mathematics

MDPI

Article

A Cloud-Based Software Defect Prediction System Using Data and Decision-Level Machine Learning Fusion

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

1 School of Computer Science, National College of Business Administration and Economics, Lahore 54000, Pakistan
2 Department of Computer Science, Virtual University of Pakistan, Lahore 54000, Pakistan
3 Center for Cyber Security, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia (UKM), Bangi 43600, Selangor, Malaysia
4 College of Computer and Information Technology, American University in the Emirates, Dubai Academic City, Dubai 503000, United Arab Emirates
5 College of Engineering and IT, University of Dubai, Dubai 14143, United Arab Emirates
6 Center for Cyber Physical Systems, EECS Dept, Khalifa University, Abu Dhabi 127788, United Arab Emirates
7 Department of Software, Faculty of Artificial Intelligence and Software, Gachon University, Seongnam 13120, Republic of Korea
* Correspondence: chan.yeun@ku.ac.ae (C.Y.Y.); adnan@gachon.ac.kr (M.A.K.)

Abstract: This research contributes an intelligent cloud-based software defect prediction system using data and decision-level machine learning fusion techniques. The proposed system detects the defective modules using a two-step prediction method. In the first step, the prediction is performed using three supervised machine learning techniques, including naïve Bayes, artificial neural network, and decision tree. These classification techniques are iteratively tuned until the maximum accuracy is achieved. In the second step, the final prediction is performed by fusing the accuracy of the used classifiers with a fuzzy logic-based system. The proposed fuzzy logic technique integrates the predictive accuracy of the used classifiers using eight if–then fuzzy rules in order to achieve a higher performance. In the study, to implement the proposed fusion-based defect prediction system, five datasets were fused, which were collected from the NASA repository, including CM1, MW1, PC1, PC3, and PC4. It was observed that the proposed intelligent system achieved a 91.05% accuracy for the fused dataset and outperformed other defect prediction techniques, including base classifiers and state-of-the-art ensemble techniques.

Keywords: machine learning; software defect prediction; data fusion; machine learning fusion; fuzzy system

MSC: 68N30

check for updates

Citation: Aftab, S.; Abbas, S.; Ghazal, T.M.; Ahmad, M.; Hamadi, H.A.; Yeun, C.Y.; Khan, M.A. A Cloud-Based Software Defect Prediction System Using Data and Decision-Level Machine Learning Fusion. *Mathematics* **2023**, *11*, 632. https://doi.org/10.3390/math11030632

Academic Editors: Vassilis C. Gerogiannis, Dipak Kumar Jana and Petr Dostál

Received: 19 November 2022
Revised: 18 January 2023
Accepted: 20 January 2023
Published: 26 January 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

An exponent increase in the use of smart computing devices has been observed during the last few years due to the availability of high-speed internet at a lower cost. Nowadays, the demand for automated online software systems is increasing steadily, which has triggered the need for high-quality software applications at a lower cost. Testing is the most expensive activity in the software development process, and plays a key role in the quality assurance process by ensuring that the end product is bug-free [1].

Many researchers in the software engineering community are working to reduce the cost of development by focusing on cost-effective testing methods [2–4]. The cost of testing can be significantly decreased if the faulty software modules (defective modules) are identified before the testing stage [1–3,5]. A software module is considered as defective when it produces an error during execution, or does not produce the expected results.

Software defect prediction (SDP) is the process used to predict the defective modules; it can also reduce the testing cost. Such a prediction can guide the quality assurance team, enabling them to focus on defective modules during testing, through which the costs of testing for non-defective modules can be saved [1,5–7]. In the modern era, all of our day-to-day activities directly or indirectly include interactions with software systems, especially since the recent COVID-19 pandemic, which has urged us to transition towards online systems. Therefore, an effective and efficient software defect prediction system must form part of the modern software development paradigm in order to achieve high-quality software with lower costs [8–10].

This research contributes an intelligent cloud-based system for SDP using data and decision-level machine learning fusion techniques. The proposed fusion-based software defect prediction system (FSDPS) incorporates two fusion modules: data fusion and decision-level machine learning fusion. The data fusion approach renders the proposed SDP system more robust, enabling it to work effectively with the diverse datasets extracted from multiple sources. This approach can also resolve the issue of training with limited datasets. Decision-level machine learning fusion involves the integration of the predictive accuracy of three supervised classifiers, including naïve Bayes (NB), artificial neural network (ANN), and decision tree (DT). In this approach, the prediction is performed using classification techniques, in which iterative tuning is performed until the maximum accuracy is achieved for each classifier. The accuracies of the optimized classification models are then integrated using a fuzzy logic-based technique for an effective performance. The proposed fuzzy logic-based fusion method integrates the predictions of the used classifiers by following eight if–then fuzzy rules. These rules were developed by analyzing the prediction accuracy of each of the used classification techniques. The cloud storage was used to store the fused prediction model so that it could be accessed from anywhere. This strategy can also be an aid in the paradigm of global software development. Five datasets from NASA's cleaned repository, including CM1, MW1, PC1, PC3, and PC4, were integrated using instance-level fusion in order to implement the proposed system. The results show that the proposed FSDPS outperforms the other techniques.

This paper is organized as follows. Section 2 provides a summary of the related studies. Section 3 proposes the FSDPS and discusses its stages and activities in detail. Section 4 discusses the detailed results of the proposed system after its implementation. Section 5 presents the threats to the validity of the proposed research. Section 5 concludes this research, together with the directions for future work.

2. Literature Review

Researchers have been working to reduce development costs by identifying faulty software modules before the testing stage. Some related studies are discussed in this section.

The authors of [11] proposed a cloud-based framework for SDP. They explored four training functions in ANN using the back-propagation method. The training functions compared in the proposed framework included Bayesian regularization (BR), scaled conjugate gradient (SCG), Broyden–Fletcher–Goldfarb–Shanno Quasi-Newton (BFGS-QN), and Levenberg–Marquardt (LM). A fuzzy logic-based engine was also incorporated to identify which training function performed better. The cleaned versions of NASA datasets were used by the researchers for the experiments, along with multiple performance measures. It was observed that the BR training function showed a higher accuracy as compared to the other functions. The authors of [12] proposed a framework for SDP using feature selection and ensemble machine learning approaches. In ensemble learning, multiple variants of each classification technique are generated by optimizing various parameters, and then the best-performing variants are integrated using ensemble learning methods. However, the used feature selection method reduced the feature set by removing the metrics not participating in the classification process. NASA defect datasets were used to implement the proposed method, which showed a higher performance as compared to the other techniques.

