Hindawi
Applied Computational Intelligence and Soft Computing
Volume 2022, Article ID 2581832, 16 pages
https://doi.org/10.1155/2022/2581832

*Research Article*

# Software Defect Prediction through Neural Network and Feature Selections

**Mutasem Shabeb Alkhasawneh**

*Software Engineering Department, Faculty of Information and Technology, Ajloun National University, P.O. Box 43, Ajloun 26810, Jordan*

Correspondence should be addressed to Mutasem Shabeb Alkhasawneh; m_sh_ka1@yahoo.com

Received 21 June 2022; Revised 6 August 2022; Accepted 11 August 2022; Published 26 September 2022

Academic Editor: Manikandan Ramachandran



Software failure such as software defect causes billion of dollar loss every year. Software failure also affects billion of people worldwide. Inadequate software testing can cause software failure. To predict the software defect, this study proposed a model consisting of feature selection and classifications. The correlation base method was used for feature selection, and radial base function neural network (RBF) was used for classification. Also, for testing the proposed system, fourteen NASA data sets were used including CM1, JM1, KC1, KC2, KC3, KC4, MC1, MC2, MW1, PC1, PC2, PC3, PC4, and PC5. The data set was divided using the well-known K-cross-validation methods which were performed to divide the data set for training and testing the RBF. The RBF were trained and tested before and after feature selections. Precision, recall, *F*-measure, and accuracy are four methods used to evaluate the performance of the proposed methods. The precision obtained for the fourteen data sets was CM1, 94.01%; JM1, 85.18%; KC1, 83.24%; KC2, 81.27%; KC3, 79.30%; KC4, 85.29%; MC1, 99.89%; MC2, 73.27%; MW1, 90.90%; PC1, 98.79%; PC2, 100%; PC3, 95.67%; PC4, 95.12%; and PC5, 80.89%. Recall was as follows: CM1, 95.78%; JM1, 87.89%; KC1, 86.24%; KC2, 83.82%; KC3, 82.10%; KC4, 86.28%; MC1, 100%; MC2, 76.67%; MW1, 92.09%; PC1, 99.98%; PC2, 100%; PC3, 96.23%; PC4, 95.17%; and PC5, 81.80%. *F*-measure was as follows: CM1, 0.95; JM1, 0.87; KC1, 0.83; KC2, 0.82; KC3, 0.85; KC4, 0.86; MC1, 0.99; MC2, 0.76; MW1, 0.95; PC1, 0.99; PC2, 0.99; PC3, 0.97; PC4, 0.95; and PC5, 0.80. The accuracy obtained was as follows: CM1, 93.99%; JM1, 84.87%; KC1, 83.25%; KC2, 79.11%; KC3, 78.25%; KC4, 83.18%; MC1, 99.01%; MC2, 70.18%; MW1, 88.90%; PC1, 98.99%; PC2, 99.80%; PC3, 94.11%; PC4, 94.4%; and PC5, 79.02%. The proposed method results were compared with the result obtained from different methods. The proposed model obtained better results than other methods for data set CM1, KC4, MC1, PC1, PC2, PC3, PC4, and PC5.

## 1. Introduction

The volume of the software global business in 2020 was estimated by $389.86 billion, and it is also expected to exceed $428.84 billion by 2028 [1]. Software defect has a huge financial impact on software industrial. Software failure causes about $1.7 trillion loses in 2017 [2]. Mars Climate Orbiter, Mariner 1, and Arian5 are examples of expensive software failure that causes $193 million, 18.5 million, and 8 million, respectively. However, testing and quality assurance cost can reach up to 23% of software total cost [3]. Moreover, fixing software faults during the software development will cost 100 times less than fixing the software after deployments [3]. Different terms are used to describe software fails such as fault, defect, bugs, problem, errors, and incident. Developer tends to imply defect term when the software is in really severe condition and may even be dangerous [4]. Indeed, software defect refers to any fault occurred during developing any software.

