

Insights of effectivity analysis of learning-based approaches towards software defect prediction

Deepti Rai, Jyothi Arcot Prashant

Department of Computer Science and Engineering, M. S. Ramaiah University of Applied Sciences, Bengaluru, India

Article Info

Article history:

Received May 30, 2023

Revised Jul 12, 2023

Accepted Dec 13, 2023

Keywords:

Deep learning

Machine learning

Software defect prediction

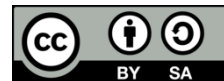
Software fault

Software quality

ABSTRACT

Software defect prediction is one of the essential sets of operation towards mitigating issues of risk management in software development known to contribute towards enhancing the quality of software. There is evolution of various methodologies towards resolving this issue while learning-based methodology is witnessed to be the most dominant contributor. The problem identified is that there are yet many unsolved queries associated with practical viability of such learning-based approach adoption in software quality management. Proposed approaches discussed in this paper contributes towards mitigating this challenge by introducing a simplified, compact, and crisp analysis of effectiveness associated with learning-based schemes. The paper presents its major findings of effectivity analysis of machine learning, deep learning, hybrid, and other miscellaneous approaches deployed for fault prediction followed by highlighting research trend. The major findings infer that feature selection, data imbalance, interpretability, and in adequate involvement of context are prime gaps in existing methods. The paper also contributes towards research gap as well as essential learning outcomes of present review work.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Deepti Rai

Department of Computer Science and Engineering, M. S. Ramaiah University of Applied Sciences

New BEL Rd, M S R Nagar, Mathikere, Bengaluru, Karnataka 560054, India

Email: deeraisecond@gmail.com

1. INTRODUCTION

The organization involved in development of software products are required to deal with various risk and threats in its development process irrespective of the size of an organization and one such important risk factor is associated with software defects [1]. There are various types of software defects viz. configuration defect, maintainability defects, compatibility defect, usability defect, security defect, performance defect, and functional defect [2]–[8]. The presence of defect in the finally developed software results in higher amount of risk associated with financial and resources which can be controlled when an effective defect predictive is applied. The mechanism of software defect prediction involves using statistical models as well as software metric in order to determine the potential defect before even they surface in real time [9]. The prim intention of software defect prediction is mainly to enhance the quality of software as well as minimize the software development cost by early evaluation of risk in the form of software defect [10]. At present, there are various standard methodologies and approaches associated with software defect prediction viz. i) the first type of such approach is termed as static-analysis based predictive approach where the source code is evaluated and critical defects are identified on the basis of security vulnerabilities, inconsistency, and coding errors [11], ii) the second type of such approach is called as dynamic-analysis based predictive approach which is carried out during runtime of software by monitoring its behavior in the form of different

runtime errors, buffer overflows, and memory leaks [12], iii) the third type is known as model-based predictive approach which make use of either machine learning or statistical model in order to perform prediction. Usually, such process depends upon the initial training with the standard dataset which is facilitated with both information about defects and software quality [13], iv) the fourth type is called expert-based predictive approach which is completely based on experience and skilled knowledge of domain expert in order to determine the defect [14], and v) the last type of approach is called as hybrid-based predictive approach which integrates multiple variants of above-mentioned approaches for similar cause [15]. Out of all this, machine learning approaches are found to be frequently adopted owing to their potential ability to learn the unique patterns from the massive-sized dataset of software metric in order to perform prediction. Apart from this, it is also found that such approaches also consider usage of specific form of metrics viz. i) cohesion-based metric (quantity of methods in specific module or class), ii) coupling metric (quantity of inter-module dependencies), and iii) complexity-based metric (quantity of conditionals, depth of nesting, size of code) [16]. Adoption of such metric is carried out in order to perform machine learning-based training with an agenda of forecasting the presence of defects in specific modules or classes. After the training is accomplished, different classes or modules with maximal risk probability is subjected for identification.

Prior to understanding and realize the statement of the problem, it is essential to determine the varied labels of issues and ongoing challenges associated with software defect prediction as follows: i) the first research problem is associated with establishing the connectivity between the defect and the attributes, which is one of the significant factors for ascertaining the accuracy of defect. Further, adoption of accurate metric level is still undefined; although there are various studies being carried out; but the outcome have been accomplished over different set of attributes, where a proper inference cannot be drawn effectively, ii) there is lack of robust and reliable parameter towards assessing the performance of such predictive tool; apart from this adoption of standard criterion is significantly missing in existing scheme which can be proven for wide-range of applicability in multiple and different software projects, iii) usage of local data is carried out towards model learning from some old project; however very often the development team encounters higher dependencies of such local data which are unavailable sometimes due to manifold reason. Prediction scheme using cross-projects is one of the alternative solutions to deal with this issue where prediction of defect is carried out in specific project while analysis is carried out in different software project. Unfortunately, predictive scheme developed on cross-project may have claimed for higher accuracy; but their outcome is strictly restricted to some set of software domain and are sub-optimal in its predictive analysis, iv) at present, there is no report of any available generalized framework for software defect prediction; adoption of different predictive methodologies on different types of dataset is the prime reason for this issue, v) majority of the research models towards software defect prediction is carried out on the basis of simplified usage without any emphasis towards cost modelling associated with various uncertain factors. This results in misclassification as well as interpretability issues, and vi) class imbalance has always been a critical problem in software defect prediction irrespective of various literatures being consistently addressing them. Therefore, the clear statement of the problem can be arrived as “developing an optimal predictive scheme towards accurately determining software defect with balanced computational efficiency and wide range of applicability on different software project is yet a computationally challenging task”.

The above-mentioned issue can be more effectively justified on the basis of varied ranges of relevant literatures. Adoption of machine learning approaches towards prediction of defects explicitly considering the use case of mobile application is discussed by Jorayeva *et al.* [17]. The discussion carried out by Abdu *et al.* [18] have stated varied methods for usage of semantic attributes using deep learning approaches for predicting the software defects. Further work stating the significance of deep learning approaches is noted in work of Akimova *et al.* [19] and Giray *et al.* [20]. Aziz *et al.* [21] has presented discussion associated with multiple inheritance metrics in order to perform analysis of software fault prediction. Generalized discussion towards various learning-based methodologies were carried out by Cao [22], Mahmud *et al.* [23], Son *et al.* [24] and Kotte and Qyser [25]. From the perspective of machine learning approach, Khan *et al.* [26] have presented discussion of predictive approaches using artificial neural network (ANN). Pal and Sillitti [27] have presented discussion about various predictive approaches for cross-projects. All the above-mentioned relevant literatures proves that there are significant number of research studies towards varied solution in software defect prediction.

However, it is yet not totally clear to understand the best solution or clear and pin-pointed interpretation of research gaps. Therefore, the proposed study manuscript presents a compact and yet clear visualization of the effectiveness of some of the recent and most relevant schemes of software defect prediction. The core objective of this manuscript is to review the potential approaches reported in current era towards software defect prediction with crisp information. The new value of the research is presented in the form of following contribution of this manuscript as i) the study reviews most significant literatures associated with both learning and non-learning approaches, ii) all the reviewed literatures have been exhaustively reviewed with respect to problems being addressed, specification of adopted methodologies,

reported beneficial factors, and identified shortcoming, iii) a highlights of recent research trends towards adoption of individual approaches of both machine learning and deep learning is carried out to understand the frequency of usage of specific methodologies, iv) a clear insight is furnished towards the identified research gap from the review work to determine the open-end challenges associated with it, and v) learning outcome of the study in the form of significant contribution to further offer a researcher's viewpoint towards facilitating further direction of study by considering constructive suggestion. Hence, the proposed review work offers a crisp, updated, and informative contents towards adoption of learning-based approaches in software defect prediction. The next section outlines the adopted methodology in order to carry out the present review work.