The authors of [13] presented a classification framework for the detection of faulty software modules before the testing stage. The proposed framework used an ensemble machine learning-based classification model with the multi-layer perceptron (MLP) technique. The proposed framework detected the faulty modules in three dimensions. This was achieved firstly by tuning the MLP until the maximum accuracy was achieved; secondly, the tuned version of MLP was ensembled with the bagging technique; and thirdly, the tuned version was ensembled with the boosting technique. To implement the proposed ensemble machine learning-based classification framework, cleaned versions of NASA's software defect datasets were used. The performance was compared with the techniques known from published research. In [14], the researchers presented a novel feature selection technique for SDP. They proposed a feature selection and ANN-based framework to limit the testing costs in SDLC. They used MLP architecture along with the oversampling method in order to tackle the class imbalance in the dataset. To implement the proposed framework, clean versions of software defect datasets from the NASA repository were used, and various statistical measures are used to assess the performance. The results indicated that the proposed technique performed well, especially with the oversampling technique.

In [15], the researchers used a hybrid classification technique for SDP, which integrated NB and ANN. For its implementation, five benchmark datasets were used, including KC1, KC2, CM1, JM1, and PC1. The proposed technique performed better when compared with NB, ANN, and SVM. The authors of [16] predicted software defects using various supervised machine learning techniques. The SMOTE technique was used by the researchers to resample the data, along with the feature selection method for dimensionality reduction. The experiments were performed on two widely used datasets from the PROMISE repository, KC1 and JM1. The results indicated that RF performed better when compared to the other techniques, with the best results obtained when boosting with RF and bagging with DT. The authors of [17] proposed an enhanced wrapper-based feature selection method which selects the features in an iterative manner. For the prediction, DT and NB were used after tuning, and the experiments were performed on 25 benchmark datasets for a detailed analysis. The performance of the proposed feature selection technique with the used classification methods was analyzed using three measures, including the AUC, F-measure, and accuracy. The results showed that the proposed enhanced feature selection technique performed better than the other methods and selected fewer features with a lower computational cost and high accuracy. In [18], the effectiveness of an ensemble of classification techniques for SDP was discussed. The researchers developed two approaches in this research. In the first approach, the classification is performed using base classifiers, including the k-nearest neighbor (k-NN), DT, and NB. In the second approach, ensembles are used for classification, and the results indicated that the ensemble approach has a tendency to perform better than the other classification techniques. The experiments were performed on 21 benchmark software defect datasets. The authors of [19] presented an integrated technique to predict the workload for the next time slot in distributed clouds. The proposed technique integrates the Savitzky–Golay filter and wavelet decomposition with stochastic configuration networks. The researchers highlighted the significance of the effective and efficient services that could be provided by distributed cloud data centers after the implementation of the proposed technique.

Table 1 presents a summary of the literature review. It shows the proposed techniques for SDP, the dataset repository from which the datasets were extracted for experiments, the names of the used datasets, and the performance measures which were used for the performance analysis.

Table 1. Literature review summary.

Reference	Prediction Technique	Dataset Repository	Datasets	Performance Measures
Daoud, M. S. et al. [11]	Four training functions of back propagation in ANN are used for SDP. The fuzzy logic-based technique is proposed for the identification of the best training function.	NASA	CM1, JM1, KC1, KC3, MC1, MC2, MW1, PC1, PC2, PC3, PC4, PC5	Specificity, precision, F-measure, recall, accuracy, AUC, R2, MSE
Ali, U. et al. [12]	A metric selection-based variant ensemble machine learning technique is proposed for software defect prediction.	NASA	JM1, KC1, PC4, PC5	F-measure, accuracy, MCC
Iqbal, A. et al. [13]	An ANN-based ensemble machine learning technique is proposed for SDP.	NASA	KC1, MW1, PC4, PC5	F-measure, accuracy, AUC, MCC
Iqbal, A. et al. [14]	A multi-filter feature selection technique is used with ANN for SDP.	NASA	CM1, JM1, KC1, KC3, MC1, MC2, MW1, PC1, PC2, PC3, PC4, PC5	F-measure, accuracy, AUC, MCC
Arasteh, B. et al. [15]	Proposes a technique by integrating the ANN and NB for SDP.	PROMISE	KC1, KC2, CM1, PC1, JM1	Accuracy, precision
Alsaeedi, A. et al. [16]	Various supervised classification techniques are used for SDP, including: SVM, DT, RF, bagging, and boosting. The SMOTE technique is used to tackle the issue of class imbalance.	NASA	PC1, PC2, PC3, PC4, PC5, MC1, MC2, JM1, MW1, KC3	Accuracy, precision, F-measure, recall, true-positive rate, false-positive rate, probability of false alarm, specificity, G-measure
Balogun, A. O. et al. [17]	An enhanced wrapper feature selection technique is proposed. The proposed technique is used with NB and DT.	PROMISE, NASA, AEEEM,	EQ, JDT, ML PDE, CM1, KC1, KC2, KC3, MW1, PC1, PC3, PC4, PC5, ANT, CAMEL, JEDIT, REDKITOR, TOMCAT, VELOCITY, XALAN, SAFE, ZXING, APACHE, ECLIPSE, SWT	Accuracy, F-measure, AUC
Alsawalqah, H. et al. [18]	Heterogeneous ensemble classifiers are used for SDP.	PROMISE, NASA,	PC1, PC2, PC3, PC4, PC5, KC1, KC3, CM1, JM1, MC1, MW1, ant-1.7, camel-1.6, ivy-2.0, jedit-4.3, log4j-1.2, ucene-2.4, poi-3.0, tomcat-6, xalan-2.6, xerces-1.4	Precision, recall, G-mean

To the body of previously published work, this research contributes an intelligent system using data and decision-level machine learning fusion to detect defect-prone software modules. The major contributions of the proposed framework are discussed below.

1. Data fusion was performed, through which the developed classification models were rendered more robust and effective for the test datasets. The proposed system was implemented on a fused dataset, which was generated by fusing publicly available defect prediction datasets from NASA's repository, including CM1, MW1, PC1, PC3, and PC4.
2. The prediction accuracy of three classifiers, including NB, ANN, and DT, was integrated using a fuzzy logic technique. The proposed framework used eight fuzzy logic-based if–then rules for decision-level accuracy fusion.

