Received 12 June 2023, accepted 15 June 2023, date of publication 19 June 2023, date of current version 28 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3287326

RESEARCH ARTICLE

A Novel Approach to Improve Software Defect Prediction Accuracy Using Machine Learning

IQRA MEHMOOD[1], SIDRA SHAHID[1], HAMEED HUSSAIN[2], INAYAT KHAN[3], SHAFIQ AHMAD[4], SHAHID RAHMAN[2], NAJEEB ULLAH[3], (Member, IEEE), AND SHAMSUL HUDA[5]

[1]Department of Computer Science, University of Agriculture Faisalabad, Faisalabad 38040, Pakistan
[2]Department of Computer Science, University of Buner, Buner 19290, Pakistan
[3]Department of Computer Science, University of Engineering and Technology, Mardan 23200, Pakistan
[4]Industrial Engineering Department, College of Engineering, King Saud University, Riyadh 11421, Saudi Arabia
[5]School of Information Technology, Deakin University, Burwood, VIC 3128, Australia

Corresponding author: Inayat Khan (inayatkhan@uetmardan.edu.pk)

This work was supported by King Saud University, Riyadh, Saudi Arabia, through the Researchers Supporting Project under Grant RSP2023R387.

ABSTRACT In software engineering community, defect prediction is one the active domain. For the software's success, it is essential to reduce the software engineering and data-mining gap. Software defects prediction forecasts the source code errors before the testing phase. Methods for predicting software defects, such as clustering, statistical methods, mixed algorithms, metrics based on neural networks, black box testing, white box testing and machine learning are frequently used to explore the effect area in software. The main contribution of this research is the use of feature selection for the first time to increase the accuracy of machine learning classifiers in defects pre-diction. The objective of this study is to improve the defects prediction accuracy in five data sets of NASA namely; CM1, JM1, KC2, KC1, and PC1. These NASA data sets are open to public. In this research, the feature selection technique is use with machine-learning techniques; Random Forest, Logistic Regression, Multilayer Perceptron, Bayesian Net, Rule ZeroR, J48, Lazy IBK, Support Vector Machine, Neural Networks, and Decision Stump to achieve high defect prediction accuracy as compared to without feature selection (WOFS). The research workbench, a machine-learning tool called WEKA (Waikato Environment for Knowledge Analysis), is used to refine da-ta, preprocess data, and apply the mentioned classifiers. To assess statistical analyses, a mini tab statistical tool is used. The results of this study reveals that accuracy of defects prediction with feature selection (WFS) is improve in contrast with the accuracy of WOFS.

INDEX TERMS Defect prediction, accuracy, feature selection, machine learning.

I. INTRODUCTION

In software system, unexpected performance in response to a client's need is known as a defect. Software testers typically notice this unusual behavior in software. Software testers notices errors in the software testing process. The term "software fault" is also use to describe, "irregularities in the software development process that frequently result in soft-ware failure and fall short of user expectations" [1].

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

A defect is a lack of imperfection caused by an error, fault, or failure in the software development process or product. According to the paradigm, "error" refers to human behavior that leads to inappropriate out-comes, and "defect" refers to a decision that leads to incorrect outcomes when trying to solve a problem.

The process of predicting software defects involves the detection of defected modules and a variety of testing requirements. It is extremely difficult in software engineering to design a good defect prediction model, which would predict malfunctioning software modules or software defects in earlier phases of the software development life cycle. Re-viewing the source code, doing beta testing, integration testing, system testing, and unit testing are all steps in the traditional process of finding software errors. Therefore, it be-comes challenging to carry out these tests as software expands in size, complexity, and size of source code [2].

In recent years, software defect prediction has become increasingly popular. Prediction of software defects has a direct impact on software quality. Defective software modules have a significant impact on the quality of the product, which causes price overruns, a delay in the software's completion timeframe, and increased maintenance costs [3].

The first and second fundamental methods of software quality assurance are defect detection and defect prevention. The goal of defect prevention is to stop potential faults as soon as possible. Defect prediction addressing current flaws. According to Memon et al. [1], the process of enhancing software quality through defect prevention is the focus of our research, which aims to increase software quality by forecasting faults. Defect prevention activities include designing the algorithm, reviewing the execution of the algorithm, and identifying errors in the planning of software need [4]. Prior to the software product's deployment process, the primary goal of defect prediction is to predict flaws, errors, or defects in software products to anticipate the deliverable maintenance effort and quality [5]. The defect prevention approach is use to improve software quality [1]. Predicting errors is a crucial step in creating good software. Because software deployment precedes defect prediction to increase overall system performance and ensure user satisfaction. Early detection of errors or faults results in adequate resource allocation, which reduces time and cost while also producing a high-quality output. As a result, software defect prediction models take an active role in helping people learn how to evaluate software and improve its quality [6].

The software-testing phase is more effective having the defect-prediction process, which identifies problematic software modules. Utilizing efficient defect prediction approaches or models, several techniques, and approaches have produced outstanding results. It is crucial to combine an efficient defect prediction model with a successful measurement system [7]. The deployment of high quality, user-satisfying software is possible through the prediction of software defects. Software-quality assurance practices, such as code review is frequently uses for identification of software defect [8]. Numerous methods have been use to overcome issues or concerns with software fault prediction. There are numerous strategies for defect prediction mentioned in the literature study; however, no one method applies to all datasets. Because it depends on the dataset's characteristics. It can be difficult to choose the best method for fault prediction. Machine Learning is the most effective technique for defect prediction [9]. Defect prediction techniques (DPT) used throughout the SDLC in order to prevent such failures in software products [10].

Based on particular machine learning algorithms and data sets, machine learning has given IT systems the ability to recognize different types of patterns with efficient solution. Additionally, the outcomes produced by machine learning are based on prior knowledge of relevant material [11]. Systems now have the potential to learn automatically based on past performance. Machine learning predicts that computers can learn from data or previous knowledge, recognize patterns in the data, and then make judgements with a minimum human intervention. It is an attractive field because it enables you to build on prior knowledge to acquire practical business rule logics and much more. What makes this unique? However, the machine learning process is not straightforward. The value of machine learning in the twenty-first century is that it enables continuous learning from data and future prediction. This is a powerful collection of algorithms and models applied across industries to enhance software operations and discover patterns and abnormalities in data [12].

Machine learning functions similarly to an individual learning approach. As humans, machine learning makes decisions based on knowledge [13]. It is describe as the estimation of a system's hidden structures using minimal prior data. Classification, clustering, and regression are examples of machine learning problems [14]. Utilizing different machine learning patterns, various machine-learning methods can boost software quality and efficiency [5]. Additionally, a larger part in reducing re-work is play by the process of forecasting the software issue or defect early to increase software quality [9].

Software defect prediction using machine learning algorithms has several advantages. It enables organizations to prioritize testing efforts, allocate resources effectively, and make informed decisions about software quality. By identifying high-risk areas early, developers can address potential issues before they impact end-users, resulting in improved customer satisfaction and reduced maintenance efforts. In this research, authors contribute in testing phase to increase the accuracy of machine learning algorithm to better predict the defects for user.

A. CONTRIBUTION

The main contribution of this research is the use of feature selection for the first time to increase the accuracy of machine learning classifiers in defects prediction. The objective of this study is to improve the defects prediction accuracy in five data sets. The machine-learning techniques used in this research are; Random Forest, Logistic Regression, Multi-layer Perceptron, Bayesian Net, Rule ZeroR, J48, Lazy IBK, Support Vector Machine, Neural Networks, and Decision Stump to achieve high defect prediction accuracy as com-pared to WOFS.

B. PAPER ORGANIZATION

The remaining paper is organized as; in section II related work is presented, section III is devoted to proposed work, in section IV results and discussion is presented, and finally conclusion and future work is presented at the end of the paper.

II. LITERATURE REVIEW

The most desirable study field is defect prediction via machine learning, data metrics, and other methods. Different approaches have provided various models and interpretations. There have been numerous studies published on the analysis of software fault pre-diction from 1990 to 2022 [15], [16]. Size and complexity metrics for defect prediction, were developed by Benton and Neil in 1999. Software size and software complexity metrics were discuss in the defect prediction process. Using software metrics, processing high-quality data, multivariate methods, and a critique of existing methods, defect prediction is carried out. According to their calculations, each thousands of lines of code (KLOC) contains about 23 flaws.

For defect prediction, Vanmali et al. [17] used machine learning (ML) approaches. To predict the fault, they used neural networks. Similar comparisons between Neural Net-work and other approaches were uses, and it was conclude that Neural Network outperformed other methodologies in terms of error detection. They also discussed the application of various ML techniques. They use the PROMISE data set and hypothesized that the best indicators of programming are responses to classes, LOC, and the absence of good coding. Additionally, they are engaged in comparative research on ensemble approaches for software best practices.

Biçer et al. [18] investigated the situation in which it is impractical to survey thoroughly every component of complicated frameworks. They examined numerous tactics for their stages and described the qualities of good conformity indicators. They assembled their analyses with regard to static code measures and discovered that these flaw identifiers produce results that are consistent across a wide range of applications, are cost-effective to use, and can be adjusted to the point of interest of current business conditions. They took into account realistic conditions for evaluating programming expenses and agreed that a better evaluation allowed for tenfold greater financial gains. They demonstrate how quality pointers can be found early on in the process of product improvement by utilizing reliable measurements and ML approaches.

To categorize various datasets related to liver patients, Ramana et al. [19] examined a few selected machine learning classification techniques. Using two datasets, the effective-ness of a few machine learning classification algorithms was assessed. The first dataset included records for 751 liver patients from Andhra Pradesh in India with 12 attributes. The University of California, Irvine (UCI) Machine Learning Repository provided the second dataset, which included 345 records with five attributes. With a core-i7 processor and 4 GB of RAM, the WEKA data mining open source machine learning tool or workbench was utilized for the trials. Here, the Naive Bayes classifier, K-Nearest Neighbor Algorithm, Back Propagation Neural Network Algorithm, C4.5, and Support Vector Machines were taken into consideration as machine learning classification techniques. Accuracy, Specificity, Sensitivity, and Precision were the four criteria used by these algorithms to evaluate the outcomes. K-Nearest Neighbor Algorithm, Back propagation, and Support Vector Ma-chines provide superior results with all feature set combinations when utilizing a chosen dataset.

In order to predict software defects, Gray et al. [20] created models that took into ac-count the quality of the datasets, which were noisy and contained missing values that might affect the outcomes. The researcher concluded that the influence of quality relies on the dataset while building a model and predicting defects, where data cleansing could be a major factor.

Askari and Bardsiri [21] applied artificial neural networks for defect prediction. For effective extension machine-learning algorithms, evolutionary approaches were combine with SVM learning for prediction. Machine learning models using NASA Datasets were used to test the support vector approach. They concluded that, when accuracy and precision are combine, SVM learning outperformed other methods in terms of accuracy and precision.

Prasad et al. [22] talked about various ML classification techniques, including super-vised (Bayesian Network, Ensemble Method/Random Forests, SVM, Decision Tree), unsupervised (j48, Random forest, Naive Bayes classifier, k-mean clustering, Hierarchical Clustering), and semi-supervised (Low-density separation, SVM, expectation-maximization, and class mass normalization) methods that are The best classification strategy to forecast the defect for a high quality software.