Software defect problems are addressed by the inspectors who meet to inspect the software. Later on, the programmer and moderator make the change [4]. Inspector needs to spend time, effort, and money to detect the software defect. In addition, the accuracy of detecting the software defect can be affected by numerous factors, one of them is the human errors. Hence, the human error can make the process of

determining software defect more complicated and a high-risk task. Therefore, additional opinion is needed to help inspectors to achieve more accurate software defects. Software defect prediction refers to the process that can predict the defected code and specify the area of the defected code through using classifier [5]. For the last few decades' prediction, the software defect has been the subject of interest for many researchers. Therefore, many automatic software defect prediction models were proposed that make the inspectors and programmers spend more time and money in enhancing the model than predicting the software defects. In addition, the perditions and classification process made by humans have faced many challenges such as human error and low performance, and most of the prediction and classification process was performed through an automated system. Artificial neural networks (ANNs) is the most common method used by researchers for prediction. ANNs find their way in different applications in the real world like medical diagnosis such as heart disease, Wisconsin breast cancer, thyroid disease, and Pima Indian diabetes [6]. Also, ANN is being applied in the agriculture such as planet leaf disease [7], predicting mass of the fruits [8], and olive oil harvesting time [9]; weather forecasting [10]; language recognitions [11]; and many other applications.

All the data set can be used for training and testing the neural network. However, not all the features in the data set have the same importance for prediction the software defect. For instance, in the large dimensionality problem, features with less importance could cause the generalized performances of ANN to deteriorate [12]. Therefore, feature selection tools can be used to determine the features with high importance and to eliminate the feature with less importance [13]. Moreover, different tools can be used to select the feature such as discriminant analysis (DA), principal component analyses (PA), decision tree (T), and multilayer perceptron (MLP). Also, feature selection found many applications worldwide to solve real-world problems such as landslide prediction [14, 15], medical application [16, 17], food industries [18, 19], face recognitions [20, 21], and many other applications [22–24].

The aim of the study can be achieved by answering the following research questions:

(1) How the RBF neural network perform when it used to predict the software defect for fourteen NASA data sets?

(2) How the RBF neural network perform when it used to predict the software defect for fourteen NASA data sets after eliminating nonrelated features?

(3) How the RBF neural network perform when it compares to previous methods that used to predict software defect for fourteen NASA data sets?

(4) How the RBF neural network perform when it compares to previous methods that used to predict the software defect for fourteen NASA data sets after eliminating nonrelated features?

In Section 1, an introduction for this study is introduced. Section 2 summarized the previous work that focused on the prediction methods that used to predict the software defection using the same data set that is used in this study. Section 3 introduced the data set and evaluation methods. Section 5 explained the experimental setup. Result is discussed in Section 6.

## 2. Related Work

A frame work to predict software defect code area is introduced in [5]. The proposed system was performed through two steps: feature selection—using abstract syntax trees (AST) method—and prediction—using conventional neural networks. The system proposed in this study was trained and tested using seven data sets. The data set was imported from PROMISE repository. Also, the system performance was verified using three evaluation metrics including precision, recall, and *F*-measure. The proposed system was trained and tested two times; firstly, with all features and secondly with features that are selected by AST, the system achieved better performance when trained and tested with features that are selected by AST.

Software defect prediction using the Bayesian neural network was proposed in [25]. Nine data sets consisting of nine features obtained from the PROMISE data repository were used to train the Bayesian neural network. Training neural network was performed in two phases, using the data without reducing the number of features in the first phase. While in the second training phase, the number of features was reduced to five features by using two methods of feature selection, namely, Relief (Relief Fat tribute Eval with Ranker search method) and Subset Eval attribute evaluator with best first search method. Meanwhile, area under curve (AUC) was the evaluation performance tool. Bayesian neural network with data after feature selection has shown better performance over the performance achieved by using data with all features for training.

Deep belief network (DBN) was used to predict the defect in the software if exists. The DBN performance was evaluated using precision, recall, and *F*1-measure. As expected, DBN achieved better results when trained after eliminating unimportant features. A ten Java bench mark data set obtained from the PROMISE repository in [26] was used to predict the software defections. In addition, the abstract syntax trees (ASTs) were used as feature selection tool to select the most important factors that can improve the software defect prediction. DBN trained with data after selecting the most important feature achieved better results.