3. The performance of the proposed fusion-based intelligent system was compared with that of other state-of-the-art defect prediction systems, and it was observed that the proposed system outperformed the other methods and achieved a 91.05% accuracy for the fused dataset.

3. Materials and Methods

This research presents an intelligent SDP system using data and decision-level machine learning fusion techniques (Figure 1). There are two layers in the proposed FSDPS: training and testing. Each of the two layers further consist of several stages. The training layer consists of four stages: (1) data fusion, (2) data pre-processing, (3) classification, and (4) fusion. This layer involves the development of the fused classification model by integrating the predictive accuracy of NB, ANN, and DT. The test layer consists of one stage, namely prediction. This stage involves the classification of the software module as defective or non-defective using the fused model.

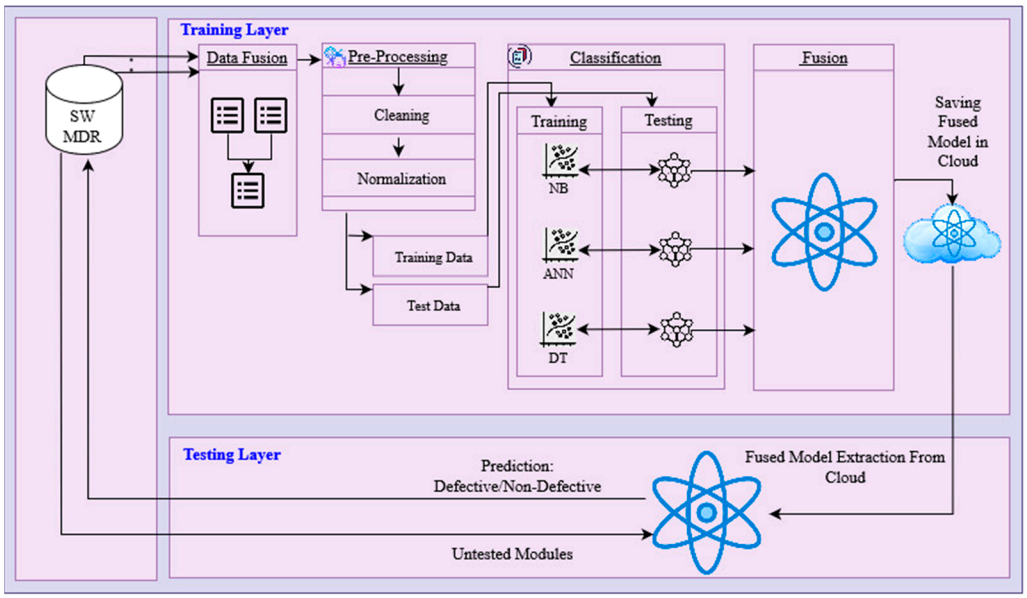


Figure 1. Proposed FSDPS architecture.

The workflow of the training layer begins with the data fusion stage, in which multiple datasets are extracted from the software metric dataset repository (SWMDR) and then integrated using instance-level fusion. The prediction model, which is trained on the fused dataset, is more effective and robust for the test datasets, which are extracted from multiple resources. In this research, five widely used, cleaned datasets from the NASA repository were selected for fusion [20], including CM1, MW1, PC1, PC3, and PC4. These datasets are available in [21]. There are, in total, 38 attributes and 3579 instances in the fused dataset. Each of the selected datasets represents one software component, and the instances in the dataset reflect the software modules. The features represent the software quality metrics, which are recorded during development. One of these 38 features of the fused dataset is the output class to be predicted, whereas the other 37 features are used for the prediction. The output class reflects whether the particular module is defective or not.

The second stage of the training layer is data pre-processing, which is responsible for performing three activities using the fused dataset: (1) cleaning, (2) normalization, and (3) splitting. The data cleaning activity in the pre-processing stage handles the missing values using the mean imputation method. Missing and null values in the attributes can lead to false results. Normalization is the second activity in the pre-processing stage; it involves the transformation of the attribute values into a specific range. The activities of cleaning and normalization both simplify the data so as to help the classification framework

to work effectively and efficiently. The data splitting activity involves the division of the data into training and test sets, following the class split rule, with a 70:30 ratio.

Classification is the third stage of the training layer; it is responsible for classifying the modules as defective or non-defective. The input of this stage is the pre-processed training and testing datasets. For the classification, three techniques in the family of supervised machine learning are used, including NB, ANN, and DT. During the development of the classification model, the classifiers are optimized repeatedly until the maximum accuracy is achieved. First, the classification model is developed using the training data, and the optimization is iteratively performed until the maximum accuracy is achieved using the test data. For NB, the default parameters are used, as their optimization decreases the performance. In ANN, two hidden layers are used, with 33 neurons in each layer. In this study, the initial learning rate value was 0.01; however, the highest performance was achieved with 0.02. DT was tuned by setting the value of the confidence factor to 0.3. However, during this stage, the default values of the remaining parameters are used. This stage finishes by producing the classification models of NB, ANN, and DT.

The decision-level fusion is the fourth and last stage of the training layer. This stage involves the fusion of the optimized classification models using fuzzy logic. Fuzzy rules are used to generate the membership functions through which the prediction accuracies of the used machine learning techniques are integrated for a higher performance. These rules are developed by carefully analyzing the performance of each of the classifiers used on the test dataset. The fusion stage finishes by storing the fused model in the cloud for later use. As compared to server storage, cloud storage was selected here due to its many advantages, including its easy access and security. Moreover, the strategy of cloud storage can be helpful in global software development, as in this case, the fused model will be easily accessible from anywhere.

The if–then conditions based on fuzzy rules are listed below:

IF (naïve Bayes predicts defective, neural network predicts defective and decision tree predicts defective) THEN (the module is defective).

IF (naïve Bayes predicts defective, neural network predicts defective and decision tree predicts non-defective) THEN (module is defective).

IF (naïve Bayes predicts defective, neural network predicts non-defective and decision tree predicts defective) THEN (module is defective).

IF (naïve Bayes predicts non-defective, neural network predicts defective and decision tree predicts defective) THEN (module is defective).

IF (naïve Bayes predicts non-defective, neural network predicts non-defective and decision tree also predicts non-defective) THEN (module is not defective).

IF (naïve Bayes predicts defective, neural network predicts non-defective and decision tree predicts non-defective) THEN (module is not defective).

IF (naïve Bayes predicts non-defective, neural network predicts non-defective and decision tree predicts defective) THEN (module is not defective).

