Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.037933

Article

Data and Ensemble Machine Learning Fusion Based Intelligent Software Defect Prediction System

Sagheer Abbas[1], Shabib Aftab[1,2], Muhammad Adnan Khan[3,4], Taher M. Ghazal[5,6], Hussam Al Hamadi[7] and Chan Yeob Yeun[8,*]

[1]School of Computer Science, National College of Business Administration & Economics, Lahore, 54000, Pakistan
[2]Department of Computer Science, Virtual University of Pakistan, Lahore, 54000, Pakistan
[3]Department of Software, Faculty of Artificial Intelligence and Software, Gachon University, Seongnam, 13120, Korea
[4]Riphah School of Computing & Innovation, Faculty of Computing, Riphah International University, Lahore Campus, Lahore, 54000, Pakistan
[5]School of Information Technology, Skyline University College, University City Sharjah, Sharjah, UAE
[6]Center for Cyber Security, Faculty of Information Science and Technology, UKM, Bangi, Selangor, 43600, Malaysia
[7]College of Engineering and IT, University of Dubai, 14143, UAE
[8]EECS Department, Center for Cyber Physical Systems, Khalifa University, Abu Dhabi, 127788, UAE

*Corresponding Author: Chan Yeob Yeun. Email: chan.yeun@ku.ac.ae

Received: 22 November 2022; Accepted: 16 March 2023

Abstract: The software engineering field has long focused on creating high-quality software despite limited resources. Detecting defects before the testing stage of software development can enable quality assurance engineers to concentrate on problematic modules rather than all the modules. This approach can enhance the quality of the final product while lowering development costs. Identifying defective modules early on can allow for early corrections and ensure the timely delivery of a high-quality product that satisfies customers and instills greater confidence in the development team. This process is known as software defect prediction, and it can improve end-product quality while reducing the cost of testing and maintenance. This study proposes a software defect prediction system that utilizes data fusion, feature selection, and ensemble machine learning fusion techniques. A novel filter-based metric selection technique is proposed in the framework to select the optimum features. A three-step nested approach is presented for predicting defective modules to achieve high accuracy. In the first step, three supervised machine learning techniques, including Decision Tree, Support Vector Machines, and Naïve Bayes, are used to detect faulty modules. The second step involves integrating the predictive accuracy of these classification techniques through three ensemble machine-learning methods: Bagging, Voting, and Stacking. Finally, in the third step, a fuzzy logic technique is employed to integrate the predictive accuracy of the ensemble machine learning techniques. The experiments are performed on a fused software defect dataset to ensure that the developed fused ensemble model can perform effectively on diverse datasets. Five NASA datasets are integrated to create the fused dataset: MW1, PC1, PC3, PC4, and CM1. According to the results, the proposed system exhibited superior performance to other advanced techniques for predicting software defects, achieving a remarkable accuracy rate of 92.08%.

Keywords: Ensemble machine learning fusion; software defect prediction; fuzzy logic

1 Introduction

Most of the researchers from the software engineering domain have been working to minimize the cost of the Software Development Life Cycle (SDLC) without compromising on the quality [1,2]. The activity of software testing aims to ensure the high quality of the end product [3–5]. A minor defect in software can lead to system failure and a catastrophic event in the case of a critical system [1–3]. The importance of identifying and removing the defects can be reflected by an example of "NASA's $125 million Mars Climate Orbiter", which was lost because of a minor data conversion defect [1]. Software defects can be of different types, including wrong program statements, syntax errors, and design or specification errors [1,2]. In SDLC, the testing process plays a crucial role in achieving a high-quality end product by eliminating defects [6,7]. However, it has been proved that software testing is the most expensive activity as it takes most of the resources as compared to other tasks of SDLC [3,8–10]. Identifying and fixing the defects before testing would be less costly compared to the cost of repairing the defects at later stages, especially after integration [11,12]. This objective can be achieved by incorporating an effective Software Defect Prediction (SDP) system [13,14], which can identify faulty software modules before the testing stage, allowing for focused testing efforts on those particular modules [1]. This approach can guarantee the delivery of high-quality end products with limited resources [3]. Many supervised machine learning-based defect prediction techniques and frameworks have been proposed by researchers for the effective and efficient detection of defect-prone software modules [1,2]. In the supervised machine learning technique, a classifier is trained by using a pre-labeled dataset. The dataset which is used to train the classifier includes multiple independent features and at least one dependent feature. The dependent feature is known as the output class, which is classified or predicted by exploring the hidden pattern and relationship between independent attributes and dependent attributes. That hidden pattern and relationship are learned by the supervised classifier, which is further used for prediction on the unseen dataset (testing data) [6–8]. The SDP dataset typically pertains to a specific software component, with independent attributes represented by software quality metrics collected during development. The dependent feature, on the other hand, is the predictable class which reflects whether the particular module is defective or not. Instances in the SDP dataset reflect the modules, and by classifying a specific instance as defective or non-defective, we are predicting a particular module that is reflected by the instance. This study presents the Intelligent Software Defect Prediction System (ISDPS), which utilizes data and decision-level ensemble machine learning fusion, along with a novel filter-based ensemble feature selection technique to improve accuracy and reduce costs. The ISDPS follows a three-step process for software defect prediction, beginning with the use of three supervised classification techniques—Decision Tree (DT), Support Vector Machines (SVM), and Naïve Bayes (NB)—to build classification models for SDP. The second step employs ensemble techniques such as Bagging, Voting, and Stacking to merge the predictive accuracy of the classification models. The third step uses fuzzy logic to fuse the predictive accuracy of the ensemble models. The proposed system was evaluated by combining five datasets from NASA's repository and achieved a high accuracy rate of 92.08%, outperforming other published techniques.