2. METHOD

The prime purpose of the proposed study is to carry out a review of learning-based approach towards understanding its strength and weakness towards improving software defect prediction. A desk research methodology is adopted for this purpose with simplified sequence of following task as shown in Figure 1 viz: i) the first task is to perform an information identification where the publication of research articles from reputed archives with high impact factor is sought. In this phase, all the research articles published between 2013 to till date has been collected which mainly deals with implementation plans; ii) the second task is to perform initial filtering of all the collected articles by screen the abstract to ensure that it meets the agenda of study i.e., to find implementation strategy; iii) the third task is to perform identification and removal of duplicates which refers to exactly same implementation strategy or different papers written by same author bearing nearly the same techniques; and iv) the fourth task is to perform a secondary screening where the methodology and result is screened. The methodology is screened to understand the uniqueness of the algorithm and technique used for software fault prediction while result screening is to find the strength and weakness of the associated implemented methodology. This task of both primary and secondary screening is carried out considering inclusion and exclusion criteria.

The inclusion criteria involve only the research article bearing implementation plan with enough evidence of result work while the exclusion criteria involve considering any implementation paper published before 2013. Finally, the outcome of the secondary screening process results in extraction of research gap as well as extraction of learning outcomes. The research gap is extracted on the basis of review of limitation extended to further understanding the global perspective of the methodology under screening. The learning outcomes consists of researcher's self-opinion which is based on perspective of complete review of strength and weakness of existing methodologies. In the entire course of this methodology, equal extraction process is carried out towards understanding the research trend on the basis of various phases of methodology involved in proposed study. The idea of research trend is to look for frequency of adoption of different learning-based scheme towards software defect prediction. Therefore, this methodology, in its simplified form, assists in accomplishing research objectives.

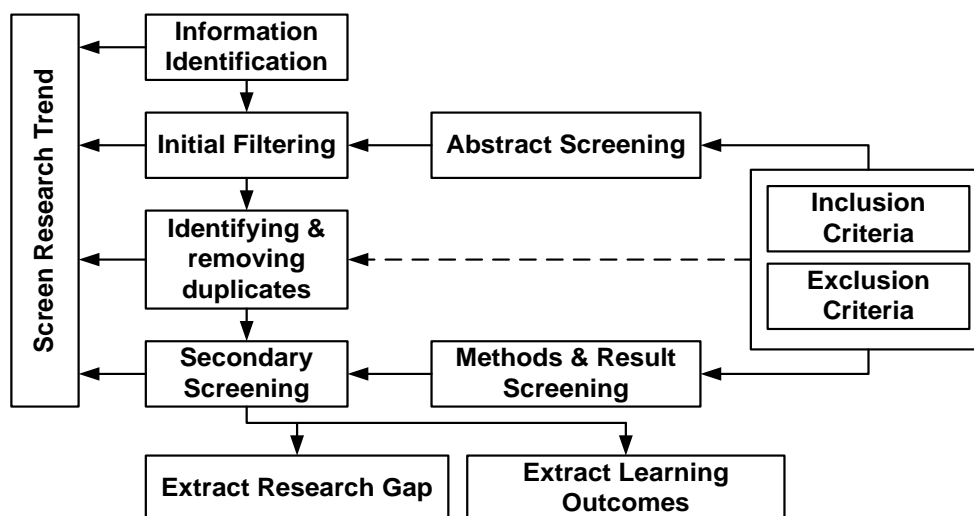


Figure 1. Adopted method in present investigation

3. RESULTS

This section presents the briefing of the study findings associated with the usage of different approaches towards prediction of software fault. There are different variants of techniques adopted towards this purpose where machine learning as well as deep learning-based algorithms are dominant techniques while some other approaches are reviewed as well. The core agenda of the discussion of this section is to highlight the effectiveness of all the reviewed research articles to understand their further applicability to deal with the problems of software fault prediction.

3.1. Existing studies deploying machine learning approaches

Adoption of machine learning-based approaches has been witnessed in different way in order to address different challenges associated with software fault prediction. The work carried out by Aftab *et al.* [28] have used a combination of decision tree (DT), artificial neural network (ANN), and naïve Bayes (NB) in order to carry out classification of different faults in software design on cloud. The system also used fuzzy logic (FL) in order to incorporate accuracy associated with the prediction. Problems associated with the sub-optimal solution due to usage of different ensemble model for prediction of software fault is addressed in work of Alazba and Aljamaan [29] where hyperparameters have been tuned up for multiple ensemble approaches with tree design e.g., CatBoost, XGBoost, gradient boosting using histogram, normal gradient boosting, AdaBoost, and random forest. Adoption of ensemble approach towards classification problem was witnessed in work of Alsawalqah *et al.* [30] where a simplified classifier and ensembled classifier is designed for robust classification. Aziz *et al.* [31] have used standard software metric i.e., Chidamber and Kemerer (CK) metric along with machine learning for investigating the possible influence of this metric on predicting software fault. The study uses ANN for model building considering repositories with and without inheritance of CK metric. Bal and Kumar [32] have addressed the issue of data imbalance while perform prediction of software fault by introducing a weighted regularization scheme of machine learning approach in order to accomplish a balanced outcome. Study towards similar problem of data imbalance is also carried out by Khout and Le [33] where ensemble learning approach is used for prediction of software fault. The study model uses DT, support vector machine (SVM), k-nearest neighbor (KNN), Bayesian network (BN), and multilayer perceptron (MLP) for performing classification.

Some of the studies have linked software defects with hardware failures in existing era and introduces a solution for this as reported in work of Boateng *et al.* [34]. According to this study model, feedforward neural network and linear regression method has been used for investigating the cost associated with hardware failures for optic networks. Adoption of machine learning has been further investigated by Khoudry *et al.* [35] using K-nearest neighbor and Gaussian process towards identifying fault associated with high-speed power mechanism. The work carried out by Lee and Seo [36] have used KNN and SVM along with latent Dirichlet allocation (LDA) in order to enhance the software bug reporting system. Adoption of unsupervised learning scheme is witnessed in work of Marjuni *et al.* [37] by using spectral classifier to deal with zero-threshold problems. The study presents a unique threshold with median absolute deviation considering eigen vector. Further, a unique case study is considered by Nasir *et al.* [38] in order to identify the level of tolerance that can offer by information-centric software. Voutsinas *et al.* [39] have used a machine learning approach in order to identify the fault considering photovoltaic system. The idea of this study model is to perform classification of different types of faults associated with use-case while it also reduces the computational cost associated with its operation in terms of minimal memory.

3.2. Existing studies deploying deep learning approaches

Deep learning approaches has been recently evolved to prove its effectiveness towards prediction of software fault owing to its potential advantage towards accuracy accomplishment in its analytical process. The work carried out by Deng and Qiu [40] have addressed the problems of semantic characteristic associated with programming language towards generation of defective code by using long short-term memory (LSTM). This is used for learning the contextual and semantic features associated with the source code. The mechanism calls for constructing an abstract tree of the program in order to evaluate each data within the tree nodes. Adoption of LSTM was also reported in work of Munir *et al.* [41] where it has been integrated with gated recurrent unit (GRU) with an objective of classifying faults. The source code is subjected to parsing and a tree is formulated along with incorporation of 32-level matrix features.

