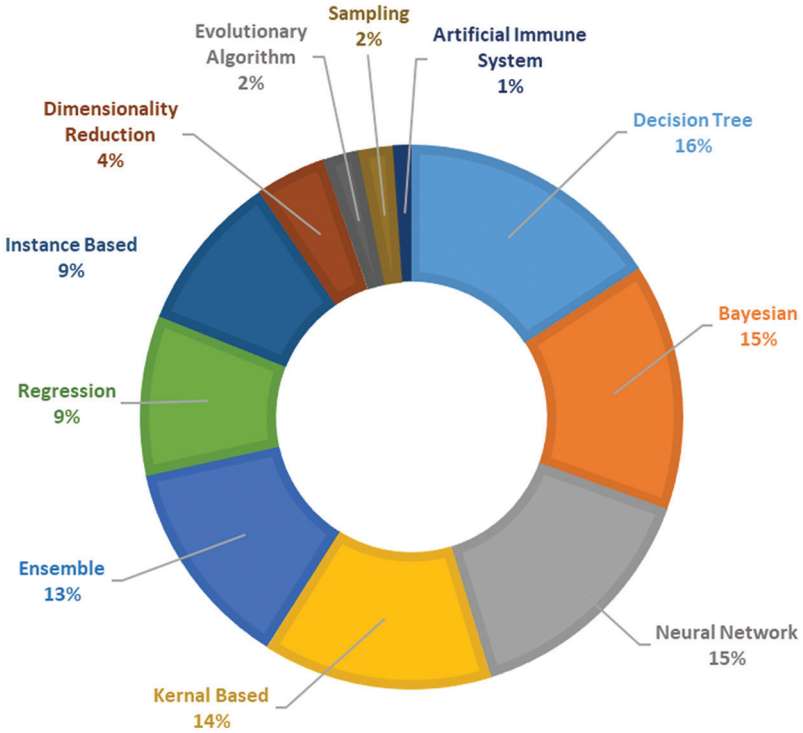

Figure 5: Distribution of studies over techniques used

RQ2: Which evaluation criteria are used to measure the performance of proposed/used prediction models/frameworks?

Researchers have used various performance measures (Tab. 5) to evaluate the performance of used/proposed defect prediction methods. However, most performance measures are calculated from the parameters of a confusion matrix [24,56,57]. The F-measure, area under ROC curve (AUC), recall, precision, and accuracy are frequently used metrics to determine the performance of defect prediction models. These measures are adopted by 79% of the selected primary studies, as shown in Fig. 6. Researchers have also used the Matthews correlation coefficient (MCC), specificity, decision cost, G-mean, precision–recall curve (PR curve), kappa statistic, and standard deviation error to evaluate the performance of prediction models.

Table 5: Performance measures

Measure	Mathematical model
Recall/TPR/ Pd	$TP/(TP+FN)$
Pf/ FPR	$FP/(FP+TN)$
Precision	$TP/(TP+FP)$
Accuracy	$(TP+TN)/(TP+FP+TN+FN)$
F-measure	$(2*recall*precision)/(recall+precision)$
MCC	$(TP \times TN - FP \times FN)/\sqrt{((TP+FP)\times(TP+FN)\times(TN+FP)\times(TN+FN))}$
AUC	Area under ROC curve is the trade-off between TRP and FPR
G-mean	$\sqrt{(pd \times (1-pf))}$
Specificity	$TN(TN+FP)$
PR curve	Trade-off between recall (TPR) and precision (positive predicted value)
Kappa statistic	Compares observed accuracy with expected accuracy
Standard deviation error	Reveals error rate of the model