2 Related Work

Researchers have presented various models and frameworks to detect faulty software modules before the testing stage. Several studies have been conducted on this topic, and some are discussed here. In [14], researchers applied a metric selection algorithm and ensemble machine learning approach to detect defective software modules. The research was conducted on six different software defect datasets taken from NASA's software repository. The performance of the proposed method was evaluated using statistical measures such as Receiver Operating Characteristic (ROC) value, Matthews's Correlation Coefficient (MCC), F-measure, and Accuracy. The study compared various search methods for metric selection using ensemble machine learning. The results demonstrated that the proposed method outperformed all other techniques, as evidenced by its superior performance compared to various supervised classifiers. Researchers in [15] proposed an Artificial Neural Network (ANN) based system for detecting faulty software modules, along with a metric selection algorithm that uses a multi-filter-based approach. The proposed system uses oversampling to handle class imbalance and performs classification in two dimensions, one with oversampling and one without. NASA's cleaned software defect datasets were used for implementation, and the system's performance was evaluated using various statistical metrics such as MCC, ROC, Accuracy, and F-measure. The study compared the proposed SDP technique with other methods, and the results showed that the proposed technique outperformed other techniques in terms of accuracy and other statistical measures. In their study [16], researchers introduced a framework for predicting faulty software modules based on Multi-Layer Perceptron (MLP) with bagging and boosting as ensemble machine learning techniques. The framework employs three approaches for predicting defective modules, including tuning the MLP for classification, creating an ensemble of tuned MLP using bagging, and developing an ensemble of tuned MLP using boosting. The study utilized four cleaned datasets from NASA's repository to implement the proposed framework and evaluated its performance using statistical metrics such as Accuracy, MCC, F-measure, and ROC. The results indicated that the proposed technique outperformed various classifiers from published research, as determined by the test of Scott Knott Effect Size Difference. According to [17], researchers conducted a thorough comparative analysis of four commonly used training techniques for back-propagation algorithms in ANN to predict defective software modules. They also proposed a fuzzy logic-based layer to determine the most effective training technique. The researchers utilized cleaned versions of NASA software defect datasets and assessed the performance of the proposed framework using several metrics, such as Accuracy, Specificity, F-measure, Recall, Precision, ROC, and Mean Square Error. The study compared the results with those of various classifiers, and the proposed framework outperformed other methods. In [18], a machine learning-based framework using a variant ensemble technique is presented for predicting faulty software modules. The proposed framework also incorporates a metric selection method to optimize the performance of the classification technique. The variant selection process involves identifying and selecting the best version of the classification technique to achieve high performance. The ensemble technique integrates the predictive accuracy of the optimized variants to further increase the accuracy of the proposed framework. The proposed framework was tested using four cleaned software defect datasets from NASA's repository, and its performance was evaluated using three statistical measures: MCC, Accuracy, and F-measure. The results of the framework were compared with various other classifiers to assess its effectiveness. The analysis showed that the proposed framework outperformed all other classifiers, indicating its superiority in predicting faulty software modules. The authors of [19] performed a thorough comparative analysis of multiple supervised machine learning techniques for software defect prediction. They used twelve cleaned software defect datasets from NASA and evaluated the performance of the models using various statistical measures such as F-measure, Precision, Accuracy, Recall, ROC, and MCC. The authors concluded that the results obtained from their study could serve as a benchmark for future comparisons between different SDP techniques. In [20], researchers designed an Artificial Neural Network (ANN) tool to detect defective software modules. They compared different training functions of ANN and identified that the Bayesian Regularization training function outperformed others. The objective of this study was to decrease the cost of software development by detecting faulty modules before they reach the testing stage, thus reducing the burden on the quality assurance team.

3 Materials and Methods

The study introduces an intelligent system that employs data fusion to detect defective software modules. The system integrates feature selection and decision-level ensemble machine learning fusion techniques to improve the accuracy and efficiency of identifying faulty software modules. The ISDPS proposed in this study can be viewed from two perspectives: external and internal. The exterior view, as shown in Fig. 1, outlines the workflow surrounding the defect prediction system. The development team initiates the workflow during the development stage of the software development life cycle (SDLC). Understanding the surrounding scenario is crucial to comprehend the importance of the proposed system. A software metric dataset is prepared during the process of software development. The dataset consists of various quality metrics captured automatically or manually during development. Every single dataset reflects a particular Software Component (SC) in which there are many modules. Each module in a specific SC dataset is reflected by an instance, and the values of quality attributes/features of that instance are populated during development. Each SC dataset consists of various quality attributes (independent features) and one dependent feature (also known as output class). Initially, the developed software components in SDLC are stored in Software Metric Dataset Repository (SMDR). Each component has its Quality Metrics Dataset (QMD), which contains the values of various quality attributes recorded during the development stage. The SMDR consists of two further sub-repositories: The untested Software Components Repository (USCR) and Tested Software Components Repository (TSCR). Initially, the developed components are not tested and are stored in USCR. Some selected components from USCR are tested by the Quality Assurance (QA) team, and an additional attribute is added to the QMD, known as "Result". This column will reflect the nominal value of "yes" if the particular module is defective and "no" if the module is non-defective. The tested component, along with its QMD with result attribute, is stored in the TSCR sub-repository. When the TSCR is initially populated with tested components, then the proposed ISDPS will come into work as it will need the pre-labeled dataset for training.

Two or more datasets from TSCR are extracted into the training layer of the proposed defect prediction system, where data fusion will be performed, and the prediction model is developed, which will be stored in the cloud for later use. The testing layer of the proposed system will receive QMD as input from USCR to perform real-time defect predictions. The "Result" attribute in QMD is populated by the testing layer after prediction and then will be sent to the QA team. The QA team will add the QMD to its related SC, and if the module is predicted as defective, then thorough testing of that particular module will be performed, and the detail of identified defect will be sent to the development team in SDLC. The development team will correct the defective module, and then it will be again transferred to USCR for the next iteration. If the module is predicted as non-defective by the proposed ISDPS, then it would be considered as good to go to the integration stage of SDLC.