IF (naïve Bayes predicts non-defective, neural network predicts defective and decision tree predicts non-defective) THEN (module is not defective).

The membership functions developed by following the if–then fuzzy rules are shown in Table 2. These membership functions are used to integrate the accuracy of NB, ANN, and DT. These if–then rules, which work as the base of membership functions, were developed after various experiments on the fusion of the predictive accuracy of the used classifiers.

Table 2. Membership functions of the proposed fuzzy logic-based fusion technique.

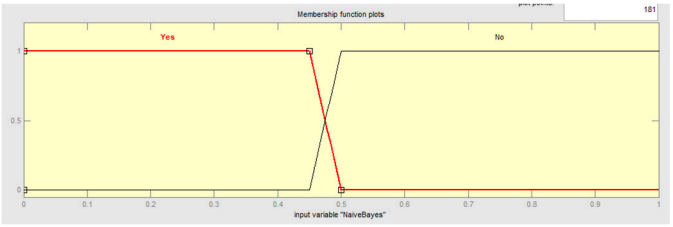
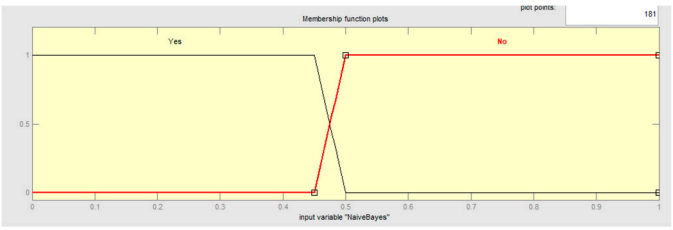
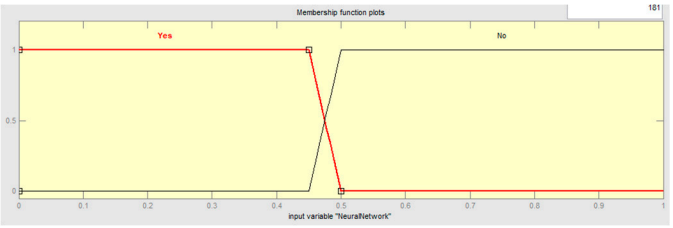
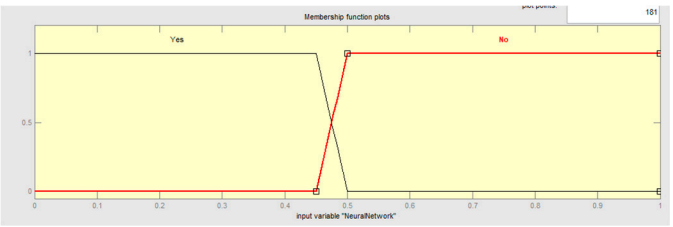
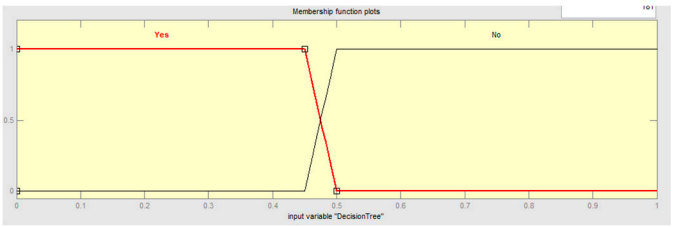
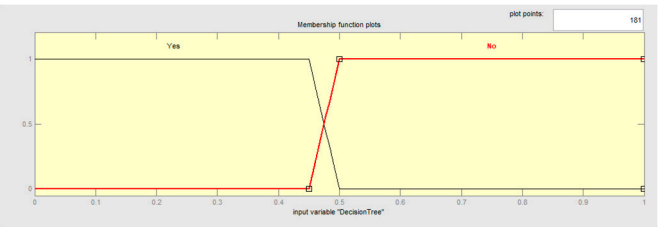
Membership Functions	Graphical Representation
$\gamma_{\mathcal{NBY}}(nb) = \left\{ \max\left(\min\left(1, \frac{0.5 - nb}{0.05}\right), 0\right)\right\}$	
$\gamma_{\mathcal{NBN}}(nb) = \left\{ \max\left(\min\left(\frac{nb - 0.45}{0.05}, 1\right), 0\right)\right\}$	
$\gamma_{\mathcal{NNY}}(nn) = \left\{ \max\left(\min\left(1, \frac{0.5 - nn}{0.05}\right), 0\right)\right\}$	
$\gamma_{\mathcal{NNN}}(nn) = \left\{ \max\left(\min\left(\frac{nn - 0.45}{0.05}, 1\right), 0\right)\right\}$	
$\gamma_{\mathcal{DTY}}(dt) = \left\{ \max\left(\min\left(1, \frac{0.5 - dt}{0.05}\right), 0\right)\right\}$	
$\gamma_{\mathcal{DTN}}(dt) = \left\{ \max\left(\min\left(\frac{dt - 0.45}{0.05}, 1\right), 0\right)\right\}$	

Table 2. *Cont.*

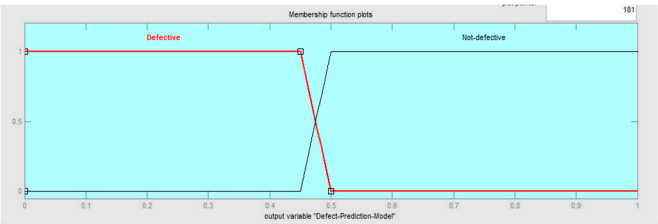
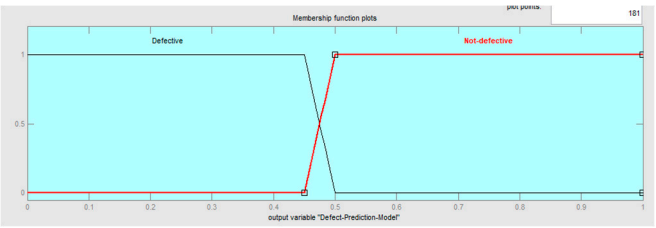
Membership Functions	Graphical Representation
$\gamma_{\mathcal{DY}}(\underline{d}) = \left\{ \max\left(\min\left(1, \frac{0.5-\underline{d}}{0.05}\right), 0\right)\right\}$	
$\gamma_{\mathcal{DN}}(\underline{d}) = \left\{ \max\left(\min\left(\frac{\underline{d}-0.45}{0.05}, 1\right), 0\right)\right\}$	

Figure 2 shows the rule surface of the proposed fuzzy logic-based fusion technique for defect prediction with respect to the NB and ANN results. Figure 3 shows the prediction process with the accuracy fusion technique, which predicts that the software module is non-defective. NB predicts that the module is non-defective with a 0.127 confidence factor, and ANN predicts the same with a 0.259 confidence factor; DT predicts that the module is defective with a 0.801 confidence factor. However, as per the defined fuzzy rules, the proposed technique predicts that the module is non-defective with a 0.248 confidence factor.