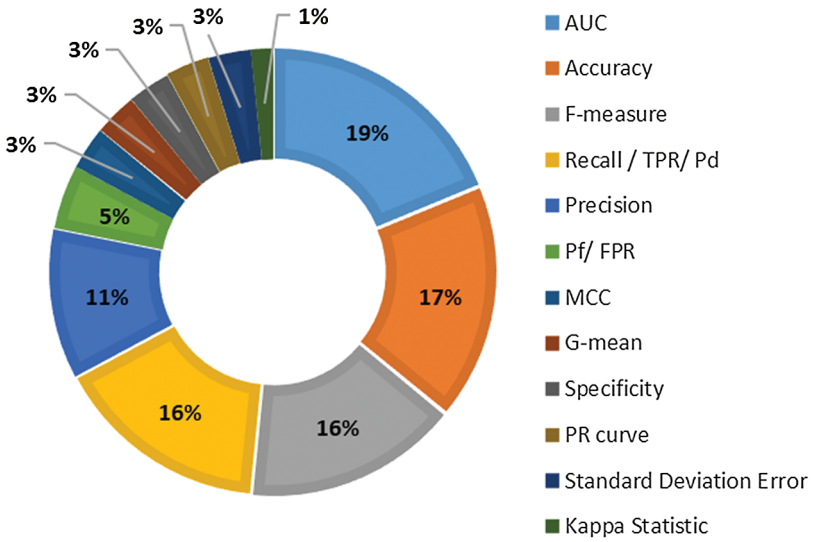

Figure 6: Distribution of studies over performance measure

To further compare and analyze results generated from defect prediction models, researchers applied comparison and difference measurement techniques such as the Scott-Knott ESD test [42], Friedman test [44,46], Nimenyi test [44], paired t-test [34,38,45,47] box-plot diagram [45], and Wilcoxon signed-rank test [36].

Many tools are available to perform data mining tasks [58–60]. Most of the selected primary studies have executed prediction models using the WEKA tool [34,35,38–40,42–44,46,47]. As shown in Fig. 7, WEKA is used by 36% of studies. MATLAB is another tool implemented by many researchers [38,43,45,49]. Some of the selected PSs have not specified the tool used, while some researchers have chosen KEEL [38,40], IBM SPSS [38,52], and LibSVM [49] as their tools.

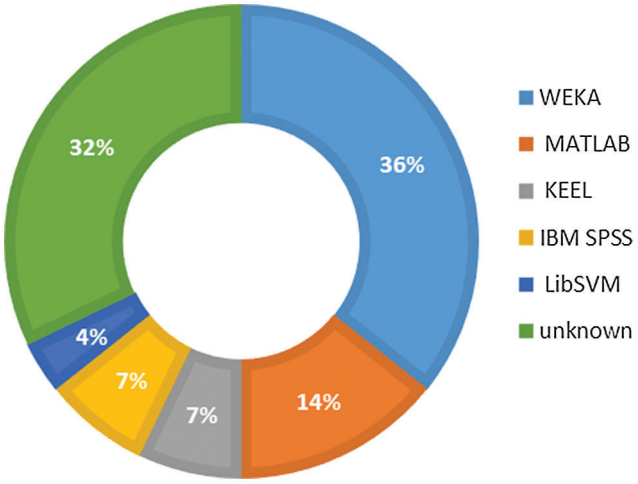

Figure 7: Distribution of studies over machine learning tools

The majority of researchers in our selected primary studies used publicly available datasets for the implementation of proposed/used classification models. A dataset is a collection of features also known as software metrics collected from previously developed software in order to check the accuracy of a proposed model. It has been observed that different classification algorithms perform differently on different datasets [61,62]. Therefore, most studies have used multiple datasets for their experiments. For instance, 73% of the selected studies have used different datasets from the PROMISE and tera-PROMISE repositories (see Fig. 8).

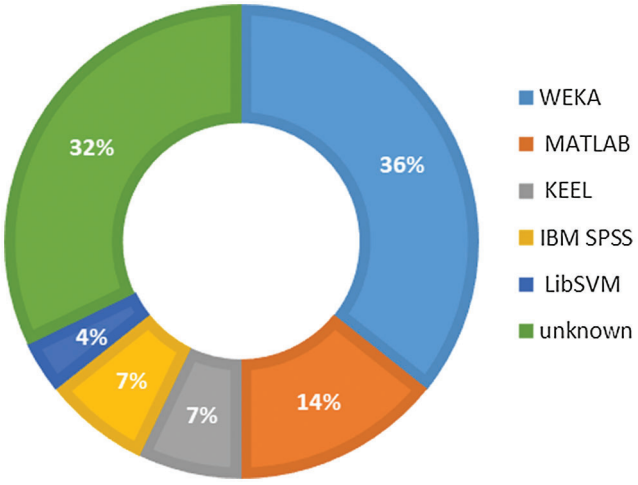

Figure 8: Distribution of studies over dataset repository

RQ3: Which tools are used for the implementation of prediction models/frameworks?

RQ4: Which datasets are selected for the experiments?

Kaur et al. [44] used open-source Java projects PMD, Find Bugs, EMMA, Trove, and Dr Java from SourceForge. Malhotra et al. [46], nine popular open-source projects were collected from a Github repository: caffeine, fast adapter, fresco, freezer, glide, design pattern, jedis, mem-cached, and MPAndroidChart. Phan et al. [50] collected bug data from the programming contest site CodeChef from the solution of four problems, i.e., SUMTRIAN, FLOW016, MNMX, and SUBINC, submitted in the C and C++ programming languages. In Malhotra et al. [55], data were collected from Android software repositories containing Bluetooth, contacts, email, gallery, and telephony data. In Refs. [37,48], the researchers used bug data from three versions of the Eclipse bug prediction dataset.