Two classifiers: random forests (RF), logistic regression (LR), and standard data sets obtained from the PROMISE repository was used in [27] to predict the software defect. The data set contains ten features that contribute to determining the place of defect in the code. In a study [27], they proposed a model to predict the software defect which reduced the number of features using the abstract syntax before entering the RF and LR. Results were evaluated by using four performance methods: precision, recall, *F*-measure, and AUC were used as discriminators. Evaluation method demonstrates that result achieved with feature selections has better accuracy.

Four classifiers: random forest (RF), Naïve Bayes, RPart, and support vector machine (SVM) was used to predict software defect predictions. The aim of the study was proposed by [28] to specify the best classifier among these four classifiers. For this, eighteen data sets were used. Twelve of these data sets were obtained from NASA data set, three from open source data set, and three from commercial data set. Also, the confusion matrix was used for performance evaluation. The study concludes that each classifier has its own strength and weakness when used to classify some data, and the strength of the classifier depends on the data type.

A study proposed in 2018 introduce a model to predict the software defect based on feature selection by Dependent Naïve Bayes (FDNB) and Naïve Bayes neural network. The proposed system was trained and tested using eight NASA data set, each data set conduct of thirty-eight features some of these features believes that does not have a strong relation with prediction the software defect. Cross-validation method with convolution matrix was used to evaluate the performance of the proposed model. The system achieved better performance when it was trained with the selected features [29].

Software defect can be determinate by using some features that are associated with software. Some of these features has a strong relationship with class of software as (defect or nondefect). The authors of [30] introduce a method to investigate the relation between the defect or nondefect of the software and the features. The study conducted eleven data set with features varied between twenty-one features to thirty-nine features (every data set has different number of features). Therefore, the similarity measure (SM) method was used to select the strongest relation feature with the software defect. The strongest features were tested using the $k$-nearest neighbor (KNN) model, and the performance of KNN was evaluated using the AUC model. The results show that KNN achieved better results when it used the selected features over all features.

Since the software defect prediction has been the subject of interest for the last few decades, comparison between different software predictions models was introduced by [31]. The study used four classifiers used by previous works which are bagging, support vector machines (SVM), decision tree (DS), and random forest (RF) classifiers. In addition, each the performance of each classifier was evaluated using precision, recall, $F$-Score, TPR, and FPR. In this study, the feature selection was not considered. The result of this study shown that RF had the best performance for the most data set.

Conventional neural network was used to predict the number of software defects within two data sets obtained from the NASA data set for the neural network training stage. For testing, two different data sets were used. The proposed classifier performance was evaluated using the mean square error. Which has shown a high performance from this model.

In addition to the previous listed work, different studies have been done to improve the methodology of predicting the software defect such as [32] introduced a frame work for software prediction. The authors of [33] proposed a methodology to use a class imbalance in software learning detection. A study introduced by [34] investigated the performance of using cost-sensitive classification in predicting the software defect.

## 3. Materials and Methods

The four main steps in this study include collecting and processing the data set followed by feature selection and software defect as shown in Figure 1.

*3.1. Data Set.* NASA data set is one of the most common bench mark data sets widely used to for software defect prediction analysis [35–39]. In this research, fourteen NASA data sets was manipulated to train and test the proposed method. Data set obtained from different resources include PROMISE repository http://openscience.us, https://data.mendeley.com/datasets/923xvkk5mm/1, and http://promise.site.uottawa.ca/SERepository/datasets-page.html. Also, the number of attributes, the number of all samples, the number of defective samples, the number of nondefective samples, the number of missing attributes, and the defect rate for every data set conducted in this study are summarized in Table 1. Furthermore, the defect rate obtained by dividing the number of defected samples to the total number of the sample. For instance, the number of attributes for JM1 data set can be seen in column two (22), the number of all samples can be seen in column three (10885), the number of defective samples can be found in column four (2106), and number of nondefective samples found in column five, while the columns six and seven show the number of missing attributes (0) and defect rate, respectively (19.35).

*3.2. Data Set Preparation and Analysis.* Data sets in this study have different range of values for every attribute. Table 2 shows a random attribute values. For example, MC1 data set has values between 0.09 and 256. This variance of values can have many effects on the neural network performance. Therefore, all data set were normalized before it passes to the radial base function neural network, i.e., each attribute value for every data set was normalized between 0 and 1. Also, the true (defect software) and false (nondefect) values were converted to be one for the true value and zero for false value.