The work carried out by Hai *et al.* [42] have presented a technique to track the bugs present within cloud environment using deep learning. The core notion of this study model is to minimize the cost and time demands for performing assessing of software defect tracking. The scheme has been implemented using multiple multi-layer perceptron (MLP) configuration over multiple standard datasets. Adoption of deep learning is also witnessed in work of Jorayeva *et al.* [43] where LSTM is integrated with convolution neural network considering the user case of open-source Android application software defects. The analysis also has perform comparative analysis for ANN, convolution neural network (CNN), and LSTM to show that

performance of CNN and LSTM is always better than ANN while CNN is slightly more better than LSTM. Similar adoption of CNN and LSTM integration is also reported in work of Farid *et al.* [44]; however, the authors have used bidirectional LSTM (Bi-LSTM) towards prediction of software defect caused due to semantics associated with the source code. The study model formulates a syntax tree to represent the vectors linked with programs where the extraction of semantics is carried out by CNN model and retention of key features are carried out by Bi-LSTM. A unique form of deep learning approach known as contrastive learning is implemented by Luo *et al.* [45] towards enhancing the identification of software defect in the form of bug. The study model initially performs pretraining on the corpus of bug reports using unsupervised scheme of learning followed by training with contrastive learning. The contribution of this scheme is that it assists in learning the semantic distinction between buggy files and defect reports. Another unique adoption of deep learning is carried out by Maduako *et al.* [46] where the defect analysis is carried out for power transmission lines. This scheme re-tunes the CNN model along with a novel pyramid network with multiscale layer in order to localize the faults. Pan *et al.* [47] have developed a computational model where an enhanced CNN model is used for predicting software faults. The author has developed a new dataset of source codes where CNN has been implemented and the outcome shows that improved CNN offers better performance on defect prediction compared to conventional CNN. Qasem *et al.* [48] have investigated the impact of deep learning towards fault prediction by integrated usage of MLP and CNN. Wang *et al.* [49] have used gated LSTM of hierarchical form where the semantic features of the codes are extracted in the form of syntax tree. The model is claimed to be capable of extracting both conventional and semantic features of software using gated fusion technique. The further studies towards mining semantic feature have been carried out by Yao *et al.* [50]. According to this study model, a CNN model is developed in the form of tree-based structured where extraction of semantic feature is carried out from grammatical structure of code as well as text information within the code.

3.3. Existing studies deploying hybrid learning approaches

There are also certain studies where hateful and offensive speech detection is carried out jointly. The work It is known that adoption of both machine learning as well as deep learning has been frequently exercised in order to accomplish a better prediction performance for software defect. However, there are also approaches which has combined both machine learning and deep learning scheme in order to form a hybrid approach to further harness the predictive potential of both the learning schemes for optimized performance. One such unique and simple form of a hybrid learning model has been presented in work of Asmawi *et al.* [51] by integrating deep learning and machine learning approach. The idea of this work is towards predicting the failures of cloud-based software. The findings of the study state that extreme gradient boosting is found to be suitable model towards processing essential features associated with disk and central processing unit (CPU) spaces while random forest and DT method is found be suitable model towards task prioritization in the course of software defect prediction. Borandag *et al.* [52] have developed another hybrid scheme by integrating recurrent neural network (RNN) with ensemble machine learning technique. The authors have used CNN, LSTM, and Bi-LSTM as a part of deep learning approach while 5 machine learning techniques (i.e., naïve Bayes, SVM, KNN, random tree, and K-Star) over benchmarked dataset evaluated over object-oriented metric. Khalid *et al.* [53] have presented another unique model towards software defect prediction where machine learning technique has been integrated with bio-inspired algorithm in order to formulate a new hybrid learning algorithm. The study has implemented K-means clustering as machine learning approach integrated with particle swarm optimization (PSO) as a part of bio-inspired algorithm in order to perform classification of features. The study outcome shows that SVM offers the best optimal results compared to other machine learning approach. Similar trend of hybrid approach is also carried out by Zhang *et al.* [54] where a back propagation neural network is integrated with cuckoo search optimization scheme. The idea of this study model is to perform prediction of faults in industrial equipment's with an outcome exhibiting improved training time and convergence performance.

3.4. Other approaches

Apart from conventional learning-based algorithms, there are various approaches towards performing prediction of software defects. It has been noted that adoption of cross-projects analysis of defect is one of the better alternative solutions to overcome the issues of historical records; however, it is also characterized with shortcoming associated with prime distinction between target software project with source project. This issue is addressed in work of Bala *et al.* [55] who have adopted a selection scheme on the basis of features and transformation. Adoption of transformation method is towards minimization of difference in distribution of projects while minimization of high dimensionality and unnecessary attributes are governed by selection of optimal features in this study model. Further, the study model presented by Hassounh *et al.* [56] have used bio-inspired approach for addressing the challenges in software defect

prediction. The authors have implemented whale optimization algorithm where the exploration process towards optimal outcome is improved. Further, process improvement is carried out using multiple method of selection i.e., random, stochastic universal sampling, linear rank, roulette wheel, and tournament. A distinct scheme towards prediction of software fault is discussed by Lee *et al.* [57] where an analytical model is presented in the form of a software network that connects both software module and developers in order to investigate the interaction between them. The idea of this concept is to derive the sub-graph that is characterized by bad structure of research objects in order to indicate the software fault. Study towards similar form of concept is implemented by Li *et al.* [58] where a tri-relation network is designed for prediction of software faults. This form of network integrates developer, module dependency, and contribution of developer in order to investigate their joint influence towards software quality. The work carried out by Phung *et al.* [59] have presented a unique scheme where software metric is formulated for error representation associated with java-based runtime. A formal modelling method is used along with machine learning in order to evaluate the patterns. The work of Tumar *et al.* [60] have used moth flame optimization scheme in order to carry out selection of feature associated with fault in software design. The study has also used an adaptive synthetic sampling method along with moth flame optimization in order to perform selection of wrapper feature as well as to mitigate the issue of imbalanced dataset. The uniqueness of this study is also to convert the bioinspired approach to a binary version using transfer function. Further, the study model has also used linear discriminant analysis (LDA) along with DT and KNN for performing classification. The work carried out by Zhang *et al.* [61] have developed a scheme to predict the defect associated with labelled data using cross-version model. The implementation design contributes towards solving issues related to class overlapping and distribution of data variation.

Table 1 highlights the summarization of the all the reviewed existing scheme deploying machine learning, deep learning, hybrid method, and other miscellaneous unique methodology towards software fault prediction. The tabular content exhibits that the existing system has adopted different variants of techniques which are associated with claimed beneficial features as well as they are also characterized with limiting features. The inference of the learning outcome of this table is that although existing software fault prediction has made some significant progress but still there are associated limitation associated with almost all the adopted methodologies.

3.5. Research trend

At present there are various research work being carried out towards software fault prediction mainly using machine learning and deep learning approaches in last decade. According to the information stated in Table 2, it can be noted that there are approximately 74,032 research articles published in last ten years. It should be noted that these publications are cumulative of all learning and non-learning-based scheme towards enhancing the predictive performance of software fault detection. Another indication inference from Table 2 is that there is a significant number of interests being shown from the research community towards software fault prediction owing to increasing number of research articles.