RQ5: What is the contribution/novelty of the works by researchers in improving the prediction performance of proposed/used frameworks/models?

Of 22 selected primary studies, 12 performed comparative analysis of various classification algorithms on SDP. Researchers compared the prediction performance of different machine learning classifiers and models [34,37–40,42–44,46,51,52,55]. Ten research papers proposed novel frameworks to improve the accuracy of defect prediction [35,36,41,45,47–50,53,54]. In these studies, researchers have proposed hybrid, layered, adaptive learning, and network-based frameworks using baseline ML classifiers.

The answer to this research question focuses on the particular techniques used to improve the performance of SDP models/frameworks/algorithms.

3.1 Data Preprocessing

Data preprocessing sanitizes data to remove inappropriate, irrelevant, redundant, and noisy data [63]. The presence of noisy and redundant data can lead to inaccurate result as shown in Tab. 6.

Table 6: Data preprocessing techniques

Technique	Problem	PS	Method
Feature selection (FS)	High dimensionality	34, 37, 38, 39, 42, 45, 46, 47, 51, 53, 55	CFS, PSO, GA. IG, CS. GR, RF, RFW, SU, Chi-Square, OneR, LR, RSA, NPE
Feature extraction	High dimensionality	39, 49, 51, 54	PCA, KPCA, AE
Feature scaling	Standardization/ normalization	54	Maximum minimization
Data sampling (DS)	Class imbalance	38, 42, 47, 51	RUS, SMOTE, ASMO
Noise filtering	Unclean, meaningless data	38, 42	IPE
Propositionalization	Relational representation to propositional	51	RELAGGS

3.2 K-fold Cross-Validation

In the k-fold cross-validation method, data is divided into k sub-samples. Each sample is used as test data for the validation of models built using k-1 sub-samples [64]. This process is repeated k times. Numerous researchers have used 10-fold cross-validation to predict performance of a model [40–42,46,48,53–55]. In Qu et al. [36], 3-fold cross-validation was used to increase the prediction accuracy of the model.

3.3 Meta-Learning

Dôres et al. [35] proposed a meta-learning framework SPF-MLP to find the best ML learner from a set of learners for a particular project. They used seven algorithms as an input set: NB, RF, C4.5, k-NN, SVM, MLP, and AB. An experiment was conducted on 71 PROMISE datasets, and it was shown through standard deviation and average rank metrics that SPF-MLP recommended the best algorithm for each dataset. To recommend a single algorithm, researchers used RF and an ensemble of seven input algorithms as meta-learners to generate a predictive model from a meta-database. SFP-MLF-EN-7 achieved an average rank of 2.556 while SFP-MLF-RF achieved an average rank of 2.472. These two techniques outperformed the seven input algorithms.

Nucci et al. [41] presented an approach called ASCI (adaptive selection of classifiers in bug prediction) to dynamically select from a set of ML classifiers the one providing the best prediction performance for a class. They used five algorithms, LR, NB, RBFN, MLP, and DT, to compare the prediction performance of the proposed ASCI and a validation-based voting ensemble technique. Results were compared using the F-measure and AUC. The comparative analysis showed that, in 77% of cases, ASCI outperformed MLP, with its F-measure, accuracy, and AUC 7%, 2%, and 1% higher, respectively. ASCI also outperformed the ensemble method in 83% of cases.

Nucci et al. [42] compared the role of different meta-learners in the ASCI method proposed in Nucci et al. [41]. They conducted experiments on 21 open-source projects with DT, C4.5, LR, MLP, NB, and SVM as meta-learners. Their results showed that the choice of meta-learner did not have a significant impact on the performance improvement of the model, and as a result, lightweight classifiers were recommended for meta-learning purposes.

3.4 Network Embedding Technique

Qu et al. [36] developed a model called node2defect using a network embedding technique. This model first created a class dependency network (CDN) of the program. Subsequently, node2ver was used to learn a vector to encode structural features of the CDN. Node2defect then concatenated the vectors with metrics, and used them in ML classifiers. RF was used as the ML classifier in this model. The proposed model was evaluated on 15 open-source Java programs, and compared with traditional ML classifiers using 3-fold cross-validation. With cross-validation, the F-measure was improved by 9.2% on average, and the AUC was improved by 3.86% on average. When compared to ASCI, the defect prediction model with node2defect showed improvements in F-measure of 9.1% and in AUC of 5.6%.