Neural network performance highly relies on the data set that used in the training stage, i.e., neural network trained with a data set that represent all class in will achieve a high performance [11]. Therefore, dividing data set to train and test neural network was subject of interest for many previous studies [40]. Moreover, there are no standard for dividing the data set for training and testing neural network porous. Some scholars used 80% of the data set for training and the rest of 20% for testing and validation purposes. Some other scholars used the well-known method $k$-fold cross validation ($k$ is the number of folds). The latter is based on dividing the data set randomly into specific number of folds ($k$). Each fold carries the same number of data set samples. Training and

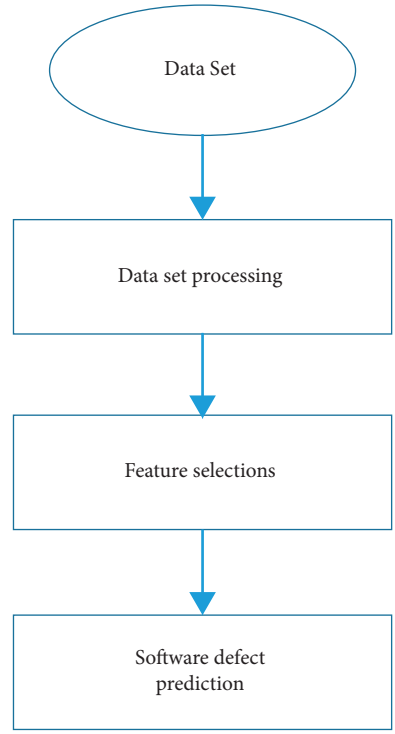


Figure 1: Work methodology flow chart.

Table 1: NASA data set.

| Data set | Attributes number | All samples number | Defective samples number | Nondefective samples number | Missing attributes number | Defect rate, % |
|---|---|---|---|---|---|---|
| CM1 | 22 | 498 | 49 | 449 | 0 | 9.83 |
| JM1 | 22 | 10885 | 2106 | 8779 | 0 | 19.35 |
| KC1 | 22 | 2109 | 326 | 1783 | 0 | 15.45 |
| KC2 | 22 | 522 | 105 | 415 | 0 | 20.50 |
| KC3 | 39 | 194 | 36 | 158 | 0 | 18.55 |
| KC4 | 39 | 125 | 44 | 81 | 0 | 35.20 |
| MC1 | 38 | 1988 | 46 | 1942 | 0 | 2.31 |
| MC2 | 39 | 125 | 44 | 81 | 0 | 35.2 |
| MW1 | 37 | 253 | 27 | 226 | 0 | 10.67 |
| PC1 | 22 | 1109 | 77 | 1032 | 0 | 6.94 |
| PC2 | 36 | 745 | 16 | 729 | 0 | 2.14 |
| PC3 | 37 | 1077 | 134 | 943 | 0 | 12.44 |
| PC4 | 37 | 1287 | 177 | 1110 | 0 | 13.75 |
| PC5 | 38 | 1711 | 471 | 1240 | 0 | 27.52 |

testing neural network using the $k$-fold cross-validation method is performed by holding one of the folds for testing and the rest of the folds for training ($k$–1). This procedure is repeated $k$ times. Moreover, each fold of the $k$-folds has the opportunity to pass the training for $k$–1 time and testing for one time [41]. In this study, the RBF neural network was trained and tested using (5 : 1) cross validation, where one of $k$-folds used for testing and four of the $k$-folds used for training.

*3.3. Background on Feature Selection.* The size of the data that used to train the neural network can have some effect on the performance of the neural network. Therefore, feature selection is widely used to choose the most important data to train the neural network. In this paper, the feature selection correlation base method was used to reduce the number of data by choosing the most important features, and this method was used before in [42, 43].