ChandraYadav et al. [23] presented the defect detection algorithm employing classification, association rule, and clustering approaches. The authors explains the Knowledge Discovery in Database (KDD) process and explains how to uncover potential outcomes by applying patterns and extracting errors or faults. The authors concluded that finding bugs or defects might be possible with little testing equipment.

In Kumar and Shukla [24] the early detection of defects was accomplish using the fuzzy logic information system. At the function level, the proposed model's fuzzy inference system uses metrics data and other error data. This model has five input variables that are utilize in one input layer and one output layer to determine if the input layer data for defects.

Mandal and Ami [25] examined how software attributes depends on quality, performance, and effectiveness to defect prediction models. Several software qualities were used to predict software module malfunctioning. In defect prediction, if right attributes are not select, the model's performance will suffer. Therefore, it is crucial to choose the right features to create a useful prediction model in order to enhance the efficiency and performance of defect prediction. In order to identify software with flaws or to approach the right model, the researcher presented an attribute selection technique.

The research findings demonstrates that the described strategy offered a comparably effective collection of features that improved the performance, quality, and efficiency of the model. In order to improve the outcome of software defect prediction, one ML classifier was use. For further improvement of the suggested approach, researchers hope to incorporate the performance of many ML classifiers in the future.

Hammouri et al. [9] addressed the defects prediction model. In order to forecast defects, supervised machine learning techniques such as Naive Bayes, Artificial Neural Networks, confusion matrices, and decision tree algorithms were apply to various datasets. Three debugging datasets were use in the experiment. The aspects of the experimental outcomes were recall, precision, RMSE measurements, F-measure, and accuracy. In addition, these experimental results demonstrate that machine learning is superior to other approaches, such as the POWM model and AR model, in terms of results and pre-diction model performance.

Memon et al. [1] evaluates the methods for predicting software errors and mitigating their effects on the production of high-quality software. They discusses several defect prediction mechanisms (based on pattern, graph mining ASA using Classifier) and prevention mechanisms through defect detection, defect analysis, and its importance to minimize the causes of system failure using the most recent technology. Additionally, they describes the advantages and disadvantages of certain systems for the creation of high-quality products.

In a study Li et al. [26] included 2456 experimental findings, 49 articles that satisfied the inclusion criteria which were published from January 2000 to March 2018. Matthew's Correlation Coefficient (MCC) terms were used to calculate the comparison prediction performance over studies in a consistent manner, with confusion matrices serving as the basis. The researchers, for the effectiveness of unsupervised defect prediction algorithms, performed a meta-analysis. Recalculating the confusion matrices from the primary experiments to get improved performance in order to compare these results in a more credible manner. Numerous frequently used outfit-learning calculations, like stacking calculations, had been widely used. In this way, the subsequent stage of work inquiry mostly entails attempting to examine the benefits and drawbacks of another ensemble learning, and then utilizing other different base classifier blends in the vote calculation. Though there was a concerning amount of inadequate reporting, undemanding benchmark datasets, and plainly incorrect experimental outcomes.

Zhong et al. [55] begins by highlighting the importance of time-series event prediction in various domains, such as finance, weather forecasting, and anomaly detection. It also emphasizes the limitations of existing prediction methods and the need for more effective techniques. The proposed method introduces a sequence labeling framework, which models the time-series data as a sequence of discrete events. The framework consists of three main components: data pre-processing, feature extraction, and sequence labelling. In the data-preprocessing step, the time-series data is pre-processed to handle missing values, noise, and other data quality issues. This ensures that the subsequent steps operate on clean and reliable data. The paper presents a new method for time-series event prediction based on sequence labeling. The approach combines data preprocessing, feature extraction, and sequence labeling techniques to accurately predict events in time-series data. The experimental results highlight the effectiveness and robustness of the proposed method, suggesting its potential for applications in various domains requiring accurate event prediction in time-series data. Further research and exploration of this method could lead to advancements in time-series analysis and prediction tasks.

Mario et al. 2023 [56] introduces a novel approach for updating REM models. By leveraging clustering and Random Forest techniques, the proposed methodology addresses the challenges of updating REM models effectively. The experimental results indicate its superiority over traditional methods, offering promising avenues for future research and practical applications in the field of monitoring and data analysis. Author's addresses the challenge of updating REM models effectively, as system dynamics and data distribution may change over time. The proposed methodology aims to enhance the accuracy and adaptability of REM models by incorporating new data while preserving the efficiency of the monitoring process. To achieve this goal, the authors propose a two-step approach. In the first step, the data points are clustered into distinct groups based on their similarity. In the second step, a Random Forest model is trained on each cluster to update the corresponding REM model. By training separate models for each cluster, the proposed methodology can capture the unique patterns and dynamics within each subset. The implications of this research are significant for various applications that rely on REM models. By improving the accuracy and adaptability of REM models, organizations can enhance their monitoring and decision-making processes, leading to more efficient resource allocation, anomaly detection, and predictive analytics.

Defect prediction approaches: Pavana et al. [57] provides an overview of machine learning algorithms and their applicability in software fault prediction. They explain that machine-learning algorithms can learn from historical data and extract patterns, allowing them to make predictions on unseen data. This makes them suitable for analyzing software metrics and identifying potential fault-prone areas. The findings of Sekaran, Kripa [58] research have important implications for software development practices. The deep learning model presented in the paper offers a promising solution for defect prediction in intra-project software, enabling developers to detect and rectify potential issues at an early stage. This can ultimately lead to improved software quality and reduced maintenance costs. Ruchika Malhotra et al. [59]

research presents the results of the experiments conducted to evaluate the performance of different ML classifiers in predicting software faults. The authors compare the accuracy, precision, recall, and F-measure of the ML models against traditional prediction models, such as the linear regression model. They also analyze the impact of different metrics on the prediction accuracy. The results indicate that ML classifiers, including decision trees, support vector machines, and random forests, outperform the traditional models, demonstrating their potential in software fault prediction.

Chen et al. [60] proposes an approach and evaluate it on several benchmark datasets commonly used in software-defect-prediction research. The experimental results demonstrated that the nested-stacking architecture combined with heterogeneous feature selection outperformed other state-of-the-art techniques. The approach showed significant improvements in terms of accuracy, precision, recall, and F1-score, indicating its effectiveness in predicting software defects. The findings of Bahaweres et al. [61] study suggest that using SMOTE in combination with neural networks can enhance software defect prediction. The approach presented in the paper has the potential to be applied in real-world software development scenarios, helping developers identify and address potential defects early in the development process, leading to higher software quality. The experimental results demonstrate that the neural network-based SMOTE approach effectively improves the accuracy of software defect prediction. The authors compare their approach with other conventional techniques and find that their proposed method outperforms them in terms of prediction accuracy and other evaluation metrics. The Qiu et al. [62] focuses on the problem of feature exploration in defect prediction, which is a critical task in software engineering to identify potential defects in software systems. The authors propose an automatic feature exploration approach to address this challenge. The proposed approach utilizes a combination of machine learning techniques and statistical analysis to automatically explore and select relevant features for defect prediction. The authors employ three feature selection methods: correlation analysis, t-test, and information gain, to identify the most significant features from a large set of candidate features. Qiao and Wang [63] emphasizes the importance of defect prediction in software development, as it helps identify and address potential issues before they become critical problems. Traditional defect prediction approaches often overlook the effort required to fix defects, leading to suboptimal allocation of development resources. To address this limitation, the authors propose an effort-aware defect prediction model that incorporates the effort factor into the prediction process. They utilize a feedforward neural network architecture, which is a type of artificial neural network known for its ability to learn complex patterns and make accurate predictions.

The prominent difference that our research work is different from the aforementioned software-defect-prediction approaches is that they are missing the feature selection and the datasets we are using are different and updated. Therefore, the summarized literature review on feature selection is presented in Table 1. It is clear from the given approaches that the work presented in this paper is different from these. However, the only related research that resembles with our research is of Karabulut et al. [68]. The key differences between [68] and this research are; their work [68] is focusing the classification accuracy not the software-defect-prediction accuracy. The data sets they are using are different and there are numerous differences in the classifiers used in both researches.

III. PROPOSED METHODOLOGY

To address challenges or problems in software defect prediction, various strategies were plan. There are numerous strategies for defect prediction mentioned in the literature; however, no single method can be apply to all datasets. Because it depends on the dataset's characteristics. It is difficult to choose which method may be use for fault prediction. Machine Learning is the most trustworthy technique for fault prediction [9]. Utilizing a machine learning technique, feature-selection comprises assessing greater accuracy among the input and the target variable. In this research, WEKA, a machine-learning tool used for feature selection and two-tail t testing for statistical analysis of the results used in mini-tab.

A. MECHANISM

Feature selection is a practical method for addressing a wide range of problems by removing unnecessary or duplicate data [27]. The main objective of selecting features is to enhance accuracy-based prediction performance to give faster and affordable predictions within a timespan [28]. In account of prediction, which variable has to be use? Which as-pect ought to be incorporate into a successful predictive model? It will be challenging to provide an answer because it relies on the domain and the nature of the da-tasets. Through feature selection, it is possible to select variables automatically from da-taset that are closely relate to the output, accuracy, and problem you are working on. As compared to filter-base-feature-selection, machine learning makes it more difficult to choose accurate statistical measures for diverse types of datasets. In such circumstances, deleting or reducing the irrelevant or non-important features from datasets can help us identify related features or appropriate characteristics that significantly contribute to our goal variable and improve the accuracy of our research.

The accuracy or performance that may be improved, is greatly influenced by the data features utilized to train machine learning models. The performance of a prediction model might adversely affect by irrelevant or only partially relevant features. For the purpose, de-signing an effective model, data set cleansing and feature selection must be on top priori-ties. The main justifications for using feature selection are;

TABLE 1. Summarized literature review on feature selection.

Ref No.	Research Paper Title	Author	Published Year	Approach Used
[64]	Minimum Redundancy Feature Selection (MRFS) from Microarray Gene Expression Data	Ding C. et al	2003	MRFS algorithm as an effective feature selection method for microarray gene expression data.
[65]	A Correlation-Based Feature Selection Algorithm for Machine Learning	Hall, Mark A.	1999	Introduces the CFS (Correlation-Based Feature Selection) algorithm, which addresses the limitations of existing feature selection methods by considering both relevance and redundancy.
[66]	Fast Correlation-Based Filter for Wrapper Approaches	Lei Yu and Huan Liu	2003	Presents a novel feature selection algorithm that combines the advantages of correlation-based filters and wrapper approaches to improve the efficiency and effectiveness of feature selection in machine learning tasks.
[67]	A Relief-Based Feature Selection Method for Classification Problems	Robnik-Šikonja and Kononenko	2003	Introduces the Relief algorithm as an effective feature selection method for classification tasks. By considering both the relevance and quality of features, Relief identifies informative and discriminative features that lead to improved classification accuracy.
[68]	A Comparative Study on the Effect of Feature Selection on Classification Accuracy	Esra Mahsereci Karabulut et al.	2012	The research paper provides a detailed comparative analysis of feature selection techniques and their impact on classification accuracy. The data sets used were Audiology, Breast Cancer, Lung cancer, vowel and Zoo.
[69]	Approaches to Multi-Objective Feature Selection: A Systematic Literature Review	Qasem Al-Tashi et al.	2020	The objective of the paper is to provide a systematic literature review on various approaches to multi-objective feature selection. Feature selection is a crucial step in machine learning and data analysis, as it aims to identify the most relevant and informative features from a given dataset. Multi-objective feature selection extends this concept by considering multiple conflicting objectives, such as maximizing classification accuracy while minimizing the number of selected features.
[70]	Feature Selection Approaches for Machine Learning Classifiers on Yearly Credit Scoring Data	Ilter Fakhouri, Damla & Kocadağlı, Ozan & Ravishanker	2019	The paper focuses on the problem of credit scoring, which involves predicting the creditworthiness of individuals based on various features. In this study, the authors investigate different feature selection approaches for machine learning classifiers using yearly credit scoring data.
[71]	Efficient Feature Selection for Intrusion Detection Systems	Seyedeh Sareh Ahmadi et al.	2019	The main focus of this study is to address the challenge of efficient feature selection in the context of intrusion detection systems (IDS). Intrusion detection systems play a crucial role in safeguarding computer networks from unauthorized access and potential attacks. However, with the increasing complexity and diversity of network data, it becomes essential to identify the most relevant features that can effectively detect intrusions while minimizing computational overhead.