Yang et al. [54] proposed a dynamic predictive threshold filtering algorithm to propose the best prediction model for a dataset. They used a complex network technology to create a set of multilayer structural feature metrics that showed the overall characteristics of a feature. The proposed model was tested using multilayer structural feature metrics, a single version of the defect cross-validation, and CK metrics on 154 versions of the software called BaseRecyclerViewAdapterHelper. The model obtained through multilayer metrics showed better performance on average among all models.

3.5 Other Proposed Techniques

Kumar et al. [45] built a model using LSSVM with linear, polynomial, and RBF kernel functions. The model was developed using object-oriented (OO) source code metrics. FS was performed to demonstrate that a small subset of OO metrics improved performance. Moreover, it was observed that the LSSVM model built with the RBF kernel performed better than models with other kernels.

Miholca et al. [47] developed HyGRAR, a non-linear hybrid model that integrated gradual relational association rule mining and an ANN to predict defects. They tested this model on 10 open-source software projects from PROMISE, and then compared it with other machine learning methods using the AUC metric. HyGRAR outperformed the other methods in 98% of cases.

Maheshwari et al. [48] proposed a three-way decision-based model for defect prediction. In the first step, modules were divided into three classes based on threshold value, using an NB classifier: defected, non-defected, and deferred. Deferred modules were classified in the second step using an RF ensemble learning technique. Their model was compared with NB baseline classifiers, and showed improved accuracy, F-measures, and decision costs in three versions of the Eclipse dataset.

Kareshk et al. [49] proposed a pretraining technique for a shallow ANN. Pretraining was performed using DAE. The proposed method was compared with four versions of SVMs (SVM, PCA-SVM, KPCA-SVM, and AE-SVM) and an ANN without pretraining on seven datasets. The proposed model performed best on four datasets and second best on the remaining three datasets.

Phan et al. [50] formulated an approach convolution on assembly instruction sequences called Application-specific convolutional neural network (ASCNN). In this approach, the source code was converted into assembly code, and a multi-view convolutional neural network was used to learn defective features from the assembly instruction sequences. Results showed the improved performance of ASCNN, suggesting that learning from assembly code might be beneficial to detect semantic bugs.

Wei et al. [53] proposed an improved NPE-SVM approach in which a manifold learning algorithm was used to reduce dimensions, and an SVM was used as a baseline defect predicting classifier. The performance of the proposed model was compared with that of SVM, LLE-SVM, and NPE-SVM and it showed superior performance on 13 datasets.

RQ6: In the case of comparative analysis, which supervised machine learning algorithms performed better than others?

From the selected primary studies, 12 out of 22 conducted a comparative analysis of classification techniques on SDP by using various datasets. Brief descriptions of their comparisons are presented in Tab. 7.

It can be inferred that the decision tree-based algorithm showed better overall prediction performance in most of the reviewed studies. Moreover, RF also performed well in most of the cases in these studies.

Table 7: Comparative studies and performance improvements

PS	ML Classifier	Comparative results
[34]	NB, LR, DT, RF, MLP	No specific pattern was observed in performance before FS; higher performance was observed for LR 23.07%, RF 46.15%, and MLP 15.38%. FS improved performance of all classifiers. However, in 69.2% cases, NB performed better than DT.

(Continued)

Table 7 (continued).

PS	ML Classifier	Comparative results
[37]	LR, NB, J48	J48 performed better than NB and LR for change metrics. For most of the cases, J48 performed better for the complete set of change metrics and source code metrics, and for the combination of both.
[38]	NB, RF, KNN, MLP, SVM, J48	All showed improvements in performance as more data processing was done. On average, RF performed better than other classifiers. J48 achieved the second-best performance among the classifiers. The data balancing technique SMOTE outperformed RUS in improving prediction performance.
[39]	DT, RF, NB, SVM, ANN, Adaboost	CFS performed well in feature reduction, followed by PCA. Because PCA showed decreased performance in most of the cases, it is not recommended for FS. Comparative analysis showed that RF had higher accuracy than all other classifiers, and ANN had the second-best accuracy on 8 out of 12 cases.
[40]	PSO, NB, DT, NN, Linear classifier	The linear classifier showed the highest prediction accuracy in 4 of 7 datasets, and proved to be the most efficient classifier. Results were also compared using standard deviation error, and showed that NN had the lowest error rate, followed by DT.
[43]	LWL, SMO, NB, J48, RF, bagging, BBN, MF	RF, Bagging+RF, and BBN performed best in terms of AUC and PRC curves. In terms of F-measure bagging and BBN showed slightly better performance than RF on JM1. For the KC3 dataset, J48, NB, and BBN performed well in terms of ROC, recall, and PRC curves. For the MC1 dataset, bagging, BBN, and SMO had high recall rates; in terms of AUC, bagging, RF, and BBN performed well. On average, bagging and BBN showed good performance in terms of recall on all datasets.
[44]	LR, NB, J48, RF, bagging, IB1	In this paper, RF performed best on all datasets, while bagging achieved the second-best rank in all cases. NB showed lower performance in most cases, and is not recommended for defect prediction.