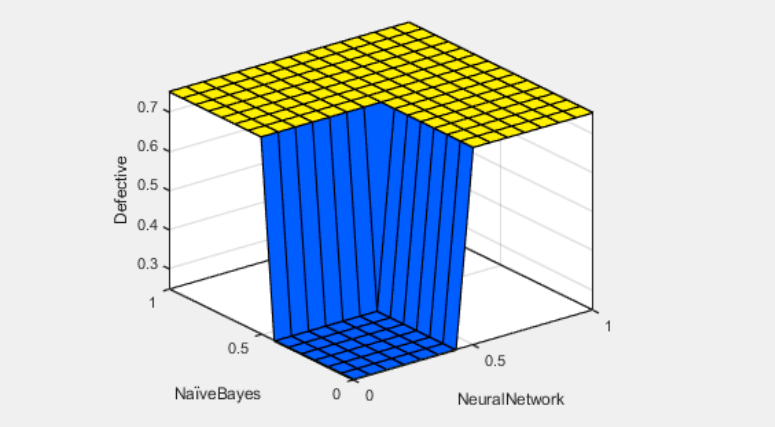


Figure 2. Rule surface of the proposed fuzzy logic-based fusion technique with NB and ANN.

It is demonstrated in Figure 4 that NB predicts that the module is non-defective with a 0.127 confidence factor, whereas ANN and DT both predict that the module is defective with 0.62 and 0.801 confidence factors, respectively; therefore, that the proposed fused model predicts that the module is defective with a 0.752 confidence factor.

The second layer of the proposed system is the testing layer, which performs real-time prediction to identify which software module is defective and requires extensive testing. This layer involves four activities. The first activity is the extraction of the dataset of the untested software module for prediction. The second activity is the extraction of the fused classification model that was saved in the cloud in the last activity of the training layer. The third activity is the prediction, in which the data of the untested software component function is used as the input to the fused model, and the output is extracted; this indicates

whether the module is defective or not. The fourth and last activity of this layer is the submission of the prediction to the software defect dataset repository.



Figure 3. Results of the proposed fuzzy logic based fusion technique for a non-defective module (0).

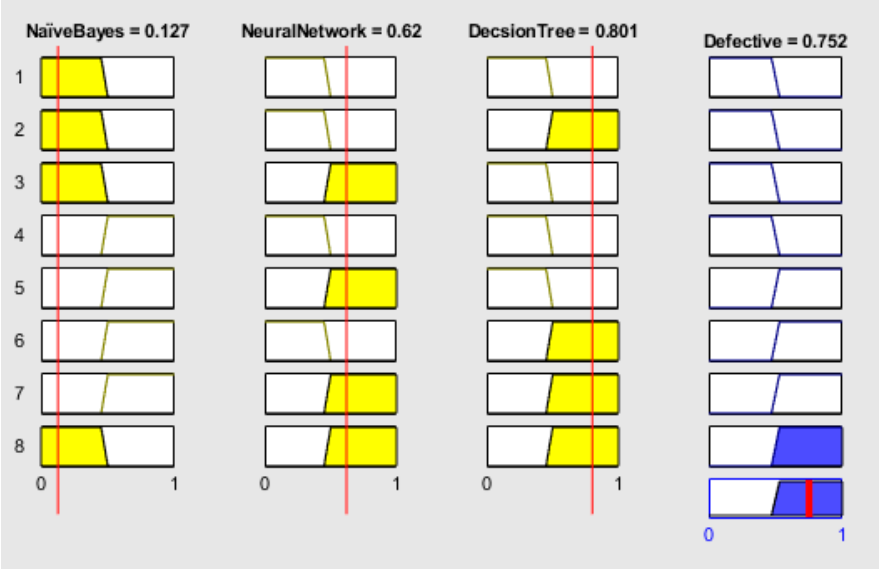


Figure 4. Results of the proposed fuzzy logic-based fusion technique for a defective module (1).

4. Results and Discussion

The proposed FSDPS was implemented using a fused software defect dataset. Matlab 2021a was used in this research to conduct the experiments and simulations. The fused dataset was created by integrating the five datasets from NASA's cleaned repository, named CM1, MW1, PC1, PC3, and PC4. The fused dataset consists of 3579 instances, of which 428 indicate that the modules are defective, whereas 3151 indicate that they are non-defective. In the pre-processing stage, the used dataset underwent cleaning and normalization processes and was then divided into two further subsets, the training set and test set, using a 70:30 ratio. The training dataset consists of 2506 instances, and the test dataset consists of 1073 instances. For the prediction, initially, three supervised machine learning techniques were used, including NB, ANN, and DT. Each classifier was optimized

so that we could obtain maximum accuracy. To analyze the performance of the proposed fusion-based software defect prediction system, the following measures were used [22,23].

$$Accuracy = \frac{(\partial OR_0/\epsilon OR_0 + \partial OR_1/\epsilon OR_1)}{\epsilon OR_0 + \epsilon OR_1} \quad (1)$$

$$Positive\ Prediction\ Value = \frac{\partial OR_1/\epsilon OR_1}{(\partial OR_1/\epsilon OR_1 + \partial OR_0/\epsilon OR_1)} \quad (2)$$

$$Negative\ Prediction\ Value = \frac{\partial OR_0/\epsilon OR_0}{(\partial OR_0/\epsilon OR_0 + \partial OR_1/\epsilon OR_0)} \quad (3)$$

$$Specificity = \frac{\partial OR_0/\epsilon OR_0}{(\partial OR_0/\epsilon OR_0 + \partial OR_0/\epsilon OR_1)} \quad (4)$$

$$Sensitivity = \frac{\partial OR_1/EOR_1}{(\partial OR_1/\epsilon OR_0 + \partial OR_1/\epsilon OR_1)} \quad (5)$$

$$False\ Positive\ Ratio = 1 - Specificity \quad (6)$$

$$False\ Negitive\ Ratio = 1 - Senstivity \quad (7)$$

$$Likelihood\ Ratio\ Positive = \frac{Sensitivity}{(1 - Specificity)} \quad (8)$$

$$Likelihood\ Ratio\ Negative = \frac{(1 - Sensitivity)}{Specificity} \quad (9)$$

In the formulas shown above, ∂OR_0 reflects the predicted non-defective modules, and ∂OR_1 reflects the predicted defective modules, whereas ϵOR_0 reflects the expected non-defective modules, and ϵOR_1 reflects the expected defective modules.