TABLE 1. *(Continued.)* **Summarized literature review on feature selection.**

[72]	Intrusion Detection using Machine Learning and Feature Selection	Heena Malhotra et al.	2019	The main focus of the article is on the application of machine learning and feature selection techniques for intrusion detection in computer networks. Intrusion detection is an important aspect of network security, as it involves identifying and preventing unauthorized access, attacks, and malicious activities within a network.
[73]	Feature Selection and Ensemble-Based Intrusion Detection System: An Efficient and Comprehensive Approach	Ebrima Jaw et al.	2021	The paper begins by highlighting the increasing importance of intrusion detection systems (IDS) in protecting computer networks from various attacks. Traditional IDSs often suffer from high false alarm rates and low detection accuracy due to the complexity and dynamic nature of network traffic. To overcome these limitations, the authors propose a two-step approach that combines feature selection and ensemble learning.
[74]	A Comparative Study of Feature Selection Approaches :2016-2020	Syed Asim Ali Shah Hafiz Muhammad Shabbir	2020	The study compares and evaluates the performance of several popular feature selection techniques that were proposed and utilized in the field between 2016 and 2020. These techniques include filter methods, wrapper methods, and embedded methods. Filter methods evaluate the features independently of the learning algorithm, wrapper methods assess the feature subsets based on the predictive performance of a specific learning algorithm, and embedded methods incorporate feature selection as part of the learning algorithm itself.
[75]	Majority Voting and Feature Selection Based Network Intrusion Detection System	Dharmaraj R. Patil and Tareek M. Pattewar	2022	Presents a novel approach to network intrusion detection using majority voting and feature selection techniques. The authors aim to improve the accuracy and efficiency of intrusion detection systems (IDS) by incorporating ensemble learning and feature selection methods.

- Train Algorithm: the feature enables ML algorithms to train datasets more quickly and accurately while staying within budget.
- Reduce Complexity: by removing complexity from the trained model, it will be simpler to review and interpret the results.
- Less duplicate trained data equals less opportunity to base decisions on noisy data, which reduces overfitting.
- Increases accuracy: accuracy will be increases by reducing irrelevant or false data. If an accurate subset is selected, accuracy will increase.
- Training time reduction: compared to a large dataset, the trained data set is smaller. As a result, training datasets take less time.

Extract key variables from data, including labelled, unlabeled, and incomplete data, using the feature selection framework [29]. By analyzing subsets of data as indicated in the following Figure 1, feature selection is use to identify redundant and irrelevant fea-tures, highlighted [30] as a result. There may be a key variable available to carry out vari-ous machine-learning activities. The self-explanatory feature selection framework is depict in the following Figure 1.

B. DATASETS

For findings of the research, Five benchmark datasets from the PROMISE collection are made available to the general public, including, JM1, CM1, KC1, PC1, and KC2 [31]. Details about each dataset is described in depth in the following Table 2 including, faulty instances, non-faulty instances, number of attributes and the percentage of faulty instances as shown below.

TABLE 2. Characteristics of datasets used.

Name of Dataset	Instances	Attributes	Missing Attributes	Non-Faulty instances	Defective instances %	Faulty Instances	Used Language
PC1	1109	22	NO	1032	6.94%	77	NA
KC1	2109	22	NO	1783	15.45%	326	C++
JM1	10885	22	NO	8779	19.35%	2106	C
CM1	498	22	NO	449	9.83	49	C
KC2	522	22	NO	415	20.49%	107	C++

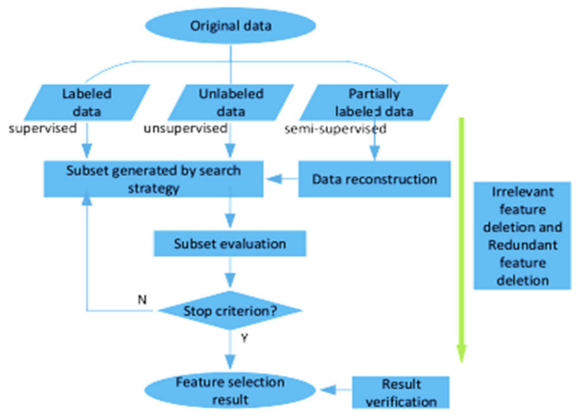


FIGURE 1. Feature selection framework.

C. SOFTWARE/TOOLS USED

1) WEKA

The University of Waikato in New Zealand has developed WEKA (Waikato Environment for Knowledge Analysis), a key AI workbench implemented in Java. WEKA is open-source software that is free and downloadable under the GNU General Public Li-cense. For running machine-learning algorithms, the WEKA platform is ideal. Without having to worry about programming languages or mathematical issues, it enables a graphical user interface that is simple to use [52]. AI calculations can be apply to diverse datasets with different dialects or call from your own Java code. The AI workbench WEKA includes a variety of image processing tools and other calculations for information analysis and vision display. The goal of WEKA, a machine-learning workbench, is to apply multiple machine learning algorithms on a wide diversity of real-problems [32]. WEKA offers 49 tools for preprocessing data, 76 algorithms for classification and regression, 8 for clustering, 3 for identifying association rules, 15 attribute/subset evaluators, and 10 search methods for feature selection. Several file formats. WEKA has a number of benefits, including; GNU-licensed free access for everyone. It supports a wide range of contemporary computing platforms. In WEKA, a vast data preparation available using machine learning modelling approaches.

2) MINI-TAB

One of the best providers of statistical analysis for quality improvement is mini tab software. It is use to summarize data in a graphical manner or to apply statistical techniques to obtain results with a user-friendly interface. Many graphs are available to show your data in a presentable manner using probability and probability distribution, and it updates automatically as your data changes. This application is useful for exploring data analysis using graphs, advanced charts, and other visual aids [33]. In mini tab, numerous analyses are there, including Z tests, t tests, Poisson rate testing, normality tests, and correlation and covariance analyses.

D. ALGORITHMS/MECHANISMS

1) LOGISTIC REGRESSION

Probability predictions are possible using the logistic regression technique. It is use to illustrate the likelihood of a particular class, such as pass/fail, lose/win, sound/wiped out or dead/alive. This might be expand to show a limited types of situations, such as deter-mining whether a photo has a cat, dog, lion, etc. Each item in the image that can be identify by giving a probability between zero and one, with the sum equaling one. Although augmentations that are far more complex exist, it is a statistical approach that is use to represent a binary (zero and one) dependent variable for logistical purposes [34].

2) BAYES NET

It is use to assess the probabilistic graphical model that Bayesian inference use for computing probabilities. By displaying the conditions dependent of edges created in a direct graph, the Bayes Network identified model condition dependence. Researchers can combine probability distributions using Bayes Net to improve compact variable factorization and gain the benefits of conditional independence. Bayes networks are used for variety of tasks, including prediction, anomaly detection, making decisions under uncertain conditions, and time series prediction [35].

3) MULTILAYER PERCEPTRON (MLP)

It is a type of ANN, a chain of perceptron's, and a system of linear classifier. The word "MLP" is use, occasionally loosely to refer to any feedforward ANN and occasionally to refer to systems built up of various layers of perceptron. MLP are occasionally refer to as "vanilla" neural systems informally, especially when they have a single secreted layer. It has three levels of nodes: the first is the input layer, the second is the hidden or unseen layer, and the third is the output layer [53].

4) J48

Ross Quinlan's build decision tree using J48. J48 is the WEKA project team's implementation of the Iterative Dichotomiser 4 algorithm. For WEKA, A data-mining tool that uses C4.5 algorithms is J48. J48 is develop in Java and is open source for making decisions. It generates tree-constructed data output based on input value. Ross Quinlan was the one who created this theorem [36]. Both discrete and continuous features are handle. It is suitable for error forecasting in data sets of various sizes. This regulation built on a classifier that learns as a tree structure, with each hub acting as either a leaf or a node.

5) DECISION STUMP

ML algorithm with a single-level decision tree is decision stumps. Predictions from a decision stump model based on a single input variable or feature. It is a well-organized grouping of if-then statements, which might be more straightforward and thus more logical than a decision tree. It is less complicated and requires less computation than the decision tree technique. It is the simplest ML approach. It presents the data set with a DT that includes a comparable numeral of the original data set's attributes [37].

6) SUPPORT VECTOR MACHINE

SVM is a type of supervised learning that can be apply to a variety of problems, including classification and regression. It synchronizes up with a certain observation or variable. In machine learning (ML), SVMs are applied learning models with associated learning calculations that dissect data used for regression analysis and classification research. With many training models, each set apart as having a spot with either of two classifications. A SVM model is a representation of the models as points in space that are map to ensure that instances of the various classes are segregate by a logically anticipated. Then, new models are map into that comparable space with the expectation that they will fit into one of several classes depending on which side of the hole they fall into [38], [54].

7) RANDOM FOREST

A classification random forest algorithm used in data science. A combination of DT known as random forest presents self-sufficiently certain regulated change. It includes a variety of variables, and the result is based on the class's most precise yield (output). Every tree has a separate sample bootstrap and each root node has data or information that is equivalent to the real data. Using the factor or variable that is randomly selected from the input factor or variable's split technique. Afterwards, each tree is developed to its fullest potential without pruning. When all trees are used in the forest technique, fresh occurrences are connected to each tree, and a voting process takes place to select the arrangement that receives the most votes as the initial instance expectation [39].

8) LAZY IBK

IBK is a core subset of the classification algorithm family. Lazy IBK (instance Based Learner) is the name of the KNN algorithm used in WEKA. IBK is use to make predictions for test instances in the short term rather than to generate a model. To find the k "closest" instances and create a prediction, distance is measured. It differs from other Instance-based (IB) learners in that it uses a separation function based on entropy [40]. By comparing an instance to a database of pre-grouped models, it classifies the instances [41].

9) RULE ZeroR

No algorithm fails accurate outcomes based on a rule depend on a single property if the targeted distribution of data is constrained and skewed for forecasting the majority class, using the Rule ZeroR method with nominal data types [42]. Whenever Rule ZeroR evaluates using training data, it always exceeds the baseline. Performance on independent test data may not accurately reflects in training data.