Table 7 (continued).

PS	ML Classifier	Comparative results
[46]	single layer perceptron, MLP, AIRS, LVQ, SOM, CLONAL, Immune	The single-layer perceptron outperformed in terms of AUC for OO metrics. In nine projects, the AUC value for the single-layer perceptron was between 0.852 and 0.997. Results were also confirmed using Friedman's test.
[51]	BN, NB, logistic function, simple logistic function, SMO, AdaBoostM1, Lazy IBK, bagging, regression, J48, RF, random tree	Before data preprocessing, J48, Lazy IBK, BN, and random tree performed best out of 13 classifiers. Performance accuracy was improved after applying data preprocessing techniques. Results showed that propositionalizing was better than PCA and FS; moreover, a 50% improvement in performance was observed for the BN classifier. The class imbalance problem was addressed using SMOTE and ASMO, and results showed that ASMO brought higher performance of classifiers than SMOTE.
[52]	MLP, RBF, CART, KNN	MLP and RBF performed better than CART and KNN in most of the cases. For MLP, AUC values for seven datasets ranged from 0.57 to 0.99, and for RBF, they ranged from 0.56 to 0.94. KNN showed poor performance across datasets. MLP performed best on five of seven datasets. Results were confirmed using Friedman's test, and confirm MLP as the best performer.
[55]	RF, LR, DT, NB, SVM	Use of FS gave a better result for RF, DT, and SVM but did not improve performance for statistical classifiers LR and NB. In terms of precision, RF performed best on all datasets except the contacts dataset, for which NB performed best. After FS with PSO and GA, RF performed best on Bluetooth, email, and gallery datasets while the SVM performed best on contacts and telephony datasets.

4 Conclusion

The process of software defect prediction (SDP) can be used as a quality assurance activity in the software development life cycle to detect defective modules before the testing stage. This prediction can be used for the development of a quality product with lower cost, since in testing stage, only those modules detected as defective will be tested. Over the last decade, many researchers have been working to improve the performance of SDP. Although researchers have also conducted reviews and provided survey papers in this domain, there remains a lack of a current picture of research trends. This study filled the gap by providing a systematic literature review of the research papers published from 2016 to 2019.

For literature extraction, three well-known online libraries were used: IEEE Xplore, ACM, and ScienceDirect. This research was initiated with the identification and formulation of six research questions that targeted almost all of the important aspects of SDP. A comprehensive systematic research process was followed to answer the identified research questions. Initially, 1012 studies were extracted from the online libraries; a step-by-step literature extraction process was followed to shortlist the most relevant studies, which resulted in 22 papers. It has been concluded that researchers have tried to improve the prediction performance by introducing novel techniques in data preprocessing as well as integrating multiple classifiers using meta-learners. Some researchers have proposed novel frameworks by integrating multiple techniques for multiple processes. Furthermore, many researchers have performed comparative analysis of supervised machine learning classifiers on multiple datasets in order to identify the few techniques that performed best on all of the datasets. This approach can lead us to only focus on those well-performing classifiers while designing novel models and frameworks for SDP. It has been observed by analyzing these comparative studies that decision tree-based techniques performed well on most datasets, along with random forest. This SLR will guide researchers' future works by providing the present picture of research trends in the SDP domain.

Acknowledgement: Thanks to our families and colleagues, who provided moral support.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] A. Iqbal, S. Aftab, U. Ali, Z. Nawaz, L. Sana *et al.,* "Performance analysis of machine learning techniques on software defect prediction using NASA Datasets," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 5, pp. 1–9, 2019.

[2] C. Jin, "Software defect prediction model based on distance metric learning," *Soft Computing*, vol. 25, no. 1, pp. 447–461, 2021.

[3] A. Iqbal, S. Aftab and F. Matloob, "Performance analysis of resampling techniques on class imbalance issue in software defect prediction," *International Journal of Information Technology and Computer Science*, vol. 11, no. 11, pp. 44–53, 2019.