*3.4. Radial Base Function Neural Network.* For this study, the well-known radial base function neural network (RBF) was manipulated to predict the software defect. RBF neural network was being used in different applications such as medical application [44], image processing [45], and agricultures [46]. RB is a forward neural network consisting of three layers: input layer, hidden layer, and output layer as shown in Figure 2. Layers in the RBF neural network are fully connected to each other.

Table 2: Example of data set used in this study.

| Attribute number Data set | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CM1 | 1.1 | 1 | 1.4 | 5 | 24 | 19.5 | 1287 | 0.03 | 74 | 80 | 90 | F |
| JM1 | 695 | 0 | 0.09 | 1 | 2.3 | 4 | 19 | 7.6 | 3 | 33 | 11 | T |
| KC1 | 0 | 546 | 22 | 245 | 4 | 1 | 57 | 0.09 | 78 | 4 | 33 | F |
| KC2 | 4 | 2 | 0.7 | 33 | 673 | 157 | 3 | 49 | 3 | 0 | 406 | T |
| KC3 | 22 | 1 | 0 | 148 | 36 | 71 | 901 | 33 | 378 | 222 | 1 | T |
| KC4 | 0.01 | 1.01 | 1 | 2 | 55 | 19 | 35 | 28 | 75 | 20 | 33 | F |
| MC1 | 12 | 120 | 0 | 256 | 1 | 52 | 0.09 | 5 | 33 | 0 | 1.5 | T |
| MC2 | 2 | 1 | 999 | 535 | 43 | 58 | 96 | 40 | 1 | 0 | 4 | F |
| MW1 | 3 | 0 | 1 | 0.1 | 2 | 0.25 | 0.33 | 11 | 333 | 6 | 55 | F |
| PC1 | 1 | 80 | 0 | 0.4 | 7 | 1 | O.79 | 77 | 65 | 4 | 34 | T |
| PC2 | 2 | 80 | 95 | 0.8 | 36 | 0.70 | 50 | 21 | 60 | 5 | 123 | F |
| PC3 | 456 | 5 | 22 | 44 | 44 | 0.66 | 0.50 | 0.35 | 1 | 35 | 76 | T |
| PC4 | 180 | 32 | 55 | 44 | 25 | 0.15 | 33 | .09 | 0 | 80 | 5 | T |
| PC5 | 1.1 | 5.9 | 4 | 678 | 390 | 5.7 | 6 | 1 | 2.5 | 65 | 0 | F |

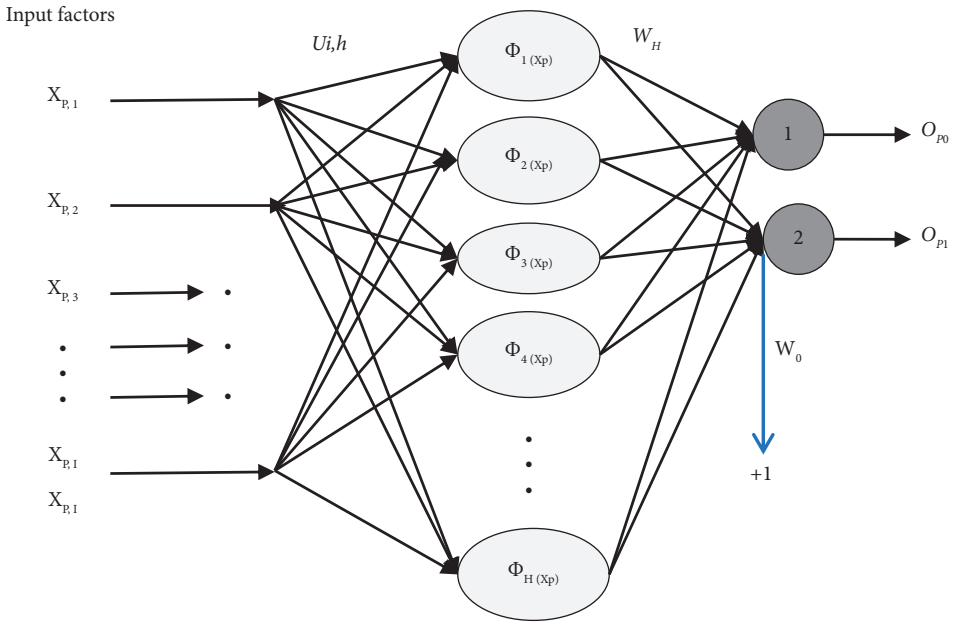


Figure 2: Radial base function neural network with single hidden and single output layer.