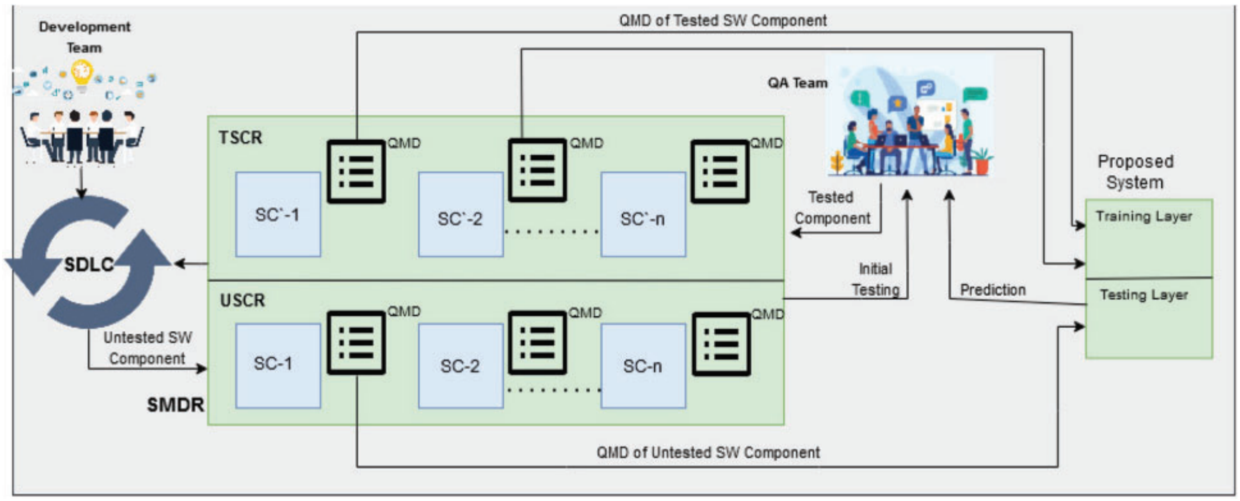


Figure 1: External view of proposed ISDPS

The internal view (Fig. 2) of the proposed ISDPS contains two layers: training and testing. These layers are comprised of several stages and activities. The workflow in the training layer initiates with the extraction of pre-labeled QMDs from SMDR. Data fusion is the first stage of the training layer in the proposed system, in which datasets of multiple software components will be extracted, and instance-level fusion will be performed. The main objective of the data fusion process is to develop an effective and efficient classification model, which can be used for prediction on diverse test datasets from multiple sources. No doubt, training the model with higher accuracy on the fused dataset is challenging but eventually fruitful for later use, especially when prediction has to be performed on multiple datasets from multiple sources. However, it should be ensured that the nature of the test data would be the same as that of the training dataset. For this study, five cleaned datasets were chosen from NASA's repository, including MW1, PC1, PC3, PC4, and CM1. The details of the used datasets and their attributes are available in [21]. After extraction, all five datasets are fused. The fused dataset consists of 3579 instances and 38 attributes. Of 38 attributes, 37 attributes are independent, whereas one attribute named "Defective" is dependent, which aims to determine whether a module is defective. The dependent attribute, also known as the output class, can contain two values, "Yes" for defective modules and "No" for non-defective modules.

Pre-processing is the second stage of the training layer, which will deal with four data pre-processing activities, including 1) cleaning, 2) normalization, 3) splitting of data for training and testing, and 4) feature selection for efficient and effective prediction. The cleaning process of the pre-processing stage will be responsible for handling the missing values in the dataset. The missing values will be replaced by using the technique of mean imputation. Missing values in any attribute of the dataset can misguide the classification model, which may result in low accuracy of the proposed framework. The second activity deals with the normalization process, which scales the values of all independent attributes to a range of 0 to 1. It has been observed that cleaning and normalization activities aid the classification techniques to work efficiently and effectively. The third activity will deal with the splitting of the dataset into the groups of training and testing datasets with a ratio of 70:30 by using the split class rule. The fourth and final activity of pre-processing stage will deal with the selection of optimum features [14,15] from training and test sets by using a novel feature selection (FS) technique. A novel filter-based ensemble feature selection (FEFS) technique is proposed in which feature selection is performed three times in a nested way. The proposed FEFS technique consists of four steps. First, the complete feature set of training data is given as input to the Correlation-based FS technique with Genetic Search (GS) method. In the second step, Consistency based FS is performed on complete training data with Best First (BF) search method. In the third step, an intersection is performed among both of the resultant feature sets from step 1 and step 2 (Correlation FS and Consistency FS). Finally, in step four, the feature set generated from the intersection operation from step 3 is given as input again to the correlation-based FS technique with the GS method, and the resultant feature set is selected from training and test datasets. The detailed steps of the proposed FEFS technique are given below (Table 1).

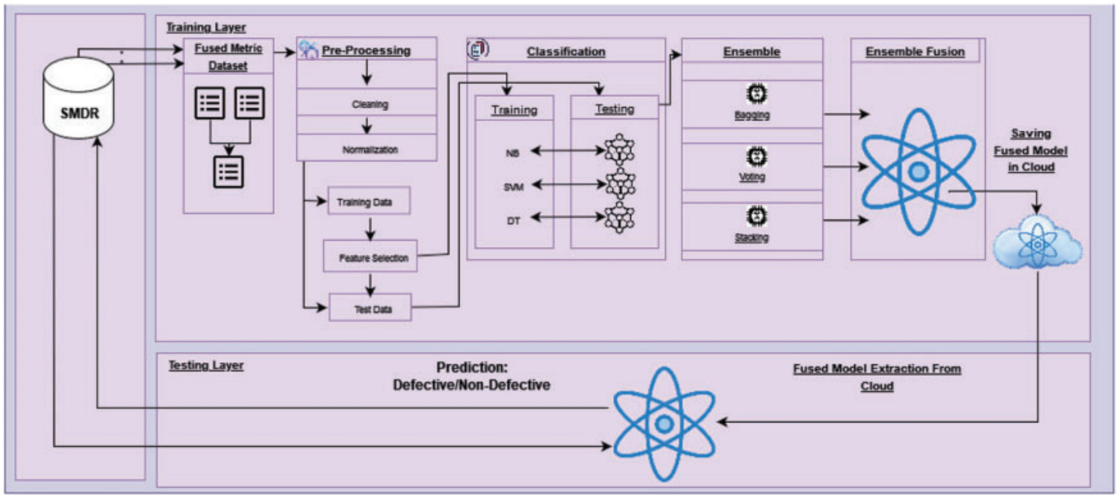


Figure 2: Internal view of proposed ISDPS

Table 1: Proposed FEFS technique

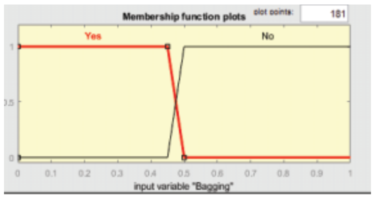
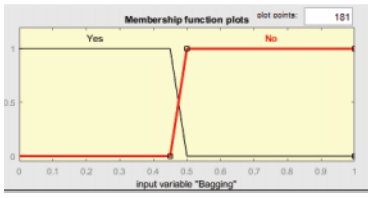
Input:
Training Dataset: Dataset A
Test Dataset: Dataset B
Attribute Evaluator: Correlation FS, Consistency FS
Search Methods: GS—BF
Output:
n = numbers of features
Steps:
1 Dataset A → Correlation FS-GS → Subset 1: a1, a2 . . . , an;
2 Dataset A → Consistency FS-BF → Subset 2: b1, b2 . . . , bn;
3 Subset 1 Intersection Subset 2 → Subset 3: c1, c2 . . . , cn;
4 Subset 3: → Correlation FS-GS → Subset 4: d1, d2, . . . dn;
5 Select Subset 4 as Feature Set from Dataset A and Dataset B.

Classification is the third stage which deals with the development of classification models to identify defective and non-defective modules. Selected features of pre-processed datasets (training and testing) are used as input to the classification stage. The study employed three supervised machine learning techniques, namely NB, SVM, and DT, for classification. These classifiers were fine-tuned iteratively to achieve the highest possible accuracy on the testing dataset. During the tuning process on training data, default parameters are used in NB as the performance decreases after optimization. In SVM, the polynomial kernel is selected, and the value of the complexity parameter (C) is set to 1. In DT, the confidence factor has been set to 0.3. The classification stage will end by producing the optimized prediction models of the used supervised machine learning techniques.