IV. RESULTS

In order to analyze the defect prediction performance by incorporating feature selection for accuracy improvement, this research use publically available benchmark datasets from NASA promise repository [31] namely CM1, PC1, JM1, KC1, KC2 and five classifiers namely Lazy IBK, Bayes Net, Rule ZeroR, Multilayer Perceptron and J48. The aforementioned datasets used by various researchers; the PC1 dataset is use in 2009 by Kaur et al. [43] In 2009 and 2019 the Kaur et al. and Jayanthi and Florence [44] respectively uses the JM1 dataset. Aleem et al. [45], Khanum et al. [46], uses the CM1 dataset for their researches. The KC1dataset is use by Manjula and Florence [47], Al-Nusirat et al. [48], Jayanthi and Florence [44]). The KC2 dataset is use in [31] and [49].

First, we compares the defects prediction accuracy based on the mentioned datasets using the specified classifiers with feature selection (WFS) and without feature selection (WOFS) in Figure 2. The Figure 2 to Figure 5 are based on the results obtained in Table 3. In Figure 2, most of the classifiers

TABLE 3. Defect prediction accuracy comparison of WFS and WOFS.

Classifier	Data Set	Accuracy WFS	Accuracy WOFS
Multilayer Perceptron	JM1	81.08%	80.95%
	CM1	88.95%	88.35%
	KC1	85.53%	85.91%
	KC2	83.71%	83.33%
	PC1	93.77%	93.05%
Bayes Net	JM1	75.77%	68.05%
	CM1	75.50%	66.26%
	KC1	75.43%	70.27%
	KC2	80.84%	78.92%
	PC1	91.70%	93.32%
Lazy IBK	JM1	76.38%	76.97%
	CM1	83.93%	86.14%
	KC1	84.25%	84.11%
	KC2	79.88%	81.03%
	PC1	91.88%	92.06%
Rule ZeroR	JM1	80.69%	80.65%
	CM1	90.16%	90.16%
	KC1	84.54%	84.54%
	KC2	79.50%	78.92%
	PC1	93.05%	93.59%
J48	JM1	81.11%	79.50%
	CM1	89.55%	88.75%
	KC1	84.35%	83.78%
	KC2	83.71%	81.99%
	PC1	93.32%	74.39%

outperforms on the specified data sets WFS as compared to WOFS for defect prediction accuracy. As clearly depicted in Figure 2 (d), all the classifiers improve the defects prediction accuracy on KC2 data set except by Lazy IBK classifier be-cause it is an entropy base. In majority results obtained through the data sets in Figure 2 (a), (b), (c), (d) and (e) the defect prediction accuracy WFS is better than WOFS while in few cases the WOFS performs slightly better than WFS due to some intrinsic problems in datasets. For better precision of the results, we apply 30-folds cross-validation. Generally, we can say that through feature selection, on average, one can improve the overall defect pre-diction accuracy by 5%.

We plots the defects prediction accuracy on the mentioned classifiers using the aforementioned data sets with feature selection (WFS) and without feature selection (WOFS) in Figure 3. It is clearly noticeable in

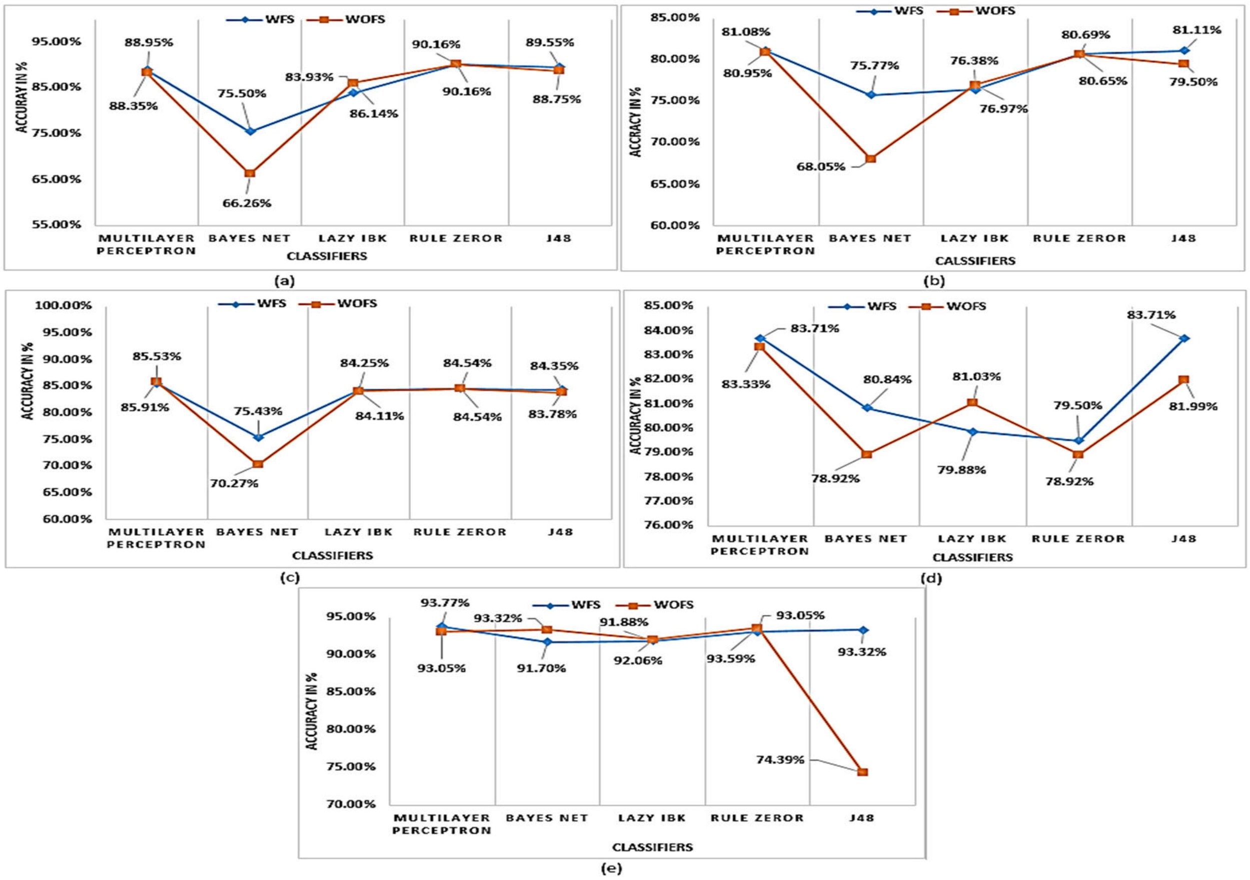


FIGURE 2. **Defects prediction accuracy based on data sets using various classifiers (a) Accuracy on CM1; WFS and WOFS (b) Accuracy on JM1; WFS and WOFS (c) Accuracy on KC1; WFS and WOFS (d) Accuracy on KC2; WFS and WOFS (e) Accuracy on and WOFS PC1; WFS.**

Figure 3 (a), (b), (c), (d) and (e) that the defect prediction accuracy WFS through the various classifiers is better than WOFS in majority cases while in few cases the WOFS performs equally or slightly better than WFS due to some intrinsic problems in datasets. As depicted in Figure 3(c) and 3(e) respectively for Bayes Net and J48 classifiers, the defect prediction accuracy through WFS outperforms than WOFS. While the accuracy through Lazy IBK in Figure 3(b) WFS is not impressive as compared to WOFS due to its entropy based nature.

The following Table 3 shows comparison of the aforementioned classifiers based on NASA promise datasets on WFS and WOFS basis. The following Figure 4 and Figure 5 based on Table 3 results. It is clearly apparent from the results in Table 3, Figure 4 and Figure 5 that the accuracy through WFS outperforms as compared to WOFS. In the datasets results comparisons, the KC2 results are best for WFS as compared to others datasets results, while the J48 classifier leads the race amongst classifiers, as obvious from the mentioned table and figures. For better precision of the results, 30-folds cross-validation is applied. Generally, we can say that through feature selection, on average, one can improve the overall defect prediction accuracy by 5%.

A. COMPARISON WITH EXISTING APPROACH

In this section we compares the proposed WFS results with existing counterpart [50] results WOFS. The comparison is on four mentioned datasets JM1, CM1, KC2 and PC1. The authors also compares the mentioned approaches on four classifiers namely, logistic regression, decision stump, random forest and support vector machine (SVM). For better precision of the results, 30-folds cross-validation is applied. All the mentioned datasets and classifiers defect predictions accuracy is in Table 4. Among the datasets the PC1 defect prediction accuracy WFS beats the other datasets results, while in the classifiers, over-all the logistic regression outperforms among the others classifiers. The WOFS accuracy is taken from the Alsaeedi and Khan, [50] paper and authors named it as AK approach (WOFS).

As clear from the following Figure 6 and Figure 7, the prediction accuracy on classifiers WFS using mentioned datasets beats the AK approach (WOFS) almost at all. This shows the supremacy of our proposed approach. In Figure 6, the individual classifier result comparison is shown in (a), (b), (c) and (d), while Figure 7 depicts the same results in bar chart.

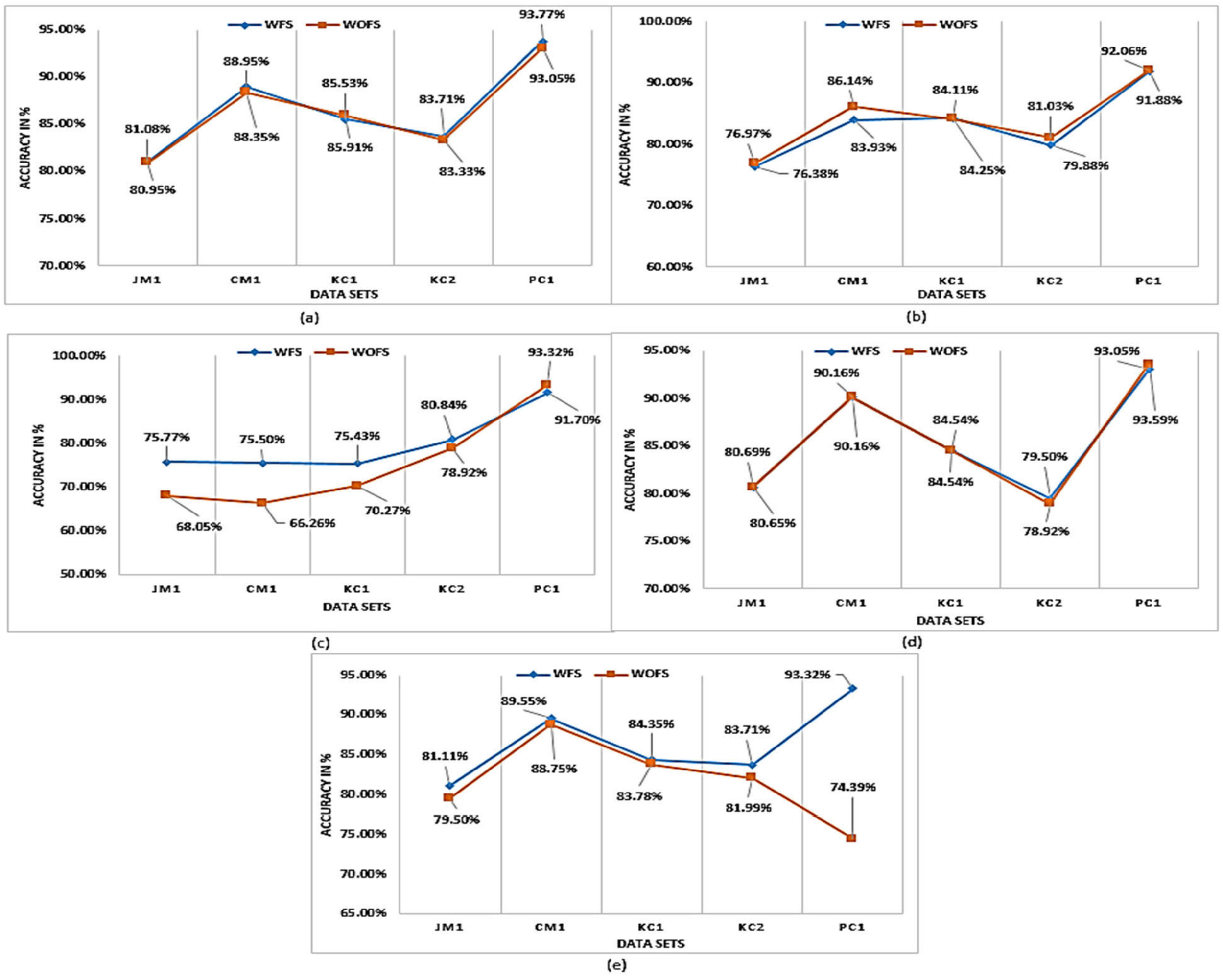


FIGURE 3. Defect prediction accuracy based on classifiers using various data sets (a) Accuracy of Multilayer Perceptron; WFS and WOFS (b) Accuracy of Lazy IBK; WFS and WOFS (c) Accuracy of Bayes Net; WFS and WOFS (d) Accuracy of Rule ZeroR; WFS and WOFS (e) Accuracy of J48; WFS and WOFS.