[4] F. Matloob, S. Aftab and A. Iqbal, "A framework for software defect prediction using feature selection and ensemble learning techniques," *International Journal of Modern Education and Computer Science*, vol. 11, no. 12, pp. 14–20, 2019.

[5] M. A. Khan, S. Abbas, A. Atta, A. Ditta, H. Alquhayz *et al.,* "Intelligent cloud-based heart disease prediction system empowered with supervised machine learning," *Computers Materials & Continua*, vol. 65, no. 1, pp. 139–151, 2020.

[6] A. Iqbal and S. Aftab, "A classification framework for software defect prediction using multi-filter feature selection technique and MLP," *International Journal of Modern Education and Computer Science*, vol. 12, no. 1, pp. 18–25, 2020.

[7] T. Menzies, Z. Milton, B. Turhan, B. Cukic, Y. Jiang *et al.,* "Defect prediction from static code features: Current results, limitations, new approaches," *Automated Software Engineering*, vol. 17, no. 4, pp. 375–407, 2010.

[8] F. Orlando, "Gartner says global IT spending to grow 3.2 percent in 2019" [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2018-10-17-gartner-says-global-it-spending-to-grow-3-2-percent-in-2019/ (Accessed: 24 Apr 2019).

[9] O. F. Arar and K. Ayan, "Software defect prediction using cost-sensitive neural network," *Applied Soft Computing*, vol. 33, no. 1, pp. 263–277, 2015.

[10] S. Mohapatra and B. Mohanty, "Defect prevention through defect prediction: A case study at Infosys," *IEEE Int. Conf. on Software Maintenance*, Florence, Italy, pp. 260–268, 2001.

[11] P. Michaels, "Faulty software can lead to astronomic costs, 2008," (Accessed: 24 Apr 2019). [Online]. Available: http://www.computerweekly.com/opinion/Faulty-software-can-lead-to-astronomic-costs, ComputerWeekly.com

[12] D. R. Ibrahim, R. Ghnemat and A. Hudaib, "Software defect prediction using feature selection and random forest algorithm," in *Int. Conf. on New Trends in Computer Science*. Amman, Jordan, pp. 252–257, 2018.

[13] R. S. Wahono, "A systematic literature review of software defect prediction: Research trends, datasets, methods and frameworks," *Journal of Software Engineering*, vol. 1, no. 1, pp. 1–16, 2015.

[14] M. Ahmad, S. Aftab, S. S. Muhammad and S. Ahmad, "Machine learning techniques for sentiment analysis: A review," *International Journal of Multidisciplinary Sciences and Engineering*, vol. 8, no. 3, pp. 27–39, 2017.

[15] M. Ahmad and S. Aftab, "Analyzing the performance of SVM for polarity detection with different datasets," *International Journal of Modern Education and Computer Science*, vol. 9, no. 10, pp. 29–36, 2017.

[16] M. Ahmad, S. Aftab and I. Ali, "Sentiment analysis of tweets using SVM," *International Journal of Computer Applications*, vol. 177, no. 5, pp. 25–29, 2017.

[17] S. Aftab, M. Ahmad, N. Hameed, M. S. Bashir, I. Ali *et al.,* "Rainfall prediction in Lahore city using data mining techniques," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 4, pp. 4–9, 2018.

[18] M. Ahmad, S. Aftab, M. S. Bashir, N. Hameed, I. Ali *et al.,* "SVM optimization for sentiment analysis," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 4, pp. 45–49, 2018.

[19] A. Iqbal, S. Aftab, I. Ullah, M. A. Saeed and A. Husen, "A classification framework to detect DoS attacks," *International Journal of Computer Network and Information Security*, vol. 11, no. 9, pp. 40–47, 2019.

[20] A. Iqbal and S. Aftab, "A feed-forward and pattern recognition ANN model for network intrusion detection," *International Journal of Computer Network and Information Security*, vol. 11, no. 4, pp. 19–25, 2019.

[21] K. E. Bennin, K. Toda, Y. Kamei, J. Keung, A. Monden *et al.,* "Empirical evaluation of cross-release effort-aware defect prediction Models, " in *IEEE Int. Conf. on Software Quality, Reliability and Security.* Vienna, Austria, pp. 214–221, 2016.

[22] L. Goel, D. Damodaran, S. K. Khatri and M. Sharma, "A literature review on cross project defect prediction," in *4th IEEE Uttar Pradesh Section Int. Conf. on Electrical, Computer and Electronics*. Mathura, India, pp. 680–685, 2017.

[23] T. Hall, S. Beecham, D. Bowes, D. Gray and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1276–1304, 2012.