However, it is further essential to know the contribution of core learning-based approaches in this regard. Therefore, this research trend is exhibited in Table 3 and Table 4 where contribution of machine learning and deep learning approaches are exhibited. Table 3 has exhibited deployment of various frequently used machine learning schemes viz. Naïve Bayes (NB), decision tree (DT), support vector machine (SVM), random forest (RF), k-nearest neighbor (KNN). The outcome of this research trend showcases that adoption of SVM is consistently on rise followed by adoption of KNN and NB approach. The deployment of DT and RF is found to be less adopted with progression of research publications. One interesting point to be noted here is that in an era of technological advancement in artificial intelligence, the adoption of conventional machine learning approach SVM, KNN, and NB is still in use owing to its potential advantage towards predictive performance in software fault detection. It is also noted that these techniques are adopted in two ways viz. i) acts as a core model for prediction or ii) it acts as a benchmark model for comparative assessment of presented predictive model. In either of both the cases, they are highly helpful from research perspective.

Table 4 highlights the trends of adoption of deep learning methods viz. Convolution neural network (CNN), long short-term memory (LSTM), recurrent neural network (RNN), generative adaptive network (GAN), radial basis function network (RBFN), multilayer perceptron (MLP), and self-organizing map (SOM). It is noted that LSTM is one of the deep learning approaches that has been extensively deployed which is followed by equal number of usages of CNN and GAN. Although, RNN has been also equivalently deployed; however, majority of the research article has reportedly used RNN as initial process followed by finally deploying its enhanced variant i.e., LSTM. Adoption of SOM is still on steady pace but not extensively found to be used as compared to other variants of deep learning approaches. Apart from this, the analysis towards non-learning approaches is really scattered and no significant trend is noticed and still it forms a minority-approaches towards software defect prediction.

Table 1. Summary of existing approaches

Authors	Problem	Methodology	Advantage	Limitation
Aftab <i>et al.</i> [28]	Software defect prediction	DT, ANN, NB, FL	91.05% accuracy	Cost effectiveness not analyzed
Alazba and Aljamaan [29]	Usage of default hyperparameters in ensemble	Optimization of hyperparameter of multiple ensemble	Effective optimization	Model does not have practical constraints
Alsawalqah <i>et al.</i> [30]	Software defect prediction	Hybrid ensemble	Supports heterogeneous classification	Not applicable for dynamic faults
Aziz <i>et al.</i> [31]	Impact of CK metric on Fault prediction	ANN	Proves the importance of inheritance in prediction	Not benchmarked with other prediction approach
Bal and Kumar [32]	Data imbalance in prediction	Weighted regularization, feed forward neural network	Effective fault detection	Study does not perform binary classification
Khuat and Le [33]	Data imbalance in prediction	DT, SVM, KNN, BN, and MLP	Simplified model	No constraints in modelling
Nyarko-Boateng [34]	Cost evaluation in hardware failures	Linear regression, feedforward neural network	Satisfactory accuracy, practical utilization	No benchmarking
Khoudry <i>et al.</i> [35]	Fault prediction for high-speed power system	KNN, Gaussian	Works both online and offline	Dynamic faults are not associated with the model
Lee and Seo [36]	Improving software bug reporting system	LDA, KNN, SVM	Increased accuracy	Use-case specific performance
Marjuni <i>et al.</i> [37]	Zero threshold issues	Unsupervised Learning, heuristic row sum method	Improved performance of classification	No benchmarking
Voutsinas <i>et al.</i> [39]	Fault detection in photovoltaic system	Machine learning	Simplified learning system, 97.11% accuracy	Needs further extensive analysis, specific to use case
Deng <i>et al.</i> [40]	Contextual/semantic feature challenges	LSTM	Simplified and user-friendly model	Less extensive analysis
Munir <i>et al.</i> [41]	Software defect prediction	LSTM	Effective classification performance	No benchmarking
Hai <i>et al.</i> [42]	Software defect tracking in cloud	MLP	Highly simplified model implementation	Demands higher iteration and training dependencies
Jorayeva <i>et al.</i> [43]	Software defect prediction in mobile application	LSTM, CNN	93% accuracy	Dynamic defects not assessed
Farid <i>et al.</i> [44]	Semantics linked with source codes	Bi-LSTM, CNN	Improve detection of faults	Assessed only on java projects
Luo <i>et al.</i> [45]	Localization of bug	Contrastive Learning	Complete automated contextual model	Models demands predefined information of bugs
Maduako <i>et al.</i> [46]	Component fault detection	CNN, pyramid network with multiscale feature	Supports representation of multiple fault characteristic	Study model specific to data type
Pan <i>et al.</i> [47]	Software defect prediction	Improved CNN	Better detection performance	Computational complexity issue
Qasem <i>et al.</i> [48]	Software defect prediction	CNN, MLP	Comprehensive analytical model	Induces computational burden
Wang <i>et al.</i> [49]	Software defect prediction	LSTM (gated)	Distributes the computational complexities during prediction	No benchmarking
Yao <i>et al.</i> [50]	Semantic Extraction	CNN, feature mining of semantics	Higher performance score	Lower assessment scope
Asmawi <i>et al.</i> [51]	Prediction of cloud-based defects	Hybrid learning	Short time for prediction	Low accuracy score
Borandag [52]	Software defect prediction	Hybrid learning	95% accuracy	Demands more processing resources
Khalid <i>et al.</i> [53]	Software defect prediction	Machine learning, bioinspired algorithm	Optimal feature learning	Premature convergence not addressed
Zhang <i>et al.</i> [54]	Fault prediction in industrial equipment	Cuckoo search optimization, backpropagation neural network	Good convergence performance, and reduced training time	Model not applicable for dynamic/uncertain defects
Bala <i>et al.</i> [55]	Cross project software defect prediction	Transformation & selection of feature	Can process high dimensional feature	Scalability issues
Hassouneh <i>et al.</i> [56]	Feature selection in software defect prediction	Whale optimization	High data quality, higher reliability	No consideration of uncertain risk attributes
Lee <i>et al.</i> [57]	Human-based software errors	Tree network of developer and module	Can be customized based on demand	Highly sensitive to slightest fault
Li <i>et al.</i> [58]	Human-based software errors	Tri-relational network	Simplify the process of prediction	Needs extensive analysis to proves it applicability
Phung <i>et al.</i> [59]	Fault identification	Software metric with error type, formal modelling, machine learning	Can identify run-time error	Applicable for java environment only
Tumar <i>et al.</i> [60]	Feature selection for fault prediction	Moth flame optimization, machine learning, transfer function	Improved classification performance	Iterative model
Zhang <i>et al.</i> [61]	Challenges in fault prediction in labelled data	Cross-version model	Effective clustering performance	Computational complexity not assessed

Table 5 highlights trends towards the adoption of different dataset [62]–[80] which are reported to be frequently deployed in existing studies. However, a unique trend is noted towards the usage of such standard and frequently reported dataset. For an example, it is noted that adoption of University College London (UCL) machine learning (ML) repository is used for varied forms of machine learning approaches itself while adoption of NASA, PROMIZE, and GHPR dataset is reported to be used more using AdaBoost and bagging, Naïve Bayes, random forest [79]. It is also noted that adoption of Poi, Synapse, Xerces, Xalam, Lucene, jEdit, Camel dataset is used for CNN-based approaches mainly. The dataset of AR6, AR4, AR5, jEdit 4.3, AR3, jEdit 4.2, AR1, jEdit 4.0, Anr 1.7, Tomcat 6.0 has been reportededly used for deploying HyGRAR modelling. The dataset KC1, KC3, PC5, PC3, PC4, MC2, PC1, MC1, CM1, CM1, MW1, JM1 is mainly used towards study model using SVM, MLP, D, KNN, RF, [80]. All these datasets consist of approximately 22-40 attributes (class attribute, size attribute, McCabe attribute, Halstead attribute) with number of instances ranging between 63-9593 while all the dataset bears a numerical data type.