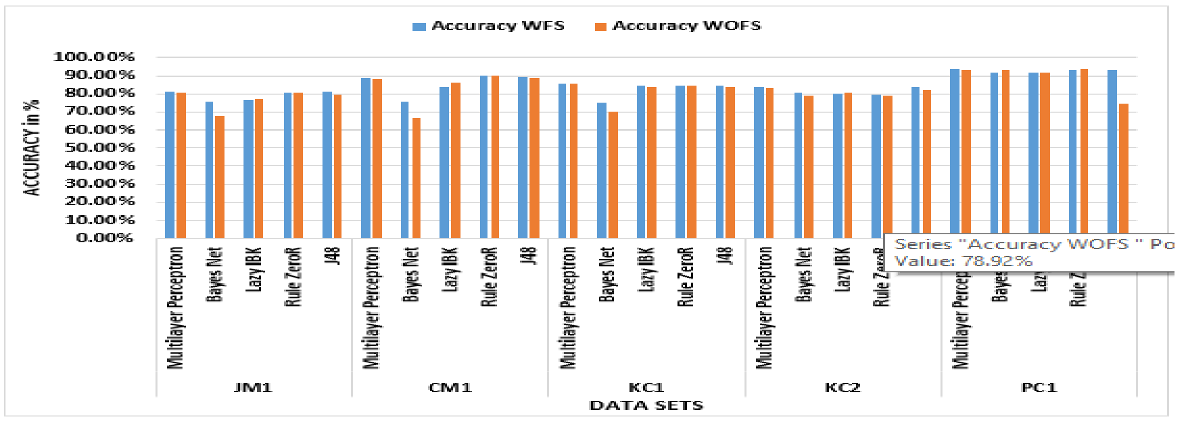


FIGURE 4. Defect prediction datasets accuracy on various classifiers WFS and WOFS.

The following Figure 8 and Figure 9 shows the prediction accuracy on datasets WFS using mentioned classifiers beats the AK approach (WOFS). The results shows that our proposed approach beats the AK approach (WOFS). In Figure 8,

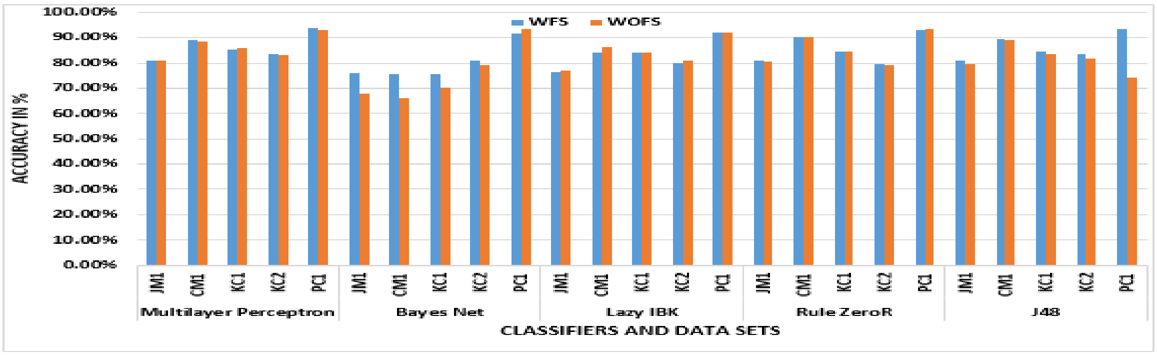


FIGURE 5. Defect prediction classifiers accuracy on various datasets WFS and WOFS.

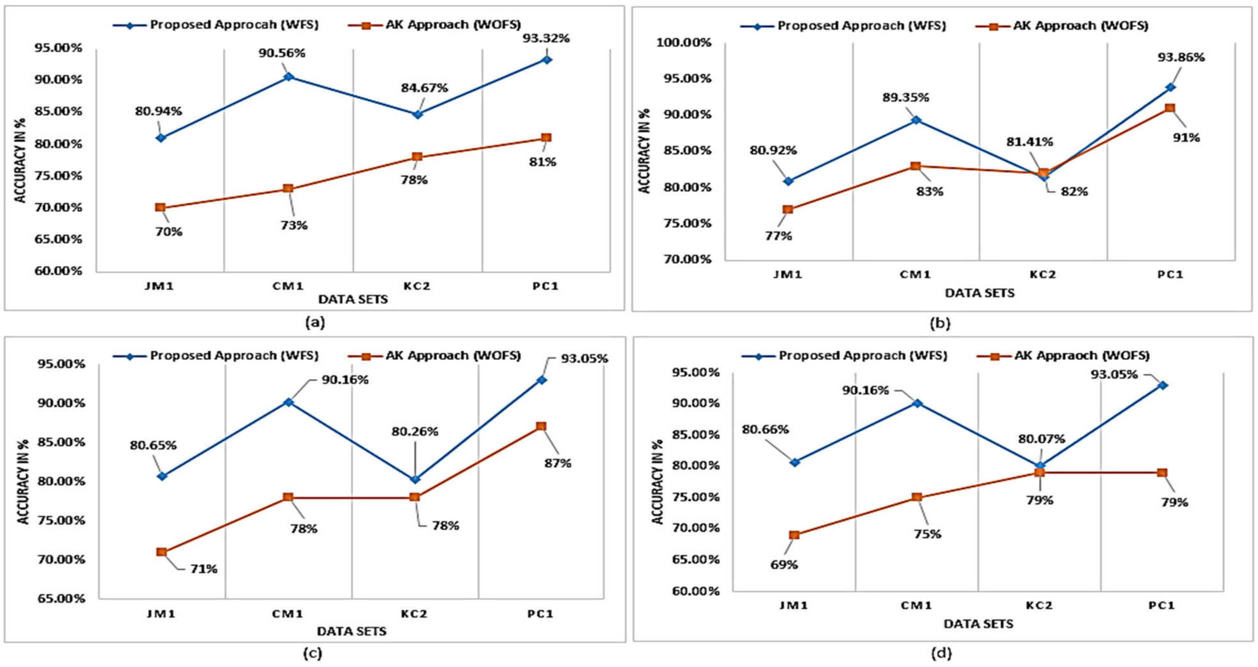


FIGURE 6. Results comparison on Classifiers basis using various data sets (a) Logistic Regression; Proposed (WFS) and AK (WOFS) (b) Random Forest; Proposed (WFS) and AK (WOFS) (c) Decision Stump; Proposed (WFS) and AK (WOFS) (d) Support Vector Machine; Proposed (WFS) and AK (WOFS).

the individual dataset results comparison as depicted in (a), (b), (c) and (d), while Figure 9 depicts the same results in bar chart.

B. ANALYSIS OF PROPOSED APPROACH

For results accuracy proof, Minitab is used for two tail T-test. Minitab screenshot of the result is depicted in Figure 10. Two variables WFS and WOFS were used for two tail testing.

To execute the test the two variables WFS and WOFS accuracy, the hypothesis are generated manually. For statistical evaluation, paired t-test is applied.

Test

H0 : Accuracy WOFS = Accuracy WFS (Null hypothesis)

H1 : Accuracy WOFS $<$ Accuracy WFS (Alternative hypothesis)

According to statistical approach, If 0.05 $<$ P-value it will be accepted, otherwise rejected. The graph of the p-value is depicted in Figure 11.

H0 is rejected due to its p-value of 0.000. As per statistical decision, the defect prediction accuracy WFS is higher than WOFS. By applying the paired T-test statistical [51] approach, It's obviously evidence that accuracy with WFS surpasses the WOFS.

V. DISCUSSION

Software defect prediction using machine learning algorithms is a field of research and practice aimed at identifying and predicting potential defects or bugs in software systems. By leveraging the power of machine learning, organizations

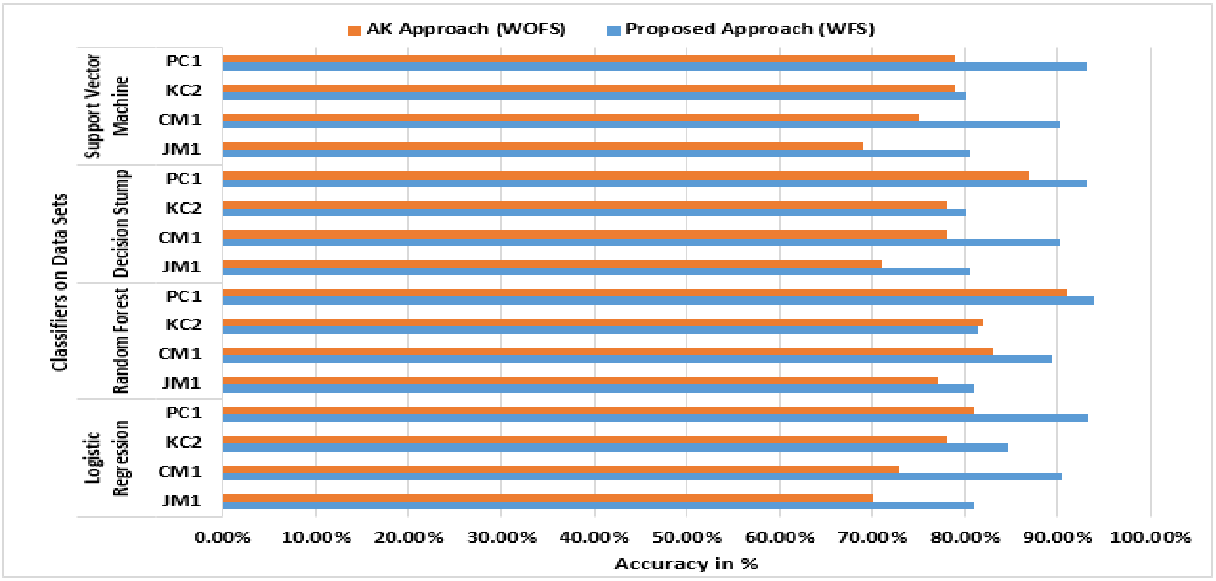


FIGURE 7. Comparison with Alsaeedi and Khan [50] results on classifiers basis using various data sets.