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

[25] C. Catal, "Software fault prediction: A literature review and current trends," *Expert Systems with Applications*, vol. 38, no. 4, pp. 4626–4636, 2011.

[26] Z. Li, X. Y. Jing and X. Zhu, "Progress on approaches to software defect prediction," *IET Software*, vol. 12, no. 3, pp. 161–175, 2018.

[27] P. V. T. Carrión, C. S. G. González, S. Aciar and G. R. Morales, "Methodology for systematic literature review applied to engineering and education," in *IEEE Global Engineering Education Conf.*, Tenerife, Spain, pp. 1364–1373, 2018.

[28] S. Ashraf and S. Aftab, "Scrum with the spices of agile family: A systematic mapping," *International Journal of Modern Education and Computer Science*, vol. 9, no. 11, pp. 58–72, 2017.

[29] S. Ashraf and S. Aftab, "Latest transformations in scrum: A state of the art review," *International Journal of Modern Education and Computer Science*, vol. 9, no. 7, pp. 12–22, 2017.

[30] M. Ahmad, S. Aftab, M. S. Bashir and N. Hameed, "Sentiment analysis using SVM: A systematic literature review," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 2, pp. 182–188, 2018.

[31] F. Anwer and S. Aftab, "Latest customizations of XP: A systematic literature review," *International Journal of Modern Education and Computer Science*, vol. 9, no. 12, pp. 26–37, 2017.

[32] S. Aftab, M. Ahmad, N. Hameed, M. S. Bashir, I. Ali *et al.,* "Rainfall prediction using data mining techniques: A systematic literature review," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 5, pp. 38–43, 2018.

[33] W. Afzal, R. Torkar and R. Feldt, "A systematic review of search-based testing for non-functional system properties," *Information and Software Technology*, vol. 51, no. 6, pp. 957–976, 2009.

[34] K. K. Ganguly and B. M. M. Hossain, "Evaluating the effectiveness of conventional machine learning techniques for defect prediction: A comparative study," in *Joint 7th Int. Conf. on Informatics, Electronics & Vision CIEV.* Kitakyushu, Japan, pp. 481–485, 2018.

[35] S. N. D. Dôres, L. Alves, D. D. Ruiz and R. C. Barros, "A meta-learning framework for algorithm recommendation in software fault prediction," in *31st Annual ACM Sym. on Applied Computing*. Pisa, Italy, pp. 1486–1491, 2016.

[36] Y. Qu, T. Liu, J. Chi, Y. Jin, D. Cui *et al.,* "Node2defect: Using network embedding to improve software defect prediction," in *33rd IEEE/ACM Int. Conf. on Automated Software Engineering*. Montpellier, France, pp. 844–849, 2018.

[37] Y. A. Alshehri, K. G. Popstojanova, D. G. Dzielski and T. Devine, "Applying machine learning to predict software fault proneness using change metrics, static code metrics, and a combination of them," in *IEEE South East Conf*. St. Petersburg, FL, USA, pp. 1–7, 2018.

[38] K. Bashir, T. Li, C. W. Yohannese and Y. Mahama, "Enhancing software defect prediction using supervised-learning based framework," in *12th Int. Conf. on Intelligent Systems and Knowledge Engineering*. Nanjing, China, pp. 1–6, 2018.

[39] G. P. Bhandari and R. Gupta, "Machine learning based software fault prediction utilizing source code metrics," in *IEEE 3rd Int. Conf. on Computing, Communication and Security.* Kathmandu, Nepal, pp. 40–45, 2018.

[40] P. D. Singh and A. Chug, "Software defect prediction analysis using machine learning algorithms," in *7th Int. Conf. on Cloud Computing, Data Science & Engineering - Confluence*, Noida, India, pp. 775–781, 2017.

[41] D. D. Nucci, F. Palomba, R. Oliveto and A. D. Lucia, "Dynamic selection of classifiers in bug prediction: An adaptive method," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 3, pp. 202–212, 2017.

[42] D. D. Nucci and A. D. Lucia, "The role of meta-learners in the adaptive selection of classifiers," in *IEEE Workshop on Machine Learning Techniques for Software Quality Evaluation*. Campobasso, Italy, pp. 7–12, 2018.

[43] J. Ge, J. Liu and W. Liu, "Comparative study on defect prediction algorithms of supervised learning software based on imbalanced classification data sets, " in *19th IEEE/ACIS Int. Conf. on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*. Busan, South Korea, pp. 399–406, 2018.

[44] A. Kaur and I. Kaur, "An empirical evaluation of classification algorithms for fault prediction in open-source projects," *Journal of King Saud University-Computer and Information Sciences*, vol. 30, no. 1, pp. 2–17, 2018.