Table 2. Trend of research publication in software fault prediction

Publication	No.
IEEE	144
MDPI	54
Springer	12863
Elsevier	22594
Hindawi	5278
ACM	27500
Taylor and Francis	5599

Table 3. Trend of usage of machine learning approaches in software fault prediction

Publication	NB	DT	SVM	RF	KNN
IEEE	2	6	10	7	3
MDPI	3	2	6	2	1
Springer	2	15	7	7	4
Elsevier	898	4558	6464	2905	2427
Hindawi	6	34	48	30	15
ACM	30801	32679	34520	32433	34032
Taylor & Francis	158	908	855	437	699

Table 4. Trend of usage of deep learning approaches in software fault prediction

Publications	CNN	LSTM	RNN	GAN	RBFN	MLP	SOM
IEEE	6	3	4	2	0	2	5
MDPI	0	2	1	0	0	0	0
Springer	1	2	1	1	0	0	8
Elsevier	2835	2429	2017	571	1751	853	910
Hindawi	41	27	7	10	0	8	5
ACM	31507	34660	31588	33941	34384	27643	33817
Taylor & Francis	188	407	169	80	338	105	714

Table 5. Trend of usage of frequently used dataset for software fault prediction

Publication	Work Carried out by
UCI ML repository	Khan <i>et al.</i> [26]
NASA Dataset	Bowes <i>et al.</i> [62], Pandey <i>et al.</i> [63], Chen and Dai [64], Mustaqeem and Saqib [65], Marapelli <i>et al.</i> [66]
Poi, Synapse, Xerces, Xalam, Lucene, jEdit, Camel	Dam <i>et al.</i> [67], Farid <i>et al.</i> [44], Hosseini <i>et al.</i> [68], Sikic <i>et al.</i> [69], Li <i>et al.</i> [70]
AR6, AR4, AR5, jEdit 4.3, AR3, jEdit 4.2, AR1, jEdit 4.0, Anr 1.7, Tomcat 6.0	Miholca <i>et al.</i> [71]
KC1, KC3, PC5, PC3, PC4, MC2, PC1, MC1, CM1, CM1, MW1, JMI	Iqbal <i>et al.</i> [72]
PROMISE dataset	Pachouly <i>et al.</i> [73], Zain <i>et al.</i> [74], Bal <i>et al.</i> [75], Bahaa <i>et al.</i> [76]
GHPR dataset	Batool and Khan [77], Pan <i>et al.</i> [78]

Therefore, from the highlight of information discussion in this section in perspective of research trend, it is noted that there is a unique pattern of deployment of learning-based approaches in software defect prediction, where a greater number of research articles is witnessed for deep learning-based approach (= 2,41,043) compared to machine learning approach (n=1,84,984) in last ten years. Apart from this, it is also

noted that various dataset too has unique mechanism of adoption which is specific to learning-based approach. However, more adoption of standard dataset is reported for machine learning schemes in contrast to deep learning schemes towards software defect prediction. The next section discusses about the learning outcomes of the review with respect to research gap.

3.6. Research gap

After reviewing the existing implementation approaches of different variants towards software defect prediction, it is noted that that both machine learning and deep learning are the most dominant approaches owing to its beneficial features. However, there are shortcomings associated with both the approaches which is reported in prior section. A closer insight to shortcomings of both the learning approaches also exhibit that there are various critical open-end problems that are yet not reported to be addressed in existing research models. Following are the research gap explored:

- a. Inability to furnish proper contextual information: majority of experiments of the existing learning-based models towards software defect prediction is carried out over historical data in order to offer a predictive outcome. However, the predictive data lacks any form of consideration towards contextual attributes e.g., requirements of software project, or any alterations in personnel or development practices.
- b. Issues of interpretability: from the viewpoint of application-based interpretation, it is noted that machine learning offers more user-friendly interpretability compared to deep learning approaches. Adoption of deep learning approaches significantly increases accuracy levels but cannot provide clear interpretability of its outcome with respect to the associated defects.
- c. Unclear evidence of applicability: adoption of both machine learning and deep learning were proven to accomplish its claimed results in existing studies addressing the specific set of problem. However, it is to be noted that such accuracy level or betterment in its outcome is only applicable if it is used with same training data. It is still unclear about its applicability in practical environment where test data could be highly dynamic and challenging even to process and thereby impose issues in predictability as well.
- d. Challenges in feature selection: from the feature engineering viewpoint, it is known that deep learning offers better outcome compared to machine learning approach. However, these hypothetical assumptions vary from case to case. At present, different variants of work carried out does not offer a concrete justification behind the selection of feature with the success factor of the model to address the problem as much.
- e. Issues of data imbalance: existing studies adopts different available standards datasets for predicting software defects; however, in real environment, the availability of information towards software detection could be very vague or less information. This adoption often leads to biased model in practical application within an organization where massive number of software projects with different challenges in information is required to be analyzed.
- f. Challenges in integration: different levels of software development team use different standards for their software project development by adopting different methodologies. However, the adoption of methodologies for project development and identification of software bugs are always considered as two different tasks. Integrating development environment with defect detection is still an open-end issue that further requires a solution.
- g. Challenges in ascertaining suitable dataset: different researchers adopt different dataset in their experiments to prove the efficiency of their study model. However, the data in real words potentially lacks any form of labelling in its information. Hence, acquiring information and labelling them is quite a computationally extensive task yet to meet its effective solution.

4. CONCLUSION

Predicting software defect has been investigated since more than a decade and has evolved with various solution; however, adoption of learning-based solution is quite high. The paper has offered discussion about exclusively about the usage of machine learning as well as deep learning-based methodologies towards software defect prediction as well as it has also highlighted usage of hybrid models and other miscellaneous methodologies reported. The study evaluation found advantages as well as shortcoming associated with each and every technique while the proposed review contributes towards highlighting the essential research gap. Further after reviewing the overall study approaches, various learning outcomes in the form of review contribution have been arrived as: i) The existing problems towards selection of an optimal features while applying learning approach is to adopt potential feature engineering method towards enhancing accuracy. Further, an integrated adoption of feature selection along with scaling of feature and reduction of dimensionality problems will be further more beneficial towards reducing computational burden of learning-based approaches; ii) Adoption of hybrid approaches as well as ensemble approaches are less in number in existing studies and hence, it is advisable towards more adoption of hybrid scheme towards minimization of

model variance that can significantly increase the ability of generalization of learning model. Various techniques e.g., stacking, boosting, and bagging. should be involved in order to address this issue; iii) Although, transfer learning is an integral part of CNN, but it could be used individually too as a distinct set of operators that can actually improve learning performance on different set of tasks. The ability of gaining knowledge can be significantly improved upon usage of transfer learning especially in presence of low scaled data availability; and iv) Adoption of active learning is also noticed to be less incorporated in existing schemes whereas this form of learning offers opting for highly informative data to be labelled by an expert. This adoption will further minimize the computational effort towards labelling demands and directly enhance the predictive accuracy. This form of learning is one of the better alternatives while dealing with imbalanced dataset or during any investigation where the labelling cost can be anticipated as maximum. Therefore, the future direction of the study will be to act upon above-mentioned learning outcomes of this review study.