To train the NB classifier, the reserved training dataset consisting of 2506 instances was used. During the training process, 1948 instances were classified as negative out of 2206 instances, whereas 107 instances were classified as positive out of 300 instances. After analyzing and comparing the output result and expected result in Table 3, we achieved 82% accuracy in the training process with NB. During the process of testing with NB, 872 instances were predicted as negative out of 945, whereas 22 instances were predicted as positive out of 128. The comparison of the expected result and output result in Table 4 reflects that 83.32% accuracy was achieved during testing with NB.

Table 3. NB training.

N = 2506 (No. of Records)		**Predicted Result ∂OR_0, ∂OR_1**	
INPUT	Expected output result (ϵOR_0, ϵOR_1)	∂OR_0 (Non-defective -0)	∂OR_1 (Defective -1)
	ϵOR_0 = 2206 (Non-defective -0)	1948	258
	ϵOR_1 = 300 (Defective -1)	193	107

Table 4. NB Testing.

N = 1073 (No. of Records)		**Predicted Result ∂OR_0, ∂OR_1**	
INPUT	Expected output result (ϵOR_0, ϵOR_1)	∂OR_0 (Non-defective -0)	∂OR_1 (Defective -1)
	ϵOR_0 = 945 (Non-defective -0)	872	73
	ϵOR_1 = 128 (Defective -1)	106	22

During the training of ANN, 2100 instances out of 2206 were classified as negative, and 56 instances out of 300 were classified as positive. The training accuracy achieved with ANN was 86.03 % (Table 5). During the testing process, 905 instances were predicted as negative out of 945, and 16 instances were predicted as positive out of 128. The comparison of the expected output and achieved output in Table 6 reflects an 85.83% accuracy.

Table 5. ANN training.

N = 2506 (No. of Records)		**Predicted Result ∂OR_0, ∂OR_1**	
INPUT	Expected output result (ϵOR_0, ϵOR_1)	∂OR_0 (Non-defective -0)	∂OR_1 (Defective -1)
	ϵOR_0 = 2206 (Non-defective -0)	2100	106
	ϵOR_1 = 300 (Defective -1)	244	56

Table 6. ANN Testing.

N = 1073 (No. of Records)		**Predicted Result ∂OR_0, ∂OR_1**	
INPUT	Expected output result (ϵOR_0, ϵOR_1)	∂OR_0 (Non-defective -0)	∂OR_1 (Defective -1)
	ϵOR_0 = 945 (Non-defective -0)	905	40
	ϵOR_1 = 128 (Defective -1)	112	16

In the training process with DT, 2073 instances out of 2206 were classified as negative, and 199 instances out of 300 were classified as positive. The output result and expected result are shown in Table 7. After comparing both the results, we achieved 90.66% accuracy. During the testing process, DT classified 887 records out of 945 records as negative, whereas 26 records out of 128 were classified as positive. The comparison of the expected output and achieved output in Table 8 reflects an accuracy of 85.09%.

Table 7. DT training.

N = 2506 (No. of Records)		**Predicted Result ∂OR_0, ∂OR_1**	
INPUT	Expected output result (ϵOR_0, ϵOR_1)	∂OR_0 (Non-defective -0)	∂OR_1 (Defective -1)
	ϵOR_0 = 2206 (Non-defective -0)	2073	133
	ϵOR_1 = 300 (Defective -1)	101	199

Table 8. DT testing.

N = 1073 (No. of Records)		**Predicted Result ∂OR_0, ∂OR_1**	
INPUT	Expected output result (ϵOR_0, ϵOR_1)	∂OR_0 (Non-defective -0)	∂OR_1 (Defective -1)
	ϵOR_0 = 945 (Non-defective -0)	887	58
	ϵOR_1 = 128 (Defective -1)	102	26

Finally, the test data were subjected to the proposed fuzzy system, along with the three predictions provided by the classifiers. The proposed FSDPS classifies the testing data on the basis of the developed fuzzy rules. The fuzzy rules were developed by keeping in mind the achieved accuracy of the classification models for the test data. The proposed fused system classified 935 out of 945 records as negative, whereas 42 instances out of 128 were classified as positive. The expected results are compared with the achieved results in Table 9, which reflects a 91.05% accuracy.

Table 9. Fused testing.

N = 1073 (No. of Records)		**Predicted Result ∂OR_0, ∂OR_1**	
INPUT	Expected output result (ϵOR_0, ϵOR_1)	∂OR_0 (Non-defective -0)	∂OR_1 (Defective -1)
	ϵOR_0 = 945 (Non-defective -0)	935	10
	ϵOR_1 = 128 (Defective -1)	86	42

The detailed results of the used classifiers, along with the result of the proposed fusion-based system, are shown in Table 10. The results of NB, ANN, and DT for the training and test datasets are shown, whereas only the results of the proposed FSDPS for the test dataset are shown. The used classifiers were tuned multiple times until we achieved the maximum accuracy. The proposed fusion-based system outperformed the other used classifiers. The accuracy of NB, ANN and DT for the test datasets was 83.32%, 85.83%, and 85.09%, respectively, whereas the proposed system outperformed all three techniques and achieved a 91.05% accuracy for the test dataset. It can be observed that the proposed fusion-based defect prediction system showed a significantly higher performance, as it integrated the predictive accuracy of all three classifiers using the fuzzy logic technique.

Table 10. ML Algorithm Comparison.