Ensemble Modeling is the fourth stage in the training layer which deals with the development of ensemble models by integrating the optimized classification classifiers (NB, SVM, and DT), which are given as input to the ensemble modeling stage. The ensemble machine learning approaches can further increase the prediction accuracy than individual optimized classification techniques [3,7,14,22,23]. Three ensemble techniques will be used for the development of ensemble models, including Bagging, Voting, and Stacking. One by one, all of the optimized classification models are used as input to the ensemble techniques for the development of ensemble models. Three ensemble models are developed in the proposed system: Bagging with DT, Voting with NB, SVM, and DT, and Stacking with NB and SVM along with DT as a Meta classifier. All three developed ensemble models have shown better accuracy on test data than each of the optimized individual classifiers.

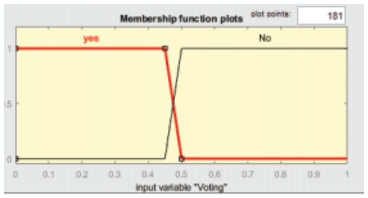
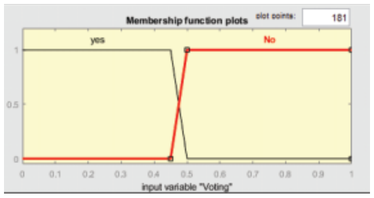
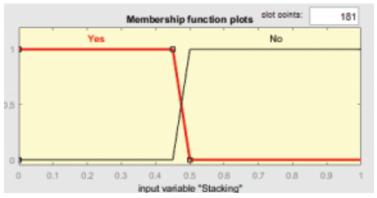
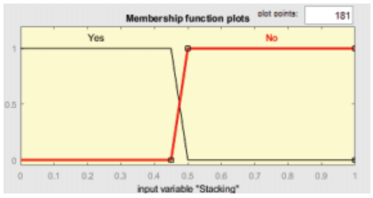
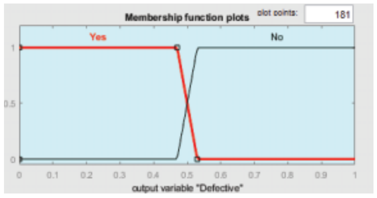
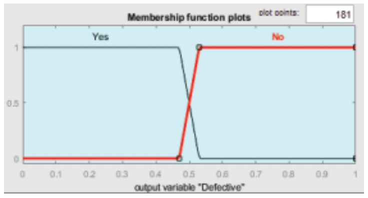
The fifth and final stage of the training layer deals with the fusion of ensemble machine-learning techniques. This stage is responsible for decision-level fusion by integrating the predictive accuracy of optimized ensemble models [24]. Fuzzy logic is used for decision-level fusion, where membership functions are developed using if-then rules (as shown in Table 2). These rules form the basis of the final prediction and enhance the accuracy of the defect prediction system. The fused ensemble model is then stored in cloud storage for real-time predictions.

Table 2: Membership functions of proposed fusion method

Membership functions	Graphical representation
$\prod_{\mathrm{B}\mathfrak{G}\mathrm{Y}}(\boldsymbol{bg}) = \left\{\max\left(\min\left(1, \frac{0.5 - bg}{0.05}\right), 0\right)\right\}$	
$\prod_{\mathrm{B}\mathfrak{G}\mathfrak{N}}(\boldsymbol{bg}) = \left\{\max\left(\min\left(\frac{bg - 0.45}{0.05}, 1\right), 0\right)\right\}$	

(Continued)

Table 2: Continued

Membership functions	Graphical representation
$\prod_{\mathcal{VŦY}}(\mathfrak{vt}) = \left\{\max\left(\min\left(1, \frac{0.5-\mathfrak{vt}}{0.05}\right), 0\right)\right\}$	
$\prod_{\mathcal{VŦN}}(\mathfrak{vt}) = \left\{\max\left(\min\left(\frac{\mathfrak{vt}-0.45}{0.05}, 1\right), 0\right)\right\}$	
$\prod_{\mathrm{sky}}(\mathfrak{sk}) = \left\{\max\left(\min\left(1, \frac{0.5-\mathfrak{sk}}{0.05}\right), 0\right)\right\}$	
$\prod_{\mathrm{skn}}(\mathfrak{sk}) = \left\{\max\left(\min\left(\frac{\mathfrak{sk}-0.45}{0.05}, 1\right), 0\right)\right\}$	
$\prod \check{D}_{ɖ}(đ) = \left\{\max\left(\min\left(1, \frac{0.5-đ}{0.05}\right), 0\right)\right\}$	
$\prod \check{D}_{ŋ}(đ) = \left\{\max\left(\min\left(\frac{đ-0.45}{0.05}, 1\right), 0\right)\right\}$	

If-Then conditions that are used to develop membership functions are given below:

IF (Bagging is yes and Voting is yes and Stacking is yes) THEN (Module is defective).

IF (Bagging is yes and Voting is yes and Stacking is no) THEN (Module is defective).

IF (Bagging is yes and Voting is no and Stacking is yes) THEN (Module is defective).

IF (Bagging is no and Voting is yes and Stacking is yes) THEN (Module is defective).

IF (Bagging is no and Voting is no and Stacking is also no) THEN (Module is not defective).

IF (Bagging is yes and Voting is no and Stacking is no) THEN (Module is not defective).

IF (Bagging is no and Voting is no and Stacking is yes) THEN (Module is not defective).

IF (Bagging is no and Voting is yes and Stacking is no) THEN (Module is not defective).

Fig. 3 depicts the ruled surface of the proposed fuzzy logic-based fusion method for final prediction, in contrast to the bagging and voting ensemble techniques. In cases where both techniques predict that the software module is not defective, the fused model will make the same prediction. Likewise, if both techniques predict that the module is defective, the fused model will also predict that the module is defective.

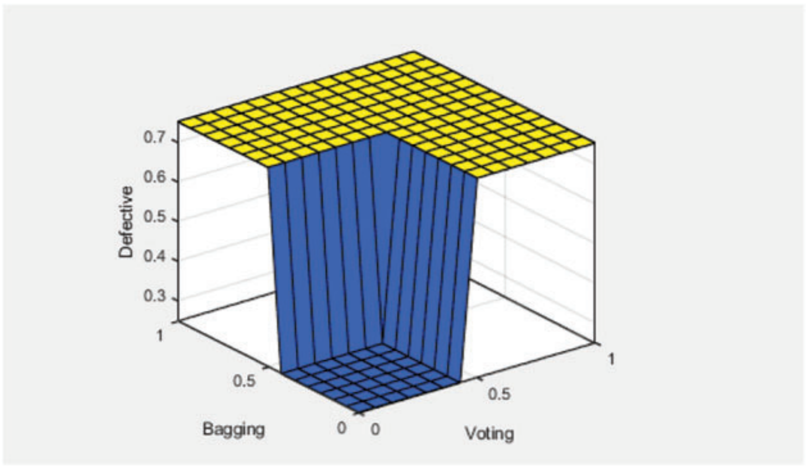


Figure 3: Rule surface of fused ensemble method with bagging and voting

Fig. 4 shows the graphical representation of fuzzy logic based if-then rule for the scenario; when bagging and stacking, both techniques predict that the particular module is defective, whereas the voting technique predicts the opposite (non-defective). In this case, the proposed technique would go for a majority decision (defective).