REFERENCES





- [1] 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," *Scientific Programming*, vol. 2023, pp. 1–18, Feb. 2023, doi: 10.1155/2023/6221388.
- [2] U. Bhardwaj, A. E. Sand, and M. Warriar, "Stability of <100> dislocations formed in W collision cascades," *Journal of Nuclear Materials*, vol. 569, Oct. 2022, doi: 10.1016/j.jnucmat.2022.153938.
- [3] M. Maddeh, S. Ayouni, S. Alyahya, and F. Hajje, "Decision tree-based design defects detection," *IEEE Access*, vol. 9, pp. 71606–71614, 2021, doi: 10.1109/ACCESS.2021.3078724.
- [4] J. Liao and W. Feng, "Compatibility defects of the fiber-optic liquid level and refractive index sensors based on modal interference," *Physica B: Condensed Matter*, vol. 624, Jan. 2022, doi: 10.1016/j.physb.2021.413398.
- [5] N. S. M. Yusop, J. Grundy, J.-G. Schneider, and R. Vasa, "Preliminary evaluation of a guided usability defect report form," in *2018 25th Australasian Software Engineering Conference (ASWEC)*, Nov. 2018, pp. 81–90, doi: 10.1109/ASWEC.2018.00019.
- [6] M. A. Puentes, Y. Lei, N. Rakotondravony, L. T. Harrison, and C. A. Shue, "Visualizing web application execution logs to improve software security defect localization," in *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, Mar. 2022, pp. 1183–1190, doi: 10.1109/SANER53432.2022.00138.
- [7] Y. Zhao, K. Damevski, and H. Chen, "A systematic survey of just-in-time software defect prediction: online supplement," *ACM Computing Surveys*, vol. 55, no. 10, pp. 1–35, Feb. 2023, doi: 10.1145/3567550.
- [8] H. Wang and L. Yuan, "Software engineering defect detection and classification system based on artificial intelligence," *Nonlinear Engineering*, vol. 11, no. 1, pp. 380–386, Jan. 2022, doi: 10.1515/nleng-2022-0042.
- [9] I. Arora and A. Saha, "ELM and KELM based software defect prediction using feature selection techniques," *Journal of Information and Optimization Sciences*, vol. 40, no. 5, pp. 1025–1045, Jul. 2019, doi: 10.1080/02522667.2019.1637999.
- [10] A. Sinha, S. Singh, and D. Kashyap, "Implication of soft computing and machine learning method for software quality, defect and model prediction," in *Multi-Criteria Decision Models in Software Reliability*, CRC Press, 2022, pp. 45–80.
- [11] C. Bird, T. Menzies, and T. Zimmermann, "The art and science of analyzing software data," *The Art and Science of Analyzing Software Data*, pp. 1–648, 2015, doi: 10.1016/C2012-0-07289-4.
- [12] R. Avros *et al.*, "Boosted decision trees for behaviour mining of concurrent programmes," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 21, Aug. 2017, doi: 10.1002/cpe.4268.
- [13] K. Wang, L. Liu, C. Yuan, and Z. Wang, "Software defect prediction model based on LASSO-SVM," *Neural Computing and Applications*, vol. 33, no. 14, pp. 8249–8259, May 2021, doi: 10.1007/s00521-020-04960-1.
- [14] R. Malhotra and M. Khanna, "Threats to validity in search-based predictive modelling for software engineering," *IET Software*, vol. 12, no. 4, pp. 293–305, Aug. 2018, doi: 10.1049/iet-sen.2018.5143.
- [15] C. Manjula and L. Florence, "Deep neural network based hybrid approach for software defect prediction using software metrics," *Cluster Computing*, vol. 22, no. S4, pp. 9847–9863, Jan. 2019, doi: 10.1007/s10586-018-1696-z.
- [16] T. H. Chen, W. Shang, M. Nagappan, A. E. Hassan, and S. W. Thomas, "Topic-based software defect explanation," *Journal of Systems and Software*, vol. 129, pp. 79–106, Jul. 2017, doi: 10.1016/j.jss.2016.05.015.
- [17] M. Jorayeava, A. Akbulut, C. Catal, and A. Mishra, "Machine learning-based software defect prediction for mobile applications: a systematic literature review," *Sensors*, vol. 22, no. 7, Mar. 2022, doi: 10.3390/s22072551.
- [18] A. Abdu, Z. Zhai, R. Algabri, H. A. Abdo, K. Hamad, and M. A. Al-antari, "Deep learning-based software defect prediction via semantic key features of source code—systematic survey," *Mathematics*, vol. 10, no. 17, Aug. 2022, doi: 10.3390/math10173120.
- [19] E. N. Akimova *et al.*, "A survey on software defect prediction using deep learning," *Mathematics*, vol. 9, no. 11, May 2021, doi: 10.3390/math9111180.
- [20] G. Giray, K. E. Bennis, Ö. Köksal, Ö. Babur, and B. Tekinerdogan, "On the use of deep learning in software defect prediction," *Journal of Systems and Software*, vol. 195, Jan. 2023, doi: 10.1016/j.jss.2022.111537.
- [21] S. R. Aziz, T. A. Khan, and A. Nadeem, "Efficacy of inheritance aspect in software fault prediction—A survey paper," *IEEE Access*, vol. 8, pp. 170548–170567, 2020, doi: 10.1109/ACCESS.2020.3022087.
- [22] H. Cao, "A systematic study for learning-based software defect prediction," *Journal of Physics: Conference Series*, vol. 1487, no. 1, Mar. 2020, doi: 10.1088/1742-6596/1487/1/012017.
- [23] M. H. Mahmud, M. T. H. Nayan, D. M. N. A. Ashir, and M. A. Kabir, "Software risk prediction: systematic literature review on machine learning techniques," *Applied Sciences*, vol. 12, no. 22, Nov. 2022, doi: 10.3390/app122211694.
- [24] L. H. Son, N. Pritam, M. Khari, R. Kumar, P. T. M. Phuong, and P. H. Thong, "Empirical study of software defect prediction: a systematic mapping," *Symmetry*, vol. 11, no. 2, Feb. 2019, doi: 10.3390/sym11020212.
- [25] A. Kotte and D. A. Moiz Qyser, "A survey of different machine learning models for software defect testing," *European Journal of Molecular & Clinical Medicine*, vol. 7, no. 9, 2021.
- [26] M. A. Khan *et al.*, "Software defect prediction using artificial neural networks: a systematic literature review," *Scientific Programming*, vol. 2022, pp. 1–10, May 2022, doi: 10.1155/2022/2117339.
- [27] S. Pal and A. Sillitti, "Cross-project defect prediction: A literature review," *IEEE Access*, vol. 10, pp. 118697–118717, 2022, doi: 10.1109/ACCESS.2022.3221184.