ML Algorithm	**Dataset**	**Accuracy**	**Specificity**	**Sensitivity**	**Positive Prediction Value**	**Negative Prediction Value**	**False Positive Value**	**False Negative Value**	**Likelihood Ratio Positive**	**Likelihood Ratio Negative**
Naïve Bayes	Training	0.8200	0.8830	0.3567	0.2932	0.9099	0.1170	0.6433	3.049	0.8200
	Testing	0.8332	0.9228	0.1719	0.2316	0.8916	0.0772	0.8281	2.225	0.8332
Artificial neural network	Training	0.8603	0.9519	0.1867	0.3456	0.8959	0.0481	0.8133	3.885	0.8603
	Testing	0.8583	0.9577	0.125	0.2857	0.8899	0.0423	0.875	2.953	0.8583
Decision tree	Training	0.9066	0.9397	0.6633	0.5994	0.9535	0.0603	0.3367	11.00	0.9066
	Testing	0.8509	0.9386	0.2031	0.3095	0.8969	0.0614	0.7969	3.310	0.8509
Fused/proposed method	Testing	0.9105	0.9894	0.3281	0.8077	0.9158	0.0106	0.6719	31.01	0.9087

The accuracy of the proposed FSDPS is compared with that of other state-of-the-art software defect prediction techniques in Table 11. It can be observed that the proposed system performed better than the other techniques using the fused dataset, and achieved 91.05% accuracy. Training a classification model on a fused dataset is a complex process, and is considered a challenging task compared to the training of a model on a single-source

dataset. It has been observed that the pattern recognition ability of machine learning methods for prediction can be enhanced by using the fuzzy logic technique [24]. The high accuracy achieved by the proposed system for the fused dataset reflects the effectiveness of fuzzy logic-based machine learning fusion techniques.

Table 11. Performance comparison of the proposed ISDPS with the other techniques.

Prediction Technique	Accuracy (%)
Stacked ensemble [9]	89.10
Fused-ANN-BR [11]	85.45
FS-VEML [12]	84.97
Boosting-OPT-MLP [13]	79.08
MLP-FS [14]	85.13
NB [18]	82.65
ANN [25]	89.96
Tree [25]	84.94
Bagging [26]	80.20
Boosting [26]	81.30
Heterogeneous [27]	89.20
ADBBO-RBFNN [28]	88.65
Bagging LWL [29]	90.10
Proposed FSDPS	91.05

5. Threat to Validity

The threat to validity is a crucial aspect of any proposed research. According to [30], it is important to explicitly analyze and mitigate threats to the validity of the proposed solution.

External validity: This type of validity analyzes whether the proposed solution is equally effective for other datasets belonging to the same problem canvas. In this study, five widely used benchmark software defect datasets, including CM1, MW1, PC1, PC3, and PC4, were fused to implement the proposed FSDPS. The datasets were taken from NASA's cleaned software defect repository. All of the five datasets have the same attributes, which is necessary for instance-level fusion. The conclusion of this study cannot be generalized to other defect datasets. However, the comprehensive experimental setup, along with the iterative parameter tuning used in this study, can be adopted by other researchers using other datasets.

Internal validity: This form of validity analyzes whether the selected prediction techniques are good enough for the selected datasets or for other datasets used to address the same problem. According to [31], various factors, including the datasets, prediction techniques, and software tools, can affect the internal validity of a software defect prediction system. In this study, three supervised classification techniques were used in the proposed FSDPS, including NB, ANN, and DT. These techniques were selected on the basis of their heterogeneity and performance. Moreover, a fuzzy logic-based fusion technique was proposed to integrate the predictive accuracy of the used classification techniques. In future studies, researchers could use other classification algorithms with different fuzzy logic techniques.

Construct validity: This form of validity concerns the selection of the performance measures that are used to analyze the performance of the proposed system. In this research, various performance measures were calculated, including the accuracy, specificity, sensitivity, positive prediction value, negative prediction value, false-positive value, false-negative value, likelihood ratio positive, and likelihood ratio negative. However, among all of the

calculated performance measures, the accuracy was used to compare the performance of the proposed FSDPS with respect to the other techniques.

6. Conclusions and Future Work

Software defect prediction involves the detection of faulty modules before the testing stage so that only defect-prone modules will be subjected to testing. An effective defect prediction system can decrease software development costs by limiting the effort involved in quality assurance activities in the testing phase. In this research, we proposed a system for software defect prediction using data and decision-level machine learning fusion techniques. The proposed system fused the predictive accuracy of three supervised classifiers: NB, ANN, and DT. The accuracy was fused using fuzzy logic-based if–then rules. To empirically evaluate the proposed system, five cleaned software defect datasets from NASA's repository were integrated using instance-level fusion. The datasets which were fused for the experiments included CM1, MW1, PC1, PC3, and PC4. The experiments reflected the higher accuracy of the proposed fusion-based defect prediction system as compared to the other techniques. The proposed system outperformed the other techniques, which reflects the effectiveness and robustness of the proposed decision-level fusion technique. In future work, a feature selection technique should be incorporated into the system for a cost-effective solution. Ensemble machine learning should also be considered for the decision-level fusion. Moreover, workload prediction for the next time slot should also be performed so as to render cloud data services effective and efficient in software defect prediction.

Author Contributions: S.A. (Shabib Aftaband), S.A. (Sagheer Abbas) and M.A. fused the data, performed the analysis, and conducted the experiments. S.A. (Shabib Aftaband), M.A. and T.M.G. prepared the original draft. H.A.H., M.A.K. and M.A. performed the detailed review and editing. C.Y.Y. and M.A.K. performed the supervision. T.M.G., S.A. (Shabib Aftaband) and T.M.G. drafted the pictures and tables. S.A. (Shabib Aftaband), C.Y.Y., H.A.H. and M.A.K. performed the revision and improved the quality of the draft. All authors have read and agreed to the published version of the manuscript.

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

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The simulation files/data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Suresh Kumar, P.; Behera, H.S.; Nayak, J.; Naik, B. Bootstrap aggregation ensemble learning-based reliable approach for software defect prediction by using characterized code feature. *Innov. Syst. Softw. Eng.* **2021**, *17*, 355–379. [CrossRef]
2. Balogun, A.O.; Basri, S.; Abdulkadir, S.J.; Hashim, A.S. Performance analysis of feature selection methods in software defect prediction: A search method approach. *Appl. Sci.* **2019**, *9*, 2764. [CrossRef]
3. Balogun, A.O.; Basri, S.; Mahamad, S.; Abdulkadir, S.J.; Capretz, L.F.; Imam, A.A.; Almomani, M.A.; Adeyemo, V.E.; Kumar, G. Empirical analysis of rank aggregation-based multi-filter feature selection methods in software defect prediction. *Electronics* **2021**, *10*, 179. [CrossRef]
4. Huda, S.; Alyahya, S.; Ali, M.M.; Ahmad, S.; Abawajy, J.; Al-Dossari, H.; Yearwood, J. A framework for software defect prediction and metric selection. *IEEE Access* **2017**, *6*, 2844–2858. [CrossRef]
5. Song, Q.; Jia, Z.; Shepperd, M.; Ying, S.; Liu, J. A general software defect-proneness prediction framework. *IEEE Trans. Softw. Eng.* **2010**, *37*, 356–370. [CrossRef]
6. Zhang, Q.; Ren, J. Software-defect prediction within and across projects based on improved self-organizing data mining. *J. Supercomput.* **2021**, *78*, 6147–6173. [CrossRef]
7. Ibrahim, D.R.; Ghnemat, R.; Hudaib, A. Software defect prediction using feature selection and random forest algorithm. In Proceedings of the International Conference on New Trends in Computer Science, Amman, Jordan, 11–13 October 2017; pp. 252–257.