Figure 2 shows the RBF neural network, where $X_p$ is the input data set. Each input represents one feature in the data set. The weight between the $i$ th input and RBF hidden layer $H$ are scaled through the input weight $U_{i,h}$.

$$Y_{p,h,i} = X_{p,iU_{i,h}}. \tag{1}$$

Equation (1) shows the input vectors to the RBF hidden layer. Hence, $Y_{p,h}$ is the input to RBF layer (hidden layer), and $h$ is the index to the hidden layer. Equation (2) shows the computation in the hidden layer. $\varphi(\cdot)$ denotes RBF activation function. RBF used different activation function. Gaussian is the most common one.

$$\varphi\left(X_p\right) = \exp\left(-\frac{\left\|y_{p,h} - C_h\right\|2}{\sigma h}\right), \tag{2}$$

where $C_h$ is the center of RBF, $\sigma h$ is the width of RBF, and $\|\cdot\|$ is the Euclidean norm.

The RBF output, $O_{P,}$ can be calculated as

$$O_P = \sum_{h=1}^{H} \varphi_h\left(X_p\right) w_h + w_0. \tag{3}$$

Hence, the weight values the output layer received from the hidden layer is denoted by $w_h$. The bias weight is denoted by $w_0$.

*3.5. Evaluation Methods.* In previous research, many models was used to predict the software defect. Meanwhile, different performance indicators were used to evaluate the performance of the software defect models such as precision [3],

recall [10], and specificity [30]. Performance indicator used in this study include precision, recall, specificity (SPE), and accuracy (ACC).

The abbreviations used in this study are given as follows:

(i) TP is the true positive, which represent the true prediction of the positive values

(ii) FP is the false positive, which represent the false prediction of the positive values

(iii) TN is the true negative, which represent the true prediction of the negative values

(iv) FN is the false negative, which represent the false prediction of the negative values

Precision: it represents the correctness classification. It calculated by dividing the number of samples that correctly predicted defect divided by the total number of correctly predicted samples:

$$\text{precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})}. \quad (4)$$

Recall: it represents the rate of defecting models. It is calculated by dividing the number of samples that correctly predicted defect divided by the total number of modules that are actually defective.

$$\text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})}. \quad (5)$$

Accuracy: it is calculated by taking the ration between the total number of correctly prediction of defect and nondefect samples and total number of prediction samples.

$$\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{TN} + \text{FP} + \text{FN})}. \quad (6)$$

F-measure: it considers the importance of recall and precision are equally.

$$F - \text{measure} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}. \quad (7)$$

## 4. Experimental Setup

In this research, the radial base function neural network was used to predict the software defect in fourteen data sets. Therefore, the experiments were performed for every data set in two stages: first stage by training and testing RBF using all the features in the data set. Secondly, choosing the best feature selection in every data set using WEKA software and then training and testing RBF using that selected data set. Moreover, the *K*-cross-validation method was used to divided the data randomly before it is used to train the RBF. Meanwhile, precision and recall, SPE, and accuracy were used to evaluate the RBF achieved results. RBF performance was compared with results mentioned in the previous work. The RBF neural network was executed using MATLAB Toolboxes 2014 B version.

## 5. Results and Discussion

In this section, the research questions will be answered, and the results will be analysis.

Table 3 carries the answer for the first research question "how the RBF neural network performance, when it is used to predict the software defect for fourteen NASA data sets?" Table 3 summarizes the performance of the RBF neural network for prediction of the defect in fourteen data sets before applying the feature selection technique. It shows the RBF neural network performance results using the prediction evaluation indicators: accuracy, *F*-measure, precision, and recall in the training and testing stage.