- [28] S. Aftab *et al.*, “A cloud-based software defect prediction system using data and decision-level machine learning fusion,” *Mathematics*, vol. 11, no. 3, Jan. 2023, doi: 10.3390/math11030632.
- [29] A. Alazba and H. Aljamaan, “Software defect prediction using stacking generalization of optimized tree-based ensembles,” *Applied Sciences*, vol. 12, no. 9, Apr. 2022, doi: 10.3390/app12094577.
- [30] H. Alsawalqah *et al.*, “Software defect prediction using heterogeneous ensemble classification based on segmented patterns,” *Applied Sciences*, vol. 10, no. 5, Mar. 2020, doi: 10.3390/app10051745.
- [31] S. R. Aziz, T. Khan, and A. Nadeem, “Experimental validation of inheritance metrics’ impact on software fault prediction,” *IEEE Access*, vol. 7, pp. 85262–85275, 2019, doi: 10.1109/ACCESS.2019.2924040.
- [32] P. R. Bal and S. Kumar, “WR-ELM: weighted regularization extreme learning machine for imbalance learning in software fault prediction,” *IEEE Transactions on Reliability*, vol. 69, no. 4, pp. 1355–1375, Dec. 2020, doi: 10.1109/TR.2020.2996261.
- [33] T. T. Khuat and M. H. Le, “Ensemble learning for software fault prediction problem with imbalanced data,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 4, pp. 3241–3246, Aug. 2019, doi: 10.11591/ijece.v9i4.pp3241-3246.
- [34] O. Nyarko-Boateng, A. F. Adekoya, and B. A. Weyori, “Using machine learning techniques to predict the cost of repairing hard failures in underground fiber optics networks,” *Journal of Big Data*, vol. 7, no. 1, Aug. 2020, doi: 10.1186/s40537-020-00343-4.
- [35] E. Khoudry, A. Belfqih, T. Ouaderhman, J. Boukherouaa, and F. Elmariami, “Real-time fault diagnosis system for high-speed power system protection based on machine learning algorithms,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 6, pp. 6122–6138, Dec. 2020, doi: 10.11591/IJECE.V10I6.PP6122-6138.
- [36] D. G. Lee and Y. S. Seo, “Improving bug report triage performance using artificial intelligence based document generation model,” *Human-centric Computing and Information Sciences*, vol. 10, no. 1, Jun. 2020, doi: 10.1186/s13673-020-00229-7.
- [37] A. Marjuni, T. B. Adji, and R. Ferdiana, “Unsupervised software defect prediction using median absolute deviation threshold based spectral classifier on signed Laplacian matrix,” *Journal of Big Data*, vol. 6, no. 1, Sep. 2019, doi: 10.1186/s40537-019-0250-z.
- [38] S. Nasir, M. Croock, and S. Al-Qaraawi, “Software engineering based fault tolerance model for information system in plants shopping center,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 6, pp. 6664–6672, Dec. 2020, doi: 10.11591/IJECE.V10I6.PP6664-6672.
- [39] S. Voutsinas, D. Karolidis, I. Voyiatzis, and M. Samarakou, “Development of a machine-learning-based method for early fault detection in photovoltaic systems,” *Journal of Engineering and Applied Science*, vol. 70, no. 1, Apr. 2023, doi: 10.1186/s44147-023-00200-0.
- [40] J. Deng, L. Lu, and S. Qiu, “Software defect prediction via LSTM,” *IET Software*, vol. 14, no. 4, pp. 443–450, Aug. 2020, doi: 10.1049/iet-sen.2019.0149.
- [41] H. S. Munir, S. Ren, M. Mustafa, C. N. Siddique, and S. Qayyum, “Attention based GRU-LSTM for software defect prediction,” *PLoS ONE*, vol. 16, no. 3, Mar. 2021, doi: 10.1371/journal.pone.0247444.
- [42] T. Hai, J. Zhou, N. Li, S. K. Jain, S. Agrawal, and I. Ben Dhaou, “Cloud-based bug tracking software defects analysis using deep learning,” *Journal of Cloud Computing*, vol. 11, no. 1, Aug. 2022, doi: 10.1186/s13677-022-00311-8.
- [43] M. Jorayeve, A. Akbulut, C. Catal, and A. Mishra, “Deep learning-based defect prediction for mobile applications,” *Sensors*, vol. 22, no. 13, Jun. 2022, doi: 10.3390/s22134734.
- [44] A. B. Farid, E. M. Fathy, A. S. Eldin, and L. A. Abd-Elmegid, “Software defect prediction using hybrid model (CBIL) of convolutional neural network (CNN) and bidirectional long short-term memory (Bi-LSTM),” *PeerJ Computer Science*, vol. 7, pp. 1–22, Nov. 2021, doi: 10.7717/peerj-cs.739.
- [45] Z. Luo, W. Wang, and C. Cen, “Improving bug localization with effective contrastive learning representation,” *IEEE Access*, vol. 11, pp. 32523–32533, 2023, doi: 10.1109/ACCESS.2022.3228802.
- [46] I. Maduako *et al.*, “Deep learning for component fault detection in electricity transmission lines,” *Journal of Big Data*, vol. 9, no. 1, Jun. 2022, doi: 10.1186/s40537-022-00630-2.
- [47] C. Pan, M. Lu, B. Xu, and H. Gao, “An improved CNN model for within-project software defect prediction,” *Applied Sciences*, vol. 9, no. 10, May 2019, doi: 10.3390/app9102138.
- [48] O. Al Qasem, M. Akour, and M. Alenezi, “The influence of deep learning algorithms factors in software fault prediction,” *IEEE Access*, vol. 8, pp. 63945–63960, 2020, doi: 10.1109/ACCESS.2020.2985290.
- [49] H. Wang, W. Zhuang, and X. Zhang, “Software defect prediction based on gated hierarchical LSTMs,” *IEEE Transactions on Reliability*, vol. 70, no. 2, pp. 711–727, Jun. 2021, doi: 10.1109/TR.2020.3047396.
- [50] W. Yao, M. Shafiq, X. Lin, and X. Yu, “A software defect prediction method based on program semantic feature mining,” *Electronics*, vol. 12, no. 7, Mar. 2023, doi: 10.3390/electronics12071546.
- [51] T. N. Tengku Asmawi, A. Ismail, and J. Shen, “Cloud failure prediction based on traditional machine learning and deep learning,” *Journal of Cloud Computing*, vol. 11, no. 1, Sep. 2022, doi: 10.1186/s13677-022-00327-0.
- [52] E. Borandag, “Software fault prediction using an RNN-based deep learning approach and ensemble machine learning techniques,” *Applied Sciences*, vol. 13, no. 3, Jan. 2023, doi: 10.3390/app13031639.
- [53] A. Khalid, G. Badshah, N. Ayub, M. Shiraz, and M. Ghouse, “Software defect prediction analysis using machine learning techniques,” *Sustainability*, vol. 15, no. 6, Mar. 2023, doi: 10.3390/su15065517.
- [54] W. Zhang, G. Han, J. Wang, and Y. Liu, “A BP neural network prediction model based on dynamic cuckoo search optimization algorithm for industrial equipment fault prediction,” *IEEE Access*, vol. 7, pp. 11736–11746, 2019, doi: 10.1109/ACCESS.2019.2892729.
- [55] 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, doi: 10.1109/ACCESS.2022.3231456.
- [56] Y. Hassouneh, H. Turabieh, T. Thaher, I. Tumar, H. Chantar, and J. Too, “Boosted whale optimization algorithm with natural selection operators for software fault prediction,” *IEEE Access*, vol. 9, pp. 14239–14258, 2021, doi: 10.1109/ACCESS.2021.3052149.
- [57] S. Y. Lee, W. E. Wong, Y. Li, and W. C. C. Chu, “Software fault-proneness analysis based on composite developer-module networks,” *IEEE Access*, vol. 9, pp. 155314–155334, 2021, doi: 10.1109/ACCESS.2021.3128438.
- [58] Y. Li, W. Eric Wong, S. Y. Lee, and F. Wotawa, “Using tri-relation networks for effective software fault-proneness prediction,” *IEEE Access*, vol. 7, pp. 63066–63080, 2019, doi: 10.1109/ACCESS.2019.2916615.
- [59] K. Phung, E. Ogunshile, and M. Aydin, “Error-type - a novel set of software metrics for software fault prediction,” *IEEE Access*, vol. 11, pp. 30562–30574, 2023, doi: 10.1109/ACCESS.2023.3262411.
- [60] I. Tumar, Y. Hassouneh, H. Turabieh, and T. Thaher, “Enhanced binary moth flame optimization as a feature selection algorithm to predict software fault prediction,” *IEEE Access*, vol. 8, pp. 8041–8055, 2020, doi: 10.1109/ACCESS.2020.2964321.