8. Mahajan, R.; Gupta, S.; Bedi, R.K. Design of software fault prediction model using br technique. *Procedia Comput. Sci.* **2015**, *46*, 849–858. [CrossRef]
9. Goyal, S.; Bhatia, P.K. Heterogeneous stacked ensemble classifier for software defect prediction. *Multimed. Tools Appl.* **2021**, *81*, 37033–37055. [CrossRef]
10. Mehta, S.; Patnaik, K.S. Stacking based ensemble learning for improved software defect prediction. In *Proceeding of Fifth International Conference on Microelectronics, Computing and Communication Systems*; Springer: Singapore, 2021; pp. 167–178.
11. Daoud, M.S.; Aftab, S.; Ahmad, M.; Khan, M.A.; Iqbal, A.; Abbas, S.; Ihnaini, B. machine learning empowered software defect prediction system. *Intell. Autom. Soft Comput.* **2022**, *31*, 1287–1300. [CrossRef]
12. Ali, U.; Aftab, S.; Iqbal, A.; Nawaz, Z.; Bashir, M.S.; Saeed, M.A. Software defect prediction using variant based ensemble learning and feature selection techniques. *Int. J. Mod. Educ. Comput. Sci.* **2020**, *12*, 29–40. [CrossRef]
13. Iqbal, A.; Aftab, S. Prediction of defect prone software modules using MLP based ensemble techniques. *Int. J. Inf. Technol. Comput. Sci.* **2020**, *12*, 26–31. [CrossRef]
14. Iqbal, A.; Aftab, S. A classification framework for software defect prediction using multi-filter feature selection technique and MLP. *Int. J. Mod. Educ. Comput. Sci.* **2020**, *12*, 42–55. [CrossRef]
15. Arasteh, B. Software fault-prediction using combination of neural network and Naive Bayes algorithm. *J. Netw. Technol.* **2018**, *9*, 95. [CrossRef]
16. Alsaeedi, A.; Khan, M.Z. Software defect prediction using supervised machine learning and ensemble techniques: A comparative study. *J. Softw. Eng. Appl.* **2019**, *12*, 85–100. [CrossRef]
17. Balogun, A.O.; Basri, S.; Capretz, L.F.; Mahamad, S.; Imam, A.A.; Almomani, M.A.; Kumar, G. software defect prediction using wrapper feature selection based on dynamic re-ranking strategy. *Symmetry* **2021**, *13*, 2166. [CrossRef]
18. Alsawalqah, H.; Hijazi, N.; Eshtay, M.; Faris, H.; Radaideh, A.A.; Aljarah, I.; Alshamaileh, Y. Software defect prediction using heterogeneous ensemble classification based on segmented patterns. *Appl. Sci.* **2020**, *10*, 1745. [CrossRef]
19. Bi, J.; Yuan, H.; Zhou, M. Temporal prediction of multiapplication consolidated workloads in distributed clouds. *IEEE Trans. Autom. Sci. Eng.* **2019**, *16*, 1763–1773. [CrossRef]
20. Shepperd, M.; Song, Q.; Sun, Z.; Mair, C. Data quality: Some comments on the NASA software defect datasets. *IEEE Trans. Softw. Eng.* **2013**, *39*, 1208–1215. [CrossRef]
21. NASA Defect Dataset. Available online: https://github.com/klainfo/NASADefectDataset (accessed on 17 September 2022).
22. Ahmed, U.; Issa, G.F.; Khan, M.A.; Aftab, S.; Khan, M.F.; Said, R.A.; Ghazal, T.M.; Ahmad, M. Prediction of diabetes empowered with fused machine learning. *IEEE Access* **2022**, *10*, 8529–8538. [CrossRef]
23. Rahman, A.U.; Abbas, S.; Gollapalli, M.; Ahmed, R.; Aftab, S.; Ahmad, M.; Khan, M.A.; Mosavi, A. Rainfall prediction system using machine learning fusion for smart cities. *Sensors* **2022**, *22*, 3504. [CrossRef]
24. Naeem, Z.; Farzan, M.; Naeem, F. Predicting the performance of governance factor using fuzzy inference system. *Int. J. Comput. Innov. Sci.* **2022**, *1*, 35–50.
25. Goyal, S.; Bhatia, P.K. Comparison of machine learning techniques for software quality prediction. *Int. J. Knowl. Syst. Sci.* **2020**, *11*, 20–40. [CrossRef]
26. Balogun, A.O.; Lafenwa-Balogun, F.B.; Mojeed, H.A.; Adeyemo, V.E.; Akande, O.N.; Akintola, A.G.; Bajeh, A.O.; Usman-Hamza, F.E. SMOTE-based homogeneous ensemble methods for software defect prediction. In *International Conference on Computational Science and Its Applications*; Springer: Cham, Switzerland, 2020; pp. 615–631.
27. Khuat, T.T.; Le, M.H. Evaluation of sampling-based ensembles of classifiers on imbalanced data for software defect prediction problems. *SN Comput. Sci.* **2020**, *1*, 108. [CrossRef]
28. Kumudha, P.; Venkatesan, R. Cost-sensitive radial basis function neural network classifier for software defect prediction. *Sci. World J.* **2016**, *11*, 126–134. [CrossRef] [PubMed]
29. Abdou, A.S.; Darwish, N.R. Early prediction of software defect using ensemble learning: A comparative study. *Int. J. Comput. Appl.* **2018**, *179*, 29–40.
30. Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M.C.; Regnell, B.; Wesslén, A. *Experimentation in Software Engineering*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012.
31. Gao, K.; Khoshgoftaar, T.M.; Seliya, N. Predicting high-risk program modules by selecting the right software measurements. *Softw. Qual. J.* **2012**, *20*, 3–42. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.