Fig. 5 illustrates that if bagging and stacking both predict that the module is non-defective, the proposed technique will also predict that the module is non-defective.

The testing layer is the implementation layer of the proposed ISDPS. In this layer, three activities are performed. The first activity deals with the extraction of unlabeled QMD from USCR for the prediction of defective software modules. The second activity involves extracting the fused model from the cloud for prediction. The third activity of the testing layer deals with real-time prediction in which unlabeled QMD is given as input to the fused model, which is then labeled after prediction. The labeled QMD is attached to its related SC in TSCR and then sent back to the development life cycle. If the software is predicted as defective, then the particular defect is rectified by the development team; otherwise, the particular software component would be considered good to go for integration.

4 Results and Discussion

An empirical analysis was conducted to assess the effectiveness of the proposed ISDPS using a fused software defect dataset. The dataset was created by combining five of NASA's cleaned datasets. The fused dataset contains 3579 instances, with 428 defective and 3151 non-defective. In

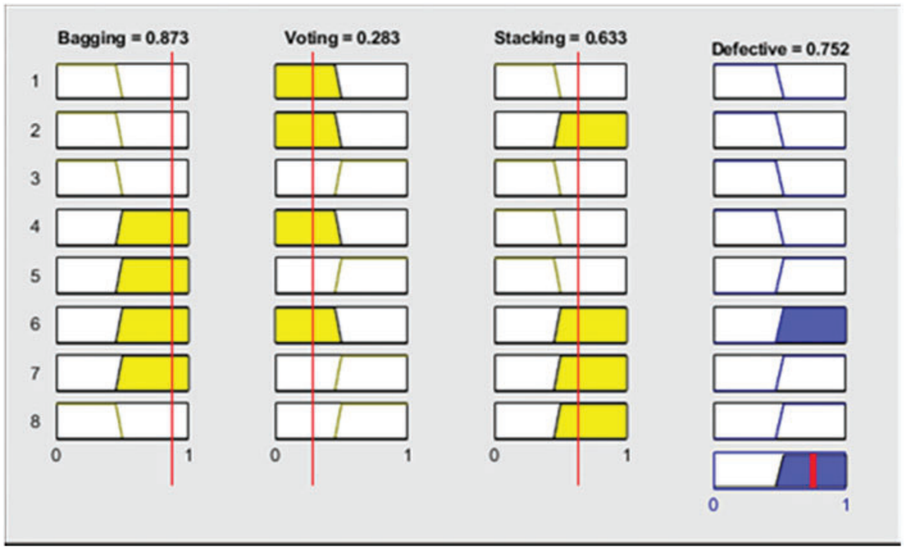


Figure 4: Result of proposed fused ensemble method with defective module (1)

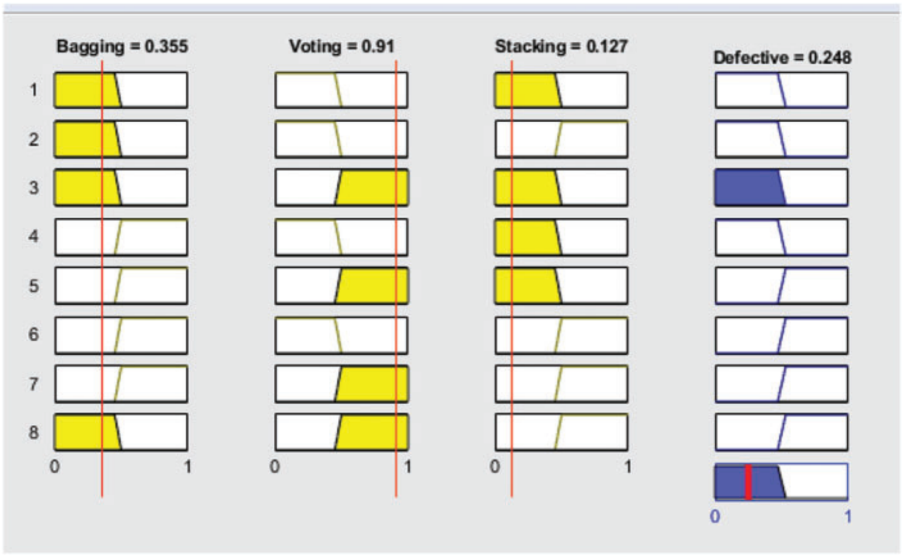


Figure 5: Result of proposed fused ensemble method with non-defective module (0)

the pre-processing stage of the training layer, the dataset underwent cleaning and normalization processes, followed by the splitting process, where the dataset was divided into training and test subsets with a 70:30 ratio. The training subset comprised 2506 instances, while the test subset contained 1073 instances. A novel FEFS technique is proposed for effective and efficient prediction, which is implemented on the complete feature set of training data to select the optimum feature set. The proposed method chose 7 out of 37 independent features. The detail of the full features of the fused dataset is available at [21], whereas the feature set selected by the proposed FEFS technique is shown in Table 3.

Table 3: Selected features using the proposed FEFS technique

No.	Selected features
1	LOC_BLANK
2	LOC_CODE_AND_COMMENT
3	CYCLOMATIC_DENSITY
4	PARAMETER_COUNT
5	HALSTEAD_CONTENT
6	NUM_OPERATORS
7	PERCENT_COMMENTS

In the proposed ISDPS, prediction is performed in three steps. Initially, three supervised machine learning techniques (NB, SVM, and DT) are iteratively optimized for classification in the first step until the highest possible accuracy is attained for each model. The optimized classification models created from these classifiers are given to the second step of prediction, where three ensemble techniques (Bagging, Voting, and Stacking) are used to integrate the predictive accuracy of used classifiers. The classifiers are integrated by ensemble methods with all possible combinations until we get three ensemble classification models, one from each ensemble technique that performed higher than the base classifier. The results of ensemble techniques are given as input to the final prediction step, which is empowered by fuzzy logic.