*5.1. Training Stage.* RBF attained 100% prediction accuracy for MC1, PC1, and PC2 data sets. RBF achieved accuracy ranges between 93.20% and 94.27% for CM1, PC3, PC4 data sets. Meanwhile, prediction accuracy for JM1, KC1, KC3, KC4, and MW1varied between 80.09% and 88.67%. The lowest RBF prediction accuracy was 70.08%. For the *F*-measure, the highest *F*-measure 1 obtained for MC, PC1, and PC2. Data sets include CM1, MW1, PC3, and PC4. RBF achieved performance result between 0.93 and 0.98. For the data set, JM1 KC1, KC2, KC3, and PC5. RBF achieved performance between 0.82 and 0.89. The lowest *F*-measure achieved by RBF was for MC2 0.77. The highest RBF precision achieved for MC1 99.91%, while the lowest was 72.63% for MC2. RBF precision for data set include CM1, MC1 PC1, PC2, PC3, and PC4 and varied between 94.19% and 98.99%, while the RBF performance varied between 80.88% and 89.91% for JM1, KC1, KC2, KC3, KC4, and PC5. Recall is the third performance indicator used in this study. As in Table 3, 100% is the highest recall achieved by RBF for MC1, and 76.82% is the lowest recall achieved using MC2. Furthermore, RBF predicted the software defect on CM1, MW1, PC1, PC2, PC3, and PC4 with recall varied between 90.09% and 99.98%. The rest of data set prediction performance ranged between 80.82%–89.91%.

*5.2. Testing Stage.* As the well-known neural network achieved better performance in the training stage over testing stage, the RBF neural network achieved the highest and lowest accuracy 98.42% and 67.29% for PC2 and MC2 data sets, respectively. Besides PC2, RBF obtained accuracy over 90% for five data set, and they are CM1, MC1, PC1, PC3, and PC4. The prediction accuracy for JM1, KC1, KC4, and MW1 is 80.05% and 86.89%, while the RBF prediction performance for the rest of the data set is lower than 78.25%. The best F-measure performance was 0.99 for PC1, MC1, and PC2. CM1, MW1, and PC3 was predicted with performance over 0.90. JM1, KC1, and KC3 were predated with performance over 0.80, while the lowest *F*-measure was obtained for KC2 and MC2. For recall performance indicator, MC1 with 99.41% and MC2 77.13% represent the highest and lowest *F*-measure, respectively. Eight data sets were predicted with performance over 90.0%: CM1, KC1, MC1, MW1, PC1, PC2, PC3, and PC4. The rest of the data set was predicted with performance over 80.0%. The last

Table 3: RBF training and testing results using all features in the data set.

| Data set | Feature number | Training results | | | | Testing results | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Precision | Recall | *F*-measure | Accuracy | Precision | Recall | *F*-measure | Accuracy |
| CM1 | 22 | 94.47 | 95.86 | 0.95 | 93.20 | 92.18 | 95.53 | 0.93 | 90.89 |
| JM1 | 22 | 85.34 | 86.82 | 0.89 | 84.38 | 81.99 | 83.98 | 0.85 | 81.18 |
| KC1 | 22 | 83.02 | 83.87 | 0.86 | 82.99 | 84.78 | 90.10 | 0.84 | 80.05 |
| KC2 | 22 | 81.83 | 81.08 | 0.84 | 79.59 | 79.82 | 81.01 | 0.79 | 76.49 |
| KC3 | 39 | 80.99 | 81.97 | 0.85 | 80.09 | 81.47 | 80.76 | 0.81 | 78.25 |
| KC4 | 39 | 84.45 | 85.72 | 0.84 | 83.89 | 81.08 | 82.09 | 0.83 | 80.08 |
| MC1 | 38 | 99.91 | 100 | 1.00 | 100 | 99.01 | 99.41 | 0.99 | 98.89 |
| MC2 | 39 | 72.63 | 76.82 | 0.77 | 70.08 | 70.59 | 77.13 | 0.73 | 67.29 |
| MW1 | 37 | 89.91 | 90.09 | 0.93 | 88.67 | 88.82 | 90.92 | 0.91 | 86.89 |
| PC1 | 22 | 98.99 | 99.98 | 1.00 | 100 | 97.99 | 98.91 | 0.99 | 97.78 |
| PC2 | 36 | 98.09 | 98.99 | 1.00 | 100 | 99.01 | 99.32 | 0.99 | 98.42 |
| PC3 | 37 | 94.42 | 97.73 | 0.98 | 93.27 | 92.23 | 93.12 | 0.95 | 90.07 |
| PC4 | 37 | 94.19 | 96.76 | 0.97 | 94.20 | 91.89 | 93.78 | 0.93 | 91.40 |
| PC5 | 38 | 80.82 | 83.79 | 0.82 | 79.84 | 78.01 | 81.40 | 0.80 | 77.32 |