- [61] J. Zhang *et al.*, “CDS: a cross-version software defect prediction model with data selection,” *IEEE Access*, vol. 8, pp. 110059–110072, 2020, doi: 10.1109/ACCESS.2020.3001440.
- [62] D. Bowes, T. Hall, and J. Petrić, “Software defect prediction: do different classifiers find the same defects?,” *Software Quality Journal*, vol. 26, no. 2, pp. 525–552, Feb. 2018, doi: 10.1007/s11219-016-9353-3.
- [63] S. K. Pandey, D. Rathee, and A. K. Tripathi, “Software defect prediction using K-PCA and various kernel-based extreme learning machine: an empirical study,” *IET Software*, vol. 14, no. 7, pp. 768–782, Dec. 2020, doi: 10.1049/iet-sen.2020.0119.
- [64] Y. Chen and H. Dai, “Improving cross-project defect prediction with weighted software modules via transfer learning,” *Journal of Physics: Conference Series*, vol. 2025, no. 1, Sep. 2021, doi: 10.1088/1742-6596/2025/1/012100.
- [65] M. Mustaqeem and M. Saqib, “Principal component based support vector machine (PC-SVM): a hybrid technique for software defect detection,” *Cluster Computing*, vol. 24, no. 3, pp. 2581–2595, Apr. 2021, doi: 10.1007/s10586-021-03282-8.
- [66] B. Marapelli, A. Carie, and S. M. N. Islam, “Software defect prediction using ROS-KPCA stacked generalization model,” in *Smart Innovation, Systems and Technologies*, vol. 326, Springer Nature Singapore, 2023, pp. 587–597.
- [67] H. K. Dam *et al.*, “Lessons learned from using a deep tree-based model for software defect prediction in practice,” in *IEEE International Working Conference on Mining Software Repositories*, May 2019, pp. 46–57, doi: 10.1109/MSR.2019.00017.
- [68] S. Hosseini, B. Turhan, and M. Mäntylä, “A benchmark study on the effectiveness of search-based data selection and feature selection for cross project defect prediction,” *Information and Software Technology*, vol. 95, pp. 296–312, Mar. 2018, doi: 10.1016/j.infsof.2017.06.004.
- [69] L. Sikic, A. S. Kurdija, K. Vladimir, and M. Silic, “Graph neural network for source code defect prediction,” *IEEE Access*, vol. 10, pp. 10402–10415, 2022, doi: 10.1109/ACCESS.2022.3144598.
- [70] J. Li, P. He, J. Zhu, and M. R. Lyu, “Software defect prediction via convolutional neural network,” in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, Jul. 2017, pp. 318–328, doi: 10.1109/QRS.2017.42.
- [71] 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, pp. 152–170, May 2018, doi: 10.1016/j.ins.2018.02.027.
- [72] A. Iqbal *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. 300–308, 2019, doi: 10.14569/ijacsa.2019.0100538.
- [73] J. Pachouly, S. Ahirrao, and K. Kotecha, “SDPTool: a tool for creating datasets and software defect predictions,” *SoftwareX*, vol. 18, Jun. 2022, doi: 10.1016/j.softx.2022.101036.
- [74] Z. M. Zain, S. Sakri, and N. H. A. Ismail, “Application of deep learning in software defect prediction: systematic literature review and meta-analysis,” *Information and Software Technology*, vol. 158, Jun. 2023, doi: 10.1016/j.infsof.2023.107175.
- [75] P. R. Bal, “Cross project software defect prediction using extreme learning machine: an ensemble based study,” in *Proceedings of the 13th International Conference on Software Technologies*, 2018, pp. 354–361, doi: 10.5220/0006886503540361.
- [76] A. Bahaa, E. M. Fathy, A. S. Eldin, and L. A. Abd-Elmegid, “A systematic literature review of software defect prediction using deep learning,” *Journal of Computer Science*, vol. 17, no. 5, pp. 490–510, May 2021, doi: 10.3844/jcssp.2021.490.510.
- [77] I. Batoool and T. A. Khan, “Software fault prediction using deep learning techniques,” *Software Quality Journal*, vol. 31, no. 4, pp. 1241–1280, Jun. 2023, doi: 10.1007/s11219-023-09642-4.
- [78] C. Pan, M. Lu, and B. Xu, “An empirical study on software defect prediction using CodeBERT model,” *Applied Sciences*, vol. 11, no. 11, May 2021, doi: 10.3390/app11114793.
- [79] R. Malhotra and S. Kamal, “An empirical study to investigate oversampling methods for improving software defect prediction using imbalanced data,” *Neurocomputing*, vol. 343, pp. 120–140, May 2019, doi: 10.1016/j.neucom.2018.04.090.
- [80] A. Alsaedi and M. Z. Khan, “Software defect prediction using supervised machine learning and ensemble techniques: a comparative study,” *Journal of Software Engineering and Applications*, vol. 12, no. 5, pp. 85–100, 2019, doi: 10.4236/jsea.2019.125007.

BIOGRAPHIES OF AUTHORS



Deepti Rai     completed her master degree in 2016 and bachelor degree in 2004 from Visvesvaraya Technological University, India. She is currently pursuing her doctoral degree in the domain of machine learning at the Department of Computer Science and Engineering, Ramaiah University of Applied Sciences, Ramaiah Technology Campus, Bengaluru, Karnataka, India. She has 10 years of experience in teaching and 6 years of Industry experience. Her research interest is in the field of machine learning, deep learning, AI, and cloud computing. She can be contacted at email: deeraisecond@gmail.com.



Jyothi Arcot Prashant     completed her Ph.D. in 2020, master degree in 2009, Bachelor degree in 2002 from Visvesvaraya Technological University, India. She is currently working as Faculty of Engineering and Technology, Department of Computer Science and Engineering, Ramaiah University of Applied Sciences, Ramaiah Technology Campus, Bengaluru, Karnataka, India. She has 16 years of experience in teaching and has published many research papers in journals indexed in SCI/SCIE, WOS, SCOPUS, and presented papers in several National and International conferences. Her research interest is in the field of wireless sensor network, IoT, embedded systems, AI, ML and deep learning. She can be contacted at email: jyothiarcotprashant@gmail.com.