The accuracy measures used for the performance analysis of the proposed ISDPS are discussed below:

$$Misrate = \frac{(AOR_1/EOR_0 + AOR_0/EOR_1)}{EOR_0 + EOR_1} \tag{1}$$

$$Accuracy = \frac{(AOR_0/EOR_0 + AOR_1/EOR_1)}{EOR_0 + EOR_1} \tag{2}$$

$$Positive\ Prediction\ Value = \frac{AOR_1/EOR_1}{(AOR_1/EOR_1 + AOR_0/EOR_1)} \tag{3}$$

$$Negative\ Prediction\ Value = \frac{AOR_0/EOR_0}{(AOR_0/EOR_0 + AOR_1/EOR_0)} \tag{4}$$

$$Specificity = \frac{AOR_0/EOR_0}{(AOR_0/EOR_0 + AOR_0/EOR_1)} \tag{5}$$

$$Sensitivity = \frac{AOR_1/EOR_1}{(AOR_1/EOR_0 + AOR_1/EOR_1)} \tag{6}$$

$$False\ Positive\ Ratio = 1 - S\text{pecificity} \tag{7}$$

$$False\ Positive\ Ratio = 1 - S\text{pecificity} \tag{8}$$

$$Likelihood\ Ratio\ Positive = \frac{\text{Sensitivity}}{(1\ - Specificity)} \tag{9}$$

$$\textit{Likelihood Ratio Negative} = \frac{(1 - \text{Sensitivity})}{\textit{Specificity}} \tag{10}$$

The training data, which consists of 2506 instances, are used to train the classifiers and ensemble models. During the NB training process, 2061 instances are correctly predicted as negative, whereas 80 instances are correctly predicted as positive. The output result and achieved results can be compared in Table 4, which reflects that the training process achieved 85.43% accuracy and a 14.57% miss rate in NB. In the process of testing, 865 instances are correctly predicted as negative, whereas 46 instances are correctly predicted as positive. After comparing the output result and expected results (Table 4), 84.90% accuracy is achieved, with a miss rate of 15.10% in NB testing.

Table 4: NB results

	Training data			Testing data		
	Samples = 2506	Output (AOR_0, AOR_1)		Samples = 1073	Output (AOR_0, AOR_1)	
Input	Expected Result (EOR_0, EOR_1)	AOR_0 (Negative-0)	AOR_1 (Positive-1)	Expected Result (EOR_0, EOR_1)	AOR_0 (Negative)	AOR_1 (Positive)
	$EOR_0 = 2206$ (Negative-0)	2061	145	$EOR_0 = 945$ (Negative-0)	865	80
	$EOR_1 = 300$ (Positive-1)	220	80	$EOR_1 = 128$ (Positive-0)	82	46

In the training process of SVM, 2135 instances are correctly predicted as negative, whereas 40 instances are correctly predicted as positive. In the training process with SVM, 86.79% accuracy is achieved, along with a miss rate of 13.21% after analyzing the results in Table 5. Testing results show that 905 instances are correctly predicted as negative, whereas 24 instances are correctly predicted as positive. After analyzing the expected and output results, the achieved accuracy in SVM testing is 86.58%, with a miss rate of 13.42%.

Table 5: SVM results

	Training data			Testing data		
	Samples = 2506	Output (AOR_0, AOR_1)		Samples = 1073	Output (AOR_0, AOR_1)	
Input	Expected Result (EOR_0, EOR_1)	AOR_0 (Negative-0)	AOR_1 (Positive-1)	Expected Result (EOR_0, EOR_1)	AOR_0 (Negative)	AOR_1 (Positive)
	$EOR_0 = 2206$ (Negative-0)	2135	71	$EOR_0 = 945$ (Negative-0)	905	40
	$EOR_1 = 300$ (Positive-1)	260	40	$EOR_1 = 128$ (Positive-0)	104	24

In DT training, 2122 instances are correctly predicted as negatives, whereas 100 instances are correctly predicted as positives. Upon reviewing the expected and achieved outputs presented in Table 6, 88.67% accuracy with an 11.33% miss rate is achieved. In the testing process of DT, 884 instances are correctly predicted as negative, whereas 51 instances are correctly predicted as positive. After analyzing the results (Table 6), 87.14% accuracy is achieved with a miss rate of 12.86%.

Table 6: DT results

Training data				Testing data		
	Samples = 2506	Output (AOR_0, AOR_1)		Samples = 1073	Output (AOR_0, AOR_1)	
Input	Expected result (EOR_0, EOR_1)	AOR_0 (Negative-0)	AOR_1 (Positive-1)	Expected result (EOR_0, EOR_1)	AOR_0 (Negative)	AOR_1 (Positive)
	$EOR_0 = 2206$ (Negative-0)	2122	84	$EOR_0 = 945$ (Negative-0)	884	61
	$EOR_1 = 300$ (Positive-1)	200	100	$EOR_1 = 128$ (Positive-0)	77	51

After the development of classification models using supervised machine learning techniques (NB, SVM, DT), ensemble machine learning models are developed. In training with the bagging technique, 2205 instances are correctly predicted as negative, whereas 164 instances are predicted as positive. After analyzing the training results shown in Table 7, 94.53% accuracy is achieved with a miss rate of 5.47%. Testing with bagging correctly predicted 913 instances as negative, whereas no of correctly predicted positive instances 55. Upon comparing the expected results with the achieved results, it can be concluded that the testing yielded an accuracy of 90.21% and a miss rate of 9.79%.

Table 7: Bagging results

Training data				Testing data		
	Samples = 2506	Output (AOR_0, AOR_1)		Samples = 1073	Output (AOR_0, AOR_1)	
Input	Expected result (EOR_0, EOR_1)	AOR_0 (Negative-0)	AOR_1 (Positive-1)	Expected result (EOR_0, EOR_1)	AOR_0 (Negative)	AOR_1 (Positive)
	$EOR_0 = 2206$ (Negative-0)	2205	1	$EOR_0 = 945$ (Negative-0)	913	32
	$EOR_1 = 300$ (Positive-1)	136	164	$EOR_1 = 128$ (Positive-0)	73	55

Training with voting correctly predicted 2196 instances as negative, whereas 39 instances were correctly predicted as positive. After analyzing the results from Table 8, 89.19% accuracy is achieved with a 10.81% miss rate. The testing process with voting correctly predicted the 897 instances as negatives, whereas 58 instances were correctly predicted as positives. The results reflect 89% accuracy and an 11% miss rate.

During the training process with stacking ensemble, 2201 instances are correctly classified as negatives, whereas 53 instances are correctly predicted as positives. Table 9 presents the output results and expected outcome, demonstrating an accuracy of 89.94% and a miss rate of 10.06%. Testing with stacking ensemble correctly classified 911 instances as negatives, whereas 54 instances were classified as positives. A comparison of the expected and output results reveals an accuracy of 89.93% and a miss rate of 10.07%.