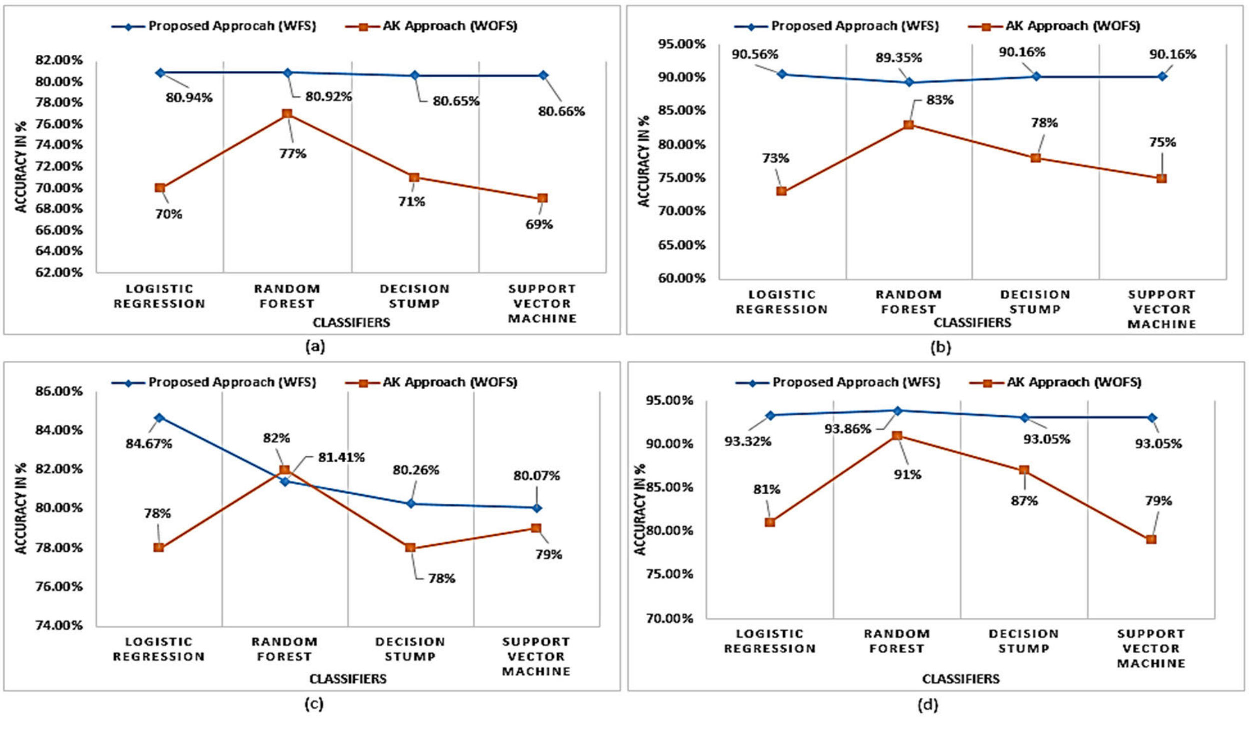


FIGURE 8. Results comparison on data sets basis using various classifiers (a) JM1; Proposed (WFS) and AK (WOFS) (b) CM1; Proposed (WFS) and AK (WOFS) (c) KC2; Proposed (WFS) and AK (WOFS) (d) PC1; Proposed (WFS) and AK (WOFS).

can proactively address and mitigate software issues before they become critical.

The process of software defect prediction involves collecting historical data from software projects, including information about code attributes, development metrics, and defect reports. This data is used to train machine learning algorithms, enabling them to learn patterns and relationships between software characteristics and the occurrence of defects.

Various machine learning algorithms applied In this research for software defect prediction, including: Random Forest, Logistic Regression, Multi-layer Perceptron, Bayesian Net, Rule ZeroR, J48, Lazy IBK, Support Vector Machine, Neural Networks, and Decision Stump. These

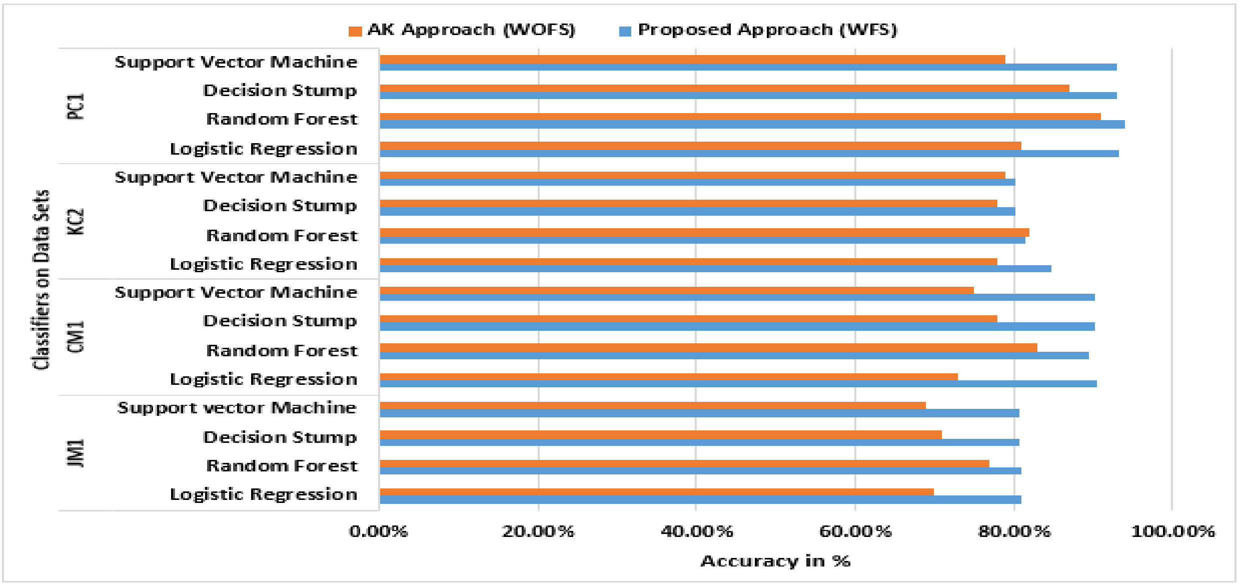
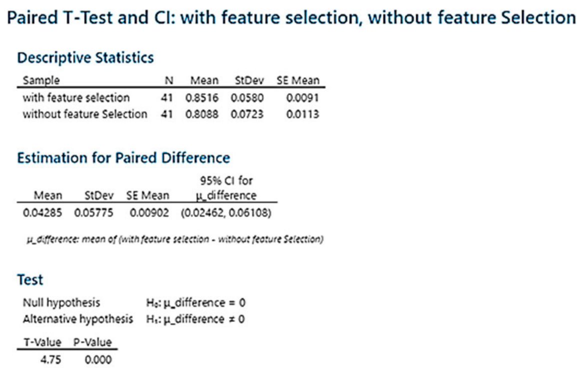


FIGURE 9. Comparison with Alsaeedi and Khan [50] results on data set basis using various classifiers.


Paired T-Test and CI: with feature selection, without feature Selection

Descriptive Statistics

Sample	N	Mean	StDev	SE Mean
with feature selection	41	0.8516	0.0580	0.0091
without feature Selection	41	0.8088	0.0723	0.0113

Estimation for Paired Difference

Mean	StDev	SE Mean	95% CI for µ_difference
0.04285	0.05775	0.00902	(0.02462, 0.06108)

µ_difference: mean of (with feature selection - without feature Selection)

Test

Null hypothesis	H_0: µ_difference = 0
Alternative hypothesis	H_1: µ_difference ≠ 0

T-Value	P-Value
4.75	0.000

FIGURE 10. T-Test using Minitab.

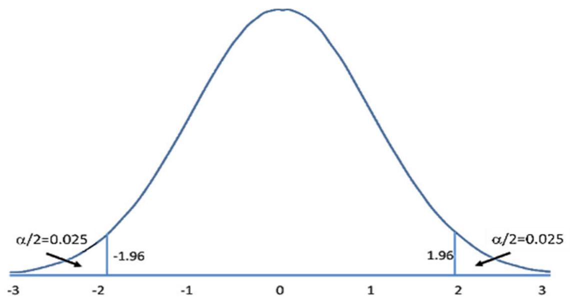


FIGURE 11. T-Test Result.

algorithms use different techniques to analyze the collected data and build predictive models that can classify new instances as either defect-prone or defect-free.

Feature selection and engineering play a crucial role in software defect prediction. Relevant features, such as complexity metrics, code churn, and code dependencies, are extracted from the software codebase. Additionally, domain-specific knowledge can be incorporated to enhance the accuracy of the predictive models.

In this research, software defect prediction process performed through feature selection using above mentioned algorithms.

Once the machine learning model is trained, it will apply to new software projects or code changes to predict the likelihood of defects. By identifying potentially problematic areas in advance, developers and testers will focus their efforts on critical sections of code and allocate resources efficiently. This proactive approach helps in reducing the overall software maintenance costs and improving software quality.

Software defect prediction using machine learning algorithms has several advantages. It enables organizations to prioritize testing efforts, allocate resources effectively, and make informed decisions about software quality. By identifying high-risk areas early, developers will address potential issues before they impact end-users, resulting in improved customer satisfaction and reduced maintenance efforts.

The accuracy and effectiveness of predictive models heavily rely on the quality and representativeness of the training data, and this accuracy is improved in this research. Additionally, the dynamic nature of software development introduces challenges in maintaining up-to-date models and adapting them to changing project characteristics.

In conclusion, software defect prediction using machine learning algorithms provides a valuable tool for software development and quality assurance teams. By harnessing the power of data and machine learning, organizations can proactively identify and address software defects, resulting in improved software reliability and customer satisfaction.

TABLE 4. Results comparison with AK results (WOFS).

Classifiers	Data Set	AK Approach (WOFS)	Proposed Approach (WFS)
Logistic Regression	JM1	70%	80.94%
	CM1	73%	90.56%
	KC2	78%	84.67%
	PC1	81%	93.32%
Random Forest	JM1	77%	80.92%
	CM1	83%	89.35%
	KC2	82%	81.41%
	PC1	91%	93.86%
Decision Stump	JM1	71%	80.65%
	CM1	78%	90.16%
	KC2	78%	80.26%
	PC1	87%	93.05%
Support Vector Machine	JM1	69%	80.66%
	CM1	75%	90.16%
	KC2	79%	80.07%
	PC1	79%	93.05%

VI. CONCLUSION AND FUTURE WORK

Software defect prediction is one of the active research domain in software engineering. Prior to testing, software defect prediction predicts flaws in source codes. To Box testing, system testing, and unit testing are the traditional methods for finding defect. As a result, it becomes challenging to carry out these tests when a project expands examine defects in software, several techniques such as classification, clustering, mixed algorithms, data mining statistical methods, neural networks, and machine learning are widely employed. To address challenges or problems with software defect prediction, various re-search methodologies have been plan. There are numerous methods used for predicting software defects, but no method exists that works for every dataset. Considering this, it is dependent on the main data in the dataset. It can be difficult to decide which method should be use for software prediction. Finding errors in the source code is the process of software defect prediction. Code review, Beta testing, Black Box testing, Integrated testing, White in size and complexity of source code. Defect detection and fixing become increasingly challenging. Models for Software defects help with these kinds of issues.