[45] L. Kumar, S. K. Sripada, A. Sureka and S. K. Rath, "Effective fault prediction model developed using Least Square Support Vector Machine (LSSVM)," *Journal of Systems and Software*, vol. 137, pp. 686–712, 2018.

[46] R. Malhotra, L. Bahl, S. Sehgal and P. Priya, "Empirical comparison of machine learning algorithms for bug prediction in open-source software, " in *Int. Conf. on Big Data Analytics and Computational Intelligence*. Chirala, India, pp. 40–45, 2017.

[47] D. L. Miholca, G. Czibula and I. G. Czibula, "A novel approach for software defect prediction through hybridizing gradual relational association rules with artificial neural networks," *Information Sciences*, vol. 441, no. 2, pp. 152–170, 2018.

[48] S. Maheshwari and S. Agarwal, "Three-way decision-based defect prediction for object-oriented software, " in *Int. Conf. on Advances in Information Communication Technology & Computing*. Bikaner, India, pp. 1–6, 2016.

[49] M. O. Kareshk, Y. Sedaghat and M. R. T. Akbarzadeh, "Pre-training of an artificial neural network for software fault prediction," in *7th Int. Conf. on Computer and Knowledge Engineering*, Mashhad, Iran, pp. 223–228, 2017.

[50] A. V. Phan and M. L. Nguyen, "Convolutional neural networks on assembly code for predicting software defects," in *21st Asia Pacific Sym. on Intelligent and Evolutionary Systems*, Hanoi, Vietnam, pp. 37–42, 2017.

[51] S. Rizwan, W. Tiantian, S. Xiaohong and U. Salahuddin, "Empirical study on software bug prediction," in *Int. Conf. on Software and e-Business*. New York, NY, United States, pp. 55–59, 2018.

[52] P. Singh and R. Malhotra, "Assessment of machine learning algorithms for determining defective classes in an object-oriented software," in *6th Int. Conf. on Reliability, Infocom Technologies and Optimization*. Noida, India, 204–209, 2018.

[53] H. Wei, C. Shan, C. Hu, H. Sun and M. Lei, "Software defect distribution prediction model based on NPE-SVM," *China Communications*, vol. 15, no. 5, pp. 173–182, 2018.

[54] Y. Yang, J. Ai and F. Wang, "Defect prediction based on the characteristics of multilayer structure of software network," in *IEEE Int. Conf. on Software Quality, Reliability and Security Companion*. Lisbon, Portugal, pp. 27–34, 2018.

[55] R. Malhotra and A. Khurana, "Analysis of evolutionary algorithms to improve software defect prediction," in *6th Int. Conf. on Reliability, Infocom Technologies and Optimization*. Noida, India, pp. 301–305, 2018.

[56] "Precision and recall" [Online]. Available: https://en.wikipedia.org/wiki/Precision_and_recall. (Accessed: 24 Apr 2019).

[57] Y. Jiao and P. Du, "Performance measures in evaluating machine learning based bioinformatics predictors for classifications," *Quantitative Biology*, vol. 4, no. 4, pp. 320–330, 2016.

[58] V. Gupta and P. Devanand, "A survey on data mining: Tools, techniques, applications, trends and issues," *International Journal of Scientific & Engineering Research*, vol. 4, no. 3, pp. 1–14, 2013.

[59] K. Rangra and K. L. Bansal, "Comparative study of data mining tools," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, no. 6, pp. 2277, 2014.

[60] "Weka 3: Data Mining Software in Java" [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka/ (Accessed: 24 Apr 2019).

[61] 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 *IEEE 23rd Int. Conf. on Software Analysis, Evolution, and Reengineering*. Suita, Japan, pp. 370–381, 2016.

[62] Z. Xu, J. Liu, Z. Yang, G. Anand and X. Jia, "The impact of feature selection on defect prediction performance: An empirical comparison," in *IEEE 27th Int. Sym. on Software Reliability Engineering*. Ottawa, Canada, pp. 309–320, 2016.

[63] A. Saleem, K. H. Asif, A. Ali, S. M. Awan and M. A. Alghamdi, "Pre-processing methods of data mining," in *IEEE/ACM 7th Int. Conf. on Utility and Cloud Computing*. London, UK, pp. 451–456, 2014.

[64] K. S. Raju, M. R. Murty and M. V. Rao, "Support vector machine with k-fold cross validation model for software fault prediction," *International Journal of Pure and Applied Mathematics*, vol. 118, no. 20, pp. 321–334, 2018.