Finally, the test dataset is given to the proposed model, which correctly predicted 926 instances as negatives out of 945 instances, whereas, on the other hand, it correctly predicted 62 instances as

Table 8: Voting results

	Training data			Testing data		
	Samples = 2506	Output (AOR_0, AOR_1)		Samples = 1073	Output (AOR_0, AOR_1)	
Input	Expected result (EOR_0, EOR_1)	AOR_0 (Negative-0)	AOR_1 (Positive-1)	Expected result (EOR_0, EOR_1)	AOR_0 (Negative)	AOR_1 (Positive)
	$EOR_0 = 2206$ (Negative-0)	2196	10	$EOR_0 = 945$ (Negative-0)	897	48
	$EOR_1 = 300$ (Positive-1)	261	39	$EOR_1 = 128$ (Positive-0)	70	58

Table 9: Stacking results

	Training Data			Testing Data		
	Samples = 2506	Output (AOR_0, AOR_1)		Samples = 1073	Output (AOR_0, AOR_1)	
Input	Expected result (EOR_0, EOR_1)	AOR_0 (Negative-0)	AOR_1 (Positive-1)	Expected result (EOR_0, EOR_1)	AOR_0 (Negative)	AOR_1 (Positive)
	$EOR_0 = 2206$ (Negative-0)	2201	5	$EOR_0 = 945$ (Negative-0)	911	34
	$EOR_1 = 300$ (Positive-1)	247	53	$EOR_1 = 128$ (Positive-0)	74	54

positives out of 128 instances. The results are shown in Table 10, according to which the proposed system has achieved 92.08% accuracy and a 7.92% miss rate.

Table 10: Fused ensemble testing

	N = 1073 (No. of samples)	Output result (AOR_0, AOR_1)	
Input	Expected result ($EOR_{0,}$ EOR_1)	AOR_0 (Negative-0)	AOR_1 (Positive-1)
	$EOR_0 = 945$ (Negative-0)	926	19
	$EOR_1 = 128$ (Positive-1)	66	62

Table 11 shows the detailed results of base classifiers and ensemble classification models on training and testing data, along with the results of the proposed ISDPS on test data. The analysis showed that the proposed system outperformed both the base classifiers (NB, SVM, and DT) and the ensembles (Bagging, Voting, and Stacking). It can be observed that the results achieved from ensemble models are better than the results of base classifiers, and the results of final prediction by decision-level fusion with fuzzy logic further increased the accuracy to 92.08%. The effectiveness of the proposed system can be inferred from its performance on the fused dataset in comparison to other models.

Table 11: Detailed results of classifiers, ensembles, and ensemble fusion

ML Algorithm	Dataset	Accuracy	Miss rate	Sensitivity	Specificity	Positive prediction value	Negative prediction value	False positive value	False negative value	Likelihood ratio negative	Likelihood ratio positive
Naïve Bayes	Training	0.8543	0.1457	0.2667	0.9343	0.3556	0.9036	0.0657	0.7333	0.7849	4.0570
	Testing	0.8490	0.1510	0.3594	0.9153	0.3651	0.9134	0.0847	0.6406	0.6999	4.2451
Support vector machines	Training	0.8679	0.1321	0.1333	0.9678	0.3604	0.8914	0.0322	0.8667	0.8955	4.1427
	Testing	0.8658	0.1342	0.1875	0.9577	0.375	0.8969	0.0423	0.8125	0.8484	4.4297
Decision tree	Training	0.8867	0.1133	0.3333	0.9619	0.5435	0.9139	0.0381	0.6667	0.6931	8.7540
	Testing	0.8714	0.1286	0.3984	0.9354	0.4554	0.9199	0.0646	0.6016	0.6431	6.1725
Bagging	Training	0.9453	0.0547	0.5467	0.9995	0.9939	0.9419	0.0005	0.4533	0.4535	1205.9467
	Testing	0.9021	0.0979	0.4297	0.9661	0.6322	0.9260	0.0339	0.5703	0.5903	12.6892
Voting	Training	0.8919	0.1081	0.13	0.9955	0.7959	0.8938	0.0045	0.87	0.8740	28.678
	Testing	0.8900	0.1100	0.4531	0.9492	0.5472	0.9276	0.0508	0.5469	0.5761	8.92090
Stacking	Training	0.8994	0.1006	0.1767	0.9977	0.9138	0.8991	0.0023	0.8233	0.8252	77.9453
	Testing	0.8993	0.1007	0.4218	0.9640	0.6136	0.9249	0.0360	0.5781	0.5997	11.7257
Proposed fussed ensemble	Testing	0.9208	0.0792	0.4844	0.9799	0.7654	0.9335	0.0201	0.5156	0.5262	24.0913

The performance of the proposed ISDPS is compared with other techniques in terms of accuracy in Table 12. It is reflected that the accuracy achieved by the proposed ISDPS is higher than other published techniques. The data fusion technique usually decreases the accuracy of the prediction system as training of classification models on the dataset extracted from multiple sources is challenging as compared to training on a dataset extracted from a single source. However, the proposed FEFS technique for the selection of optimum attributes as well as the multi-step prediction system played crucial roles in achieving high accuracy of the proposed ISDPS.

Table 12: Accuracy comparison of proposed system with other methods

Algorithm	Accuracy %	Miss rate %
MLP-FS [15]	85.13	14.87
Boosting-OPT-MLP [16]	79.08	20.92
ANN-BR-fused [17]	85.45	14.55
FS-variant ensemble ML [18]	84.97	15.03
NB [19]	82.65	17.35
MLP [25]	89.96	10.04
Tree [25]	84.94	15.06
Bagging ensemble [26]	80.20	19.8
Boosting ensemble [26]	81.30	18.7
Heterogeneous classifier [27]	89.20	10.8
Stacked ensemble [28]	89.10	10.9
RBFNN-based ADBBO [29]	88.65	11.35
LWL-based bagging ensemble [30]	90.10	9.9
Proposed ensemble ml fusion approach	92.08	7.92

5 Conclusion

Software testing is considered an expensive activity of SDLC, which aims to ensure the high quality of the end product by removing software bugs. Anticipating software faults before the testing phase can assist the quality assurance team in directing their attention towards potentially defective software modules during the testing process instead of having to scrutinize every module. This process would limit the cost of the testing process, which would ultimately reduce the overall development cost without compromising on software quality. The current study aimed to develop a system that can predict faulty software modules before the testing stage by utilizing data fusion, feature selection, and decision-level ensemble machine-learning fusion techniques. A novel FEFS technique is proposed to select optimum features from the input dataset. The proposed system used NB, SVM, and DT for initial predictions, followed by the development of ensemble models using Bagging, Voting, and Stacking. The predictions from ensemble models are then given to the decision-level fusion phase, which works on a fuzzy logic-based technique for the final prediction. The decision-level fusion integrated the predictive accuracy of ensemble models by if-then rules-based fuzzy logic. Five clean datasets are fused from NASA's software repository to implement the proposed system. After comparing the performance of the proposed ISDPS with other defect prediction techniques published in the literature, it was found that the ISDPS outperformed all other methods on the fused dataset.