In the century of technological advancement, software systems are getting increasingly complex. Therefore, it is crucial to look for flaws. A product may be release in substandard quality if the proper method for finding software flaws is not use. The most crucial aspects of software are its quality and reliability, and defect prediction is a key indicator of both of these factors. In order to improve accuracy of software defects predictions, this study analyses five NASA data sets: JM1, CM1, KC1, KC2, and PC1. To implement feature selection and achieve the best level of accuracy, machine-learning techniques including Bayesian Net, Logistic Regression, Multilayer Perceptron, Ruler ZeroR, J48, Lazy IBK, Support Vector Machine, Neural Networks, Random Forest, and Decision Stump are applied. Feature selection technique used with the WEKA machine-learning workbench on the aforementioned datasets. With feature selection, the Bayesian net algorithm's accuracy rate increases by an average of 8% whereas the Logistic Regression algorithm's best accuracy is more than 93%. As a future work, research may reveal any further methods to obtain high accuracy, and additional datasets may be test to increase the precision rate. Further research on the effects of various metaheuristic feature selection techniques to choose the best set of characteristics could be an interesting expansion. Although data imbalance is still a problem that adversely affects performance, one future goal is to investigate and compare the performance of deep learning algorithms and ensemble classifiers with various resampling strategies.

REFERENCES

[1] M. A. Memon, M.-U.-R. Magsi, M. Memon, and S. Hyder, "Defects prediction and prevention approaches for quality software development," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 8, pp. 451–457, 2018.

[2] M. Gayathri and A. Sudha, "Software defect prediction system using multilayer perceptron neural network with data mining," *Int. J. Recent Technol. Eng.*, vol. 3, no. 2, pp. 2277–3878, 2014.

[3] R. Malhotra, L. Bahl, S. Sehgal, and P. Priya, "Empirical comparison of machine learning algorithms for bug prediction in open source software," in *Proc. Int. Conf. Big Data Anal. Comput. Intell. (ICBDAC)*, Andhra Pradesh, India, 2017, pp. 40–45, doi: 10.1109/ICBDACI.2017.8070806.

[4] M. S. Rawat and S. K. Dubey, "Software defect prediction models for quality improvement: A literature study," *Int. J. Comput. Sci. Issues*, vol. 9, pp. 288–296, Jan. 2012.

[5] I. Singh, "A survey? Data mining techniques in software engineering," *Int. J. Res. IT, Manage. Eng.*, vol. 6, no. 3, pp. 30–34, Mar. 2016.

[6] N. Kalaivani and R. Beena, "Overview of software defect prediction using machine learning algorithms," *Int. J. Pure Appl. Math.*, vol. 118, pp. 3863–3873, Feb. 2018.

[7] M. Dhiauddin and S. Ibrahim, "A prediction model for system testing defects using regression analysis," *Int. J. Soft Comput. Softw. Eng.*, vol. 2, no. 7, pp. 55–68, Jul. 2012.

[8] R. Malhotra and A. Jain, "Fault prediction using statistical and machine learning methods for improving software quality," *J. Inf. Process. Syst.*, vol. 8, no. 2, pp. 241–262, Jun. 2012.

[9] A. Hammouri, M. Hammad, M. Alnabhan, and F. Alsarayrah, "Software bug prediction using machine learning approach," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 2, pp. 78–83, 2018.

[10] M. M. A. Abdallah and M. M. Alrifaee, "Towards a new framework of program quality measurement based on programming language standards," *Int. J. Eng. Technol.*, vol. 7, pp. 1–3, Oct. 2018.

[11] I. H. Sarkar, "Machine learning: Algorithms, real-world applications and research directions," *SN Comput. Sci.*, vol. 2, p. 160, 2021, doi: 10.1007/s42979-021-00592-x.

[12] P. Langley and J. G. Carbonell, "Approaches to machine learning," *J. Amer. Soc. Inf. Sci.*, vol. 35, no. 5, pp. 306–316, 1984.

[13] T. G. Dietterich, "Machine learning in ecosystem informatics and sustainability," in *Proc. 21st Int. Joint Conf. Artif. Intell.*, 2009, pp. 1–5.

[14] S. E. E. Profile, "Machine learning of hybrid classification models," in *Proc. Impact Internet Bus. Activities Serbia Worldwide*, 2014, pp. 318–323.

[15] A. Chug and S. Dhall, "Software defect prediction using supervised learning algorithm and unsupervised learning algorithm," in *Proc. 4th Int. Conf. Confluence Next Gener. Inf. Technol. Summit*, Noida, 2013, pp. 173–179, doi: 10.1049/cp.2013.2313.

[16] A. Shanthini, "Applying machine learning for fault prediction using software metrics," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 2, pp. 274–278, Jan. 2012.

[17] M. Vanmali, M. Last, and A. Kandel, "Using a neural network in the software testing process," *Int. J. Intell. Syst.*, vol. 17, no. 1, pp. 45–62, 2002.

[18] S. Biçer, A. B. Bener, and B. Çağlayan, "Defect prediction using social network analysis on issue repositories," in *Proc. Int. Conf. Softw. Syst. Process*, May 2011, pp. 63–71.

[19] B. Venkata Ramana, M. S. P. Babu, and N. B. Venkateswarlu, "A critical study of selected classification algorithms for liver disease diagnosis," *Int. J. Database Manage. Syst.*, vol. 3, no. 2, pp. 101–114, May 2011.

[20] D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson, "Reflections on the NASA MDP data sets," *IET Softw.*, vol. 6, no. 6, pp. 549–558, Dec. 2012.

[21] M. M. Askari and V. K. Bardsiri, "Software defect prediction using a high performance neural network," *Int. J. Softw. Eng. Appl.*, vol. 8, pp. 177–188, Jan. 2014.

[22] M. C. M. Prasad, L. F. Florence, and A. Arya, "A study on software metrics based software defect prediction using data mining and machine learning techniques," *Int. J. Database Theory Appl.*, vol. 8, no. 3, pp. 179–190, Jun. 2015.

[23] D. ChandraYadav and S. Pal, "Software bug detection using data mining," *Int. J. Comput. Appl.*, vol. 115, no. 15, pp. 21–25, Apr. 2015.

[24] D. Kumar and V. H. S. Shukla, "A defect prediction model for software product based on ANFIS," *Int. J. Sci. Res. Devices* vol. 3, no. 10, pp. 1024–1028, 2016.

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

[26] N. Li, M. Shepperd, and Y. Guo, "A systematic review of unsupervised learning techniques for software defect prediction," *Inf. Softw. Technol.*, vol. 122, Jun. 2020, Art. no. 106287.

[27] J. Cai, J. Luo, S. Wang, and S. Yang, "Feature selection in machine learning: A new perspective," *Neurocomputing*, vol. 300, pp. 70–79, Jul. 2018.

[28] K. P. Singh, N. Basant, and S. Gupta, "Support vector machines in water quality management," *Analytica Chim. Acta*, vol. 703, no. 2, pp. 152–162, Oct. 2011.

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

[30] S. P. Potharaju and M. Sreedevi, "A novel subset feature selection framework for increasing the classification performance of SONAR targets," *Proc. Comput. Sci.*, vol. 125, pp. 902–909, Jan. 2018.

[31] J. Petrić, D. Bowes, T. Hall, B. Christianson, and N. Baddoo, "The jinx on the NASA software defect data sets," in *Proc. 20th Int. Conf. Eval. Assessment Softw. Eng.*, Jun. 2016, pp. 1–12.

[32] J. A. Wass, "WEKA machine learning workbench," *Sci. Comput.*, vol. 24, pp. 1–4, Jan. 2007.

[33] A. Ahmad, "Use of minitab statistical analysis software in engineering technology," in *Proc. ASEE Annu. Conf. Expo.*, 2019, pp. 1–10.

[34] L. Niu, "A review of the application of logistic regression in educational research: Common issues, implications, and suggestions," *Educ. Rev.* vol. 72, no. 1, pp. 1–27, 2018.

[35] R. R. Bouckaert, "Bayesian network classifiers in weka for version 3-5-7," *Artif. Intell. Tools*, vol. 11, no. 3, pp. 369–387, 2008.

[36] G. Kaur and A. Chhabra, "Improved J48 classification algorithm for the prediction of diabetes," *Int. J. Comput. Appl.*, vol. 98, no. 22, pp. 13–17, Jul. 2014.

[37] J. P. G. Sterbenz et al., "Redundancy, diversity, and connectivity to achieve multilevel network resilience, survivability, and disruption tolerance invited paper," *Telecommun. Syst.*, vol. 56, pp. 17–31, 2014, doi: 10.1007/s11235-013-9816-9.

[38] P. W. Wang and C. J. Lin, "Support vector machines," *IEEE Intell. Syst. Appl.*, vol. 13, no. 4, pp. 18–28, Jul. 2014.

[39] L. Breiman, "Random Forests," *Mach. Learn.*, vol. 45, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.

[40] S. Mwanjelemwagha, M. Muthoni, and P. Ochieng, "Comparison of nearest neighbor (IBK), regression by discretization and isotonic regression classification algorithms for precipitation classes prediction," *Int. J. Comput. Appl.*, vol. 96, no. 21, pp. 44–48, Jun. 2014.

[41] J. Novakovic and A. S. S. V. Ilic, "Experimental study of using the K-nearest neighbour classifier with filter methods," in *Proc. Conf. Comput. Sci. Technol.*, 2016, pp. 91–99.

[42] L. Devasena, "Effectiveness analysis of ZeroR, RIDOR and PART classifiers for credit risk appraisal," *Int. J. Adv. Comput. Sci. Technol.*, vol. 3, pp. 6–11, May 2014.

[43] A. Kaur, P. S. Sandhu, and A. S. Bra, "Early software fault prediction using real time defect data," in *Proc. 2nd Int. Conf. Mach. Vis.*, 2009, pp. 242–245.

[44] R. Jayanthi and L. Florence, "Software defect prediction techniques using metrics based on neural network classifier," *Cluster Comput.*, vol. 22, no. S1, pp. 77–88, Jan. 2019.

[45] S. Aleem, L. F. Capretz, and F. Ahmed, "Benchmarking machine learning techniques for software defect detection," *Int. J. Softw. Eng. Appl.*, vol. 6, no. 3, pp. 11–23, May 2015.

[46] M. Khanum, T. Mahboob, W. Imtiaz, H. A. Ghafoor, and R. Sehar, "A survey on unsupervised machine learning algorithms for automation, classification and maintenance," *Int. J. Comput. Appl.*, vol. 119, no. 13, pp. 34–39, Jun. 2015.

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

[48] A. Al-Nusirat, F. Hanandeh, M. K. Kharabsheh, M. Al-Ayyoub, and N. Al-Dhfairi, "Dynamic detection of software defects using supervised learning techniques," *Int. J. Commun. Netw. Inf. Secur.*, vol. 11, no. 1, pp. 185–191, Apr. 2022.

[49] E. Naresh and S. P. Shankar, "Comparative analysis of the various data mining techniques for defect prediction using the NASA MDP datasets for better quality of the software product," *Adv. Comput. Sci. Technol.*, vol. 10, no. 7, pp. 2005–2017, 2017.

[50] A. Alsaeedi and M. Z. Khan, "Software defect prediction using supervised machine learning and ensemble techniques: A comparative study," *J. Softw. Eng. Appl.*, vol. 12, no. 5, pp. 85–100, 2019.

[51] A. Potochnik, M. Colombo, and C. Wright, "Statistics and probability," *Recipes Sci.*, vol. 10, pp. 167–206, Aug. 2018, doi: 10.4324/9781315686875-6.

[52] M. Liu et al., "DIG: A turnkey library for diving into graph deep learning research," *J. Mach. Learn. Res.*, vol. 22, no. 1, pp. 10873–10881, 2021.

[53] J. Isabona, A. L. Imoize, S. Ojo, O. Karunwi, Y. Kim, C.-C. Lee, and C.-T. Li, "Development of a multilayer perceptron neural network for optimal predictive modeling in urban microcellular radio environments," *Appl. Sci.*, vol. 12, no. 11, p. 5713, Jun. 2022, doi: 10.3390/app12115713.

[54] J. M. Álvarez-Alvarado, J. G. Ríos-Moreno, S. A. Obregón-Biosca, G. Ronquillo-Lomelí, E. Ventura-Ramos, and M. Trejo-Perea, "Hybrid techniques to predict solar radiation using support vector machine and search optimization algorithms: A review," *Appl. Sci.*, vol. 11, no. 3, p. 1044, Jan. 2021.

[55] Z. Zhong, S. Lv, and K. Shi, "A new method of time-series event prediction based on sequence labeling," *Appl. Sci.*, vol. 13, no. 9, p. 5329, Apr. 2023, doi: 10.3390/app13095329.

[56] M. R. Camana, C. E. Garcia, T. Hwang, and I. Koo, "A REM update methodology based on clustering and random forest," *Appl. Sci.*, vol. 13, no. 9, p. 5362, Apr. 2023, doi: 10.3390/app13095362.

[57] M. Pavana, L. Pushpa, and A. Parkavi, "Software fault prediction using machine learning algorithms," in *Proc. Int. Conf. Adv. Elect. Comput. Technol.*, 2022, pp. 185–197, doi: 10.1007/978-981-19-1111-8_16.

[58] K. Sekaran and L. Annabel, "A deep learning based model for defect prediction in intra-project software," in *Proc. 7th Int. Conf. Trends Electron. Informat.*, 2023, pp. 1148–1155, doi: 10.1109/ICOEI56765.2023.10126014.

[59] R. Malhotra and Y. Singh, "On the applicability of machine learning techniques for object-oriented software fault prediction," *Softw. Eng., Int. J.*, vol. 1, no. 1, pp. 24–37, 2011.

[60] L.-Q. Chen, C. Wang, and S.-L. Song, "Software defect prediction based on nested-stacking and heterogeneous feature selection," *Complex Intell. Syst.*, vol. 8, no. 4, pp. 3333–3348, Aug. 2022, doi: 10.1007/s40747-022-00676-y.

[61] R. Bahaweres, F. Agustian, I. Hermadi, A. Suroso, and Y. Arkeman, ''Software defect prediction using neural network based SMOTE,'' in *Proc. 7th Int. Conf. Electr. Eng., Comput. Sci. Informat. (EECSI)*, Oct. 2020, pp. 71–76, doi: 10.23919/EECSI50503.2020.9251874.

[62] Y. Qiu, Y. Liu, A. Liu, J. Zhu, and J. Xu, ''Automatic feature exploration and an application in defect prediction,'' *IEEE Access*, vol. 7, pp. 112097–112112, 2019, doi: 10.1109/ACCESS.2019.2934530.

[63] L. Qiao and Y. Wang, ''Effort-aware and just-in-time defect prediction with neural network,'' *PLoS ONE*, vol. 14, Jan. 2019, Art. no. e0211359.

[64] C. Ding and H. Peng, ''Minimum redundancy feature selection from microarray gene expression data,'' *J. Bioinf. Comput. Biol.*, vol. 10, no. 2, pp. 185–205, Apr. 2005, doi: 10.1142/s0219720005001004.

[65] M. A. Hall, ''Correlation-based feature selection for machine learning,'' Ph.D. dissertation, Univ. Waikato, 1999.

[66] R. J. Urbanowicz, M. Meeker, W. La Cava, R. S. Olson, and J. H. Moore, ''Relief-based feature selection: Introduction and review,'' *J. Biomed. Informat.*, vol. 85, pp. 189–203, Sep. 2018.

[67] I. Kononenko, ''A relief-based feature selection method for classification problems,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 1, pp. 131–136, Jan. 2003, doi: 10.1109/TPAMI.2003.1159942.

[68] E. M. Karabulut, S. A. Özel, and T. Ibrikçi, ''A comparative study on the effect of feature selection on classification accuracy,'' *Proc. Technol.*, vol. 1, pp. 323–327, Jan. 2012.

[69] Q. Al-Tashi, S. J. Abdulkadir, H. M. Rais, S. Mirjalili, and H. Alhussian, ''Approaches to multi-objective feature selection: A systematic literature review,'' *IEEE Access*, vol. 8, pp. 125076–125096, 2020, doi: 10.1109/ACCESS.2020.3007291.

[70] D. Ilter, O. Kocadagli, and N. Ravishanker, ''Feature selection approaches for machine learning classifiers on yearly credit scoring data,'' in *Recent Advances in Data Science and Business Analytics*. Istanbul, Turkey: Mmar Snan Fne Arts University Department of Statstcs, Fındıklı Campus, Sep. 2019. [Online]. Available: http://ybis2019.msgsu.edu.tr/y-BIS

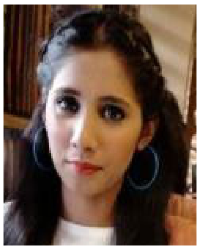
[71] S. S. Ahmadi, S. Rashad, and H. Elgazzar, ''Efficient feature selection for intrusion detection systems,'' in *Proc. IEEE 10th Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf.*, Oct. 2019, pp. 1029–1034, doi: 10.1109/UEMCON47517.2019.8992960.

[72] H. Malhotra and P. Sharma, ''Intrusion detection using machine learning and feature selection,'' *Int. J. Comput. Netw. Inf. Secur.*, vol. 11, no. 4, pp. 43–52, Apr. 2019, doi: 10.5815/ijcnis.2019.04.06.

[73] E. Jaw and X. Wang, ''Feature selection and ensemble-based intrusion detection system: An efficient and comprehensive approach,'' *Symmetry*, vol. 13, no. 10, p. 1764, Sep. 2021, doi: 10.3390/sym13101764.

[74] S. A. Syed, H. M. Shabbir, S. Rehman, and M. Waqas, ''A comparative study of feature selection approaches: 2016–2020,'' *Int. J. Sci. Eng. Res.*, vol. 11, no. 2, p. 469, 2020.

[75] C. Chen, Y. Song, S. Yue, X. Xu, L. Zhou, Q. Lv, and L. Yang, ''FCNN-SE: An intrusion detection model based on a fusion CNN and stacked ensemble,'' *Appl. Sci.*, vol. 12, no. 17, p. 8601, Aug. 2022.



IQRA MEHMOOD is pursuing the M.S. degree in software engineering from Agriculture University, Faisalabad, Pakistan. She is also serving as a digital marketing trainer in National Vocational and Technical Training Commission. She has more than three years of teaching and industry experience. She is the author of several international publications. Her research interest include software defect prediction and optimization, software engineering, machine learning, data mining, artificial intelligence, and big data.

SIDRA SHAHID received the M.S. degree in computer science. She is currently a Lecturer with the Department of Computer Science, University of Agriculture Faisalabad, Pakistan. She has more than eight years of teaching and research experience. She is the author of several national as well as international publications. Her research interests include machine learning, software engineering, data mining, data warehousing, and web mining.

HAMEED HUSSAIN received the bachelor's degree in information technology from Gomal University, Dera Ismail Khan, Pakistan, in 2007, and the M.S. and Ph.D. degrees in computer science from the COMSATS Institute of Information Technology (CIIT), Pakistan, in 2009 and 2017, respectively. He is an Active Researcher. He is the author of several international publications. His research interests include optimization, machine learning, fog and edge computing, the Internet of Things, algorithms real-time systems, resource allocation, and load balancing in high-performance computing.

INAYAT KHAN received the Ph.D. degree in computer science from the Department of Computer Science, University of Peshawar, Pakistan. His current research is based on the design and development of context-aware adaptive user interfaces for minimizing drivers' distractions. His research interests include lifelogging, healthcare, deep learning, ubiquitous computing, accessibility, and mobile-based assistive systems for people with special needs. He has published several papers in international journals and conferences in these areas.

SHAFIQ AHMAD received the Ph.D. degree from RMIT University, Melbourne, Australia. He has more than two decades of working experience both in industry and academia in Australia, Europe, and Asia. He is currently an Associate Professor with the Industrial Engineering Department, College of Engineering, King Saud University, Riyadh, Saudi Arabia. He has published a research book and several research papers in international journals and refereed conferences. His research interests include smart manufacturing, the IIoT, data analytics, multivariate statistical quality control, process monitoring and performance analysis, operations research models, and bibliometric network analysis. He is also a certified practitioner in the Six Sigma business improvement model.

SHAHID RAHMAN received the bachelor's degree in mathematics, physics and computer science from the University of Swat, Khyber Pakhtunkhwa, Pakistan, the M.S. degree in computer science from The University of Agriculture Peshawar (AUP), and the M.Sc. degree in computer science from Islamia College University, Peshawar. He is currently pursuing the Ph.D. degree in computer science with the Qurtuba University of Science and Technology, Peshawar. He has over eight years of experience in academia and research. He is a Lecturer with the Department of Computer Science, University of Buner, Khyber Pakhtunkhwa, Pakistan. He has published several research articles in leading journals and conferences. His current research interests include software engineering, cryptography, steganalysis and steganography, computer vision, machine learning, and deep learning.

NAJEEB ULLAH (Member, IEEE) received the M.S. degree in computer engineering and the Ph.D. degree in computer engineering from the Polytechnic University of Turin, Italy. He is currently an Assistant Professor with the Department of Computer Science, University of Engineering and Technology, Mardan, Khyber Pakhtunkhwa, Pakistan. He has published many research papers in peer-reviewed reputed international journals and conference proceedings. His research interests include software reliability, software process improvement, and empirical software engineering. He is also a reviewer of renowned journals.

SHAMSUL HUDA received the Ph.D. degree in computer science from the Centre for Informatics and Applied Optimization (CIAO), Federation University Australia. He is currently a Lecturer with the School of Information Technology, Deakin University, Australia. Before joining Deakin University, he was an Academician with Federation University and an Assistant Professor with the Khulna University of Engineering and Technology (KUET), Bangladesh. He is a Certified Information System Security Professional (CISSP) by the International Information System Security Certification Consortium, $(ISC)^2$. He is also a member of the Cyber Security Research and Innovation Centre (CSRI), Deakin University. He is involved in many international cyber security projects, including cybersecurity capacity maturity for nations with the Oceania Cyber Security Centre (OCSC), Melbourne, in partnership with the Global Cyber Security Capacity Centre (GCSCC), University of Oxford. He has published more than 60 journals and conference papers in well-reputed journals, including IEEE Transactions. His main research interests include communication and network security, strategies for secure operations for industrial control systems (SCADA) and critical infrastructure, intelligent countermeasure for threats against mobile systems, detection of data breaches through the darknet, the IoT security, malware analysis and detection, reverse engineering for endpoint security, and malware analysis and detection for SCADA systems.