The proposed system achieved an accuracy of 92.08% on the fused data, indicating the effectiveness of the novel FEFS and decision-level ensemble machine-learning fusion techniques. For future work, it is suggested that hybrid filter-wrapper feature selection and deep extreme machine-learning techniques should be incorporated into the proposed system. Moreover proposed design should also be optimized for cross-project defect prediction problems.

Acknowledgement: Thanks to our families & colleagues who supported us morally.

Funding Statement: This work was supported by the Center for Cyber-Physical Systems, Khalifa University, under Grant 8474000137-RC1-C2PS-T5.

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

References

[1] F. Matloob, S. Aftab, M. Ahmad, M. A. Khan, A. Fatima *et al.,* "Software defect prediction using supervised machine learning techniques: A systematic literature review," *Intelligent Automation & Soft Computing*, vol. 29, no. 2, pp. 403–421, 2021.

[2] D. R. Ibrahim, R. Ghnemat and A. Hudaib, "Software defect prediction using feature selection and random forest algorithm," in *Int. Conf. on New Trends in Computer Science*, Amman, Jordan, pp. 252–257, 2017.

[3] F. Matloob, T. M. Ghazal, N. Taleb, S. Aftab, M. Ahmad *et al.,* "Software defect prediction using ensemble learning: A systematic literature review," *IEEE Access*, vol. 10, pp. 13123–13143, 2022.

[4] A. Boucher and M. Badri, "Software metrics thresholds calculation techniques to predict fault-proneness: An empirical comparison," *Information and Software Technology*, vol. 96, pp. 38–67, 2018.

[5] L. Chen, B. Fang, Z. Shang and Y. Tang, "Tackling class overlap and imbalance problems in software defect prediction," *Software Quality Journal*, vol. 26, no. 1, pp. 97–125, 2018.

[6] S. Goyal and P. K. Bhatia, "Empirical software measurements with machine learning," in *Computational Intelligence Techniques and Their Applications to Software Engineering Problems*, Boca Raton: CRC Press, pp. 49–64, 2020.

[7] S. Huda, K. Liu, M. Abdelrazek, A. Ibrahim, S. Alyahya *et al.,* "An ensemble oversampling model for class imbalance problem in software defect prediction," *IEEE Access*, vol. 6, pp. 24184–24195, 2018.

[8] H. K. Lee and S. B. Kim, "An overlap-sensitive margin classifier for imbalanced and overlapping data," *Expert Systems with Applications*, vol. 98, pp. 72–83, 2018.

[9] 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, 2018.

[10] R. Özakıncı and A. Tarhan,. "Early software defect prediction: A systematic map and review," *Journal of Systems and Software*, vol. 144, pp. 216–239, 2018.

[11] S. S. Rathore and S. Kumar, "Towards an ensemble based system for predicting the number of software faults," *Expert Systems with Applications*, vol. 82, pp. 357–382, 2017.

[12] S. S. Rathore and S. Kumar, "A study on software fault prediction techniques," *Artificial Intelligence Review*, vol. 51, no. 2, pp. 255–327, 2019.

[13] L. H. Son, N. Pritam, M. Khari, R. Kumar, P. T. M. Phuong *et al.,* "Empirical study of software defect prediction: A systematic mapping," *Symmetry*, vol. 11, no. 2, pp. 2–28, 2019.

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

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

[16] A. Iqbal and S. Aftab, "Prediction of defect prone software modules using mlp based ensemble techniques," *International Journal of Information Technology and Computer Science*, vol. 12, no. 3, pp. 26–31, 2020.

[17] M. S. Daoud, S. Aftab, M. Ahmad, M. A. Khan, A. Iqbal *et al.,* "Machine learning empowered software defect prediction system," *Intelligent Automation & Soft Computing*, vol. 31, no. 32, pp. 1287–1300, 2022.

[18] U. Ali, S. Aftab, A. Iqbal, Z. Nawaz, M. S. Bashir *et al.,* "Software defect prediction using variant based ensemble learning and feature selection techniques," *International Journal of Modern Education & Computer Science*, vol. 12, no. 5, pp. 29–40, 2020.

[19] A. Iqbal, S. Aftab, U. Ali, Z. Nawaz, L. Sana *et al.,* "Performance analysis of machine learning techniques on software defect prediction using nasa datasets," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 5, pp. 300–308, 2019.

[20] R. Mahajan, S. K. Gupta and R. K. Bedi, "Design of software fault prediction model using br technique," *Procedia Computer Science*, vol. 46, pp. 849–858, 2015.

[21] M. Shepperd, Q. Song, Z. Sun and C. Mair, "Data quality: Some comments on the nasa software defect datasets," *IEEE Transactions on Software Engineering*, vol. 39, no. 9, pp. 1208–1215, 2013.

[22] Y. J. Cruz, M. Rivas, R. Quiza, A. Villalonga, R. E. Haber *et al.,* "Ensemble of convolutional neural networks based on an evolutionary algorithm applied to an industrial welding process," *Computers in Industry*, vol. 133, pp. 103530–103538, 2021.

[23] M. Shahhosseini, G. Hu and H. Pham, "Optimizing ensemble weights and hyperparameters of machine learning models for regression problems," *Machine Learning with Applications*, vol. 7, pp. 100251–100260, 2022.

[24] A. U. Rahman, S. Abbas, M. Gollapalli, R. Ahmed, S. Aftab *et al.,* "Rainfall prediction system using machine learning fusion for smart cities," *Sensors*, vol. 22, no. 9, pp. 3504–3519, 2022.

[25] S. Goyal and P. K. Bhatia, "Comparison of machine learning techniques for software quality prediction," *International Journal of Knowledge and Systems Science*, vol. 11, no. 2, pp. 20–40, 2020.

[26] A. O. Balogun, F. B. L. Balogun, H. A. Mojeed, V. E. Adeyemo, O. N. Akande *et al.,* "Smote-based homogeneous ensemble methods for software defect prediction," in *22nd Int. Conf. on Computational Science and its Applications*, Cagliari, Italy, pp. 615–631, 2022.

[27] T. T. Khuat and M. H. Le, "Evaluation of sampling-based ensembles of classifiers on imbalanced data for software defect prediction problems," *SN Computer Science*, vol. 1, no. 2, pp. 1–16, 2020.

[28] S. Goyal and P. K. Bhatia, "Heterogeneous stacked ensemble classifier for software defect prediction," *Multimedia Tools and Applications*, vol. 81, pp. 37033–37055, 2022.

[29] P. Kumudha and R. Venkatesan, "Cost-sensitive radial basis function neural network classifier for software defect prediction," *the Scientific World Journal*, vol. 2016, pp. 1–20, 2016.

[30] A. S. Abdou and N. R. Darwish, "Early prediction of software defect using ensemble learning: A comparative study," *International Journal of Computer Applications*, vol. 179, no. 46, pp. 29–40, 2018.