software prediction indicator used in the testing stage was precision. The highest precision obtained was 99.41% for MC1 data set, while the lowest precision obtained was 70.59% for MC2 data set. Six data sets predict with precision more than 90.0%: CM1, MC1, PC1, PC2, PC3, and PC4. In addition, five data sets predict with precision more than 80.0%: JM1, KC1, KC3, KC4, and MW1. KC2 and PC5 achieved the 79.82% for the former and 78.01% for the latter.

This section will answer for the second research question in this study "How the RBF neural network performance, when it is used to predict the software defect for fourteen NASA data sets after eliminating not related feature? Table 4 shows the number of selected features by using the correlation base method. In this study, only the top 50% of the feature selected by correlation based method was chosen.

Table 5 shows the RBF neural network performance after applying the correlation-based method as a feature selection method. The following section will discuss the RBF software prediction performance in both training and testing stages.

*5.3. Training Stage.* During the training stage, the RBF neural network attained 100% accuracy for predicting the software defect for MC1, PC1, and PC2 data sets. In addition, the RBF accuracy range between 90.71% and 96.49% for the data sets CM1, MW1, PC3, and PC4. The rest of the data set obtained accuracy range between 81.89% and 87.83% except for MC2 71.72%. The 1.00 *F*-measure was obtained for MC1, PC1, and PC2, while CM1, MW1, and PC3 obtained 0.99. While the *F*-measure performance for JM1 and PC4 was 0.90 and PC4, respectively. 0.79 is the lowest *F*-measure for software defect prediction obtained for MC2. The rest of the data set *F*-measure ranged between 0.84 and 0.89. The highest recall achieved by the RBF neural network was 100% achieved for three data sets MC1, PC1, and PC2. Data sets include CM1, JM1, MW1, PC3, and PC4 which obtained accuracy over 90.00%. Meanwhile, RBF predicts the software defect for KC1, KC2, KC3, KC4, and PC5 with recall performance range between 84.67% and 86.89%. For the precision as an evaluation method used in this study and as Table 5 shows, the RBF neural network attained the highest software defect prediction for PC1 and PC2, directly followed by 99.79% for MC1, 97.96% for PC4, 97.10% for CM1, 95.39% and 91.28% for PC3 and MW1 data set, respectively. The rest of the data ranged between 83.67% and 88.87%. The lowest precision was 72.27% for MC2.

*5.4. Testing Stage.* The testing result obtained by the RBF neural network using the data set after feature selection is listed under testing result parts in Table 5. The highest software defect prediction performance was 99.80% for PC2 data set, followed with slight difference 99.01% for MC1 data set. Data sets like PC1 achieved 98.99%, PC4 94.44%, PC3 94.11%, and CM1 93.99%. 70.18% was the lowest software defect prediction accuracy achieved for MC2 data set. The rest of the data set obtained accuracy between 79.02% and 88.90%. The highest *F*-measure obtained by the RBF neural network was 0.99 obtained for three data sets MC1, PC1, and PC, while the lowest *F*-measure was 0.76 obtained for MC2 data set. CM1, MW1, and PC3 achieved results between 0.95 and 0.97. The rest of the data set *F*-measure results were below 0.88. For recall, RBF predicts the software defect with 100% for MC1 and PC1, 99.98% and 96.23% for PC2 and PC3, respectively, 95.17% for PC4, and 92.09% for MW1. The recall performance for the rest of the data set was below 89.00%. Precession performance obtained by the RBF neural network for testing data set varied form the 100% obtained for MC1 and PC2 and 73.27% as lowest results for MC2. Precision results with performance over 90.00% obtained for data sets include CM1, MC2, MW1, PC2, PC3, and PC4. Meanwhile, the precision for two data sets—KC3 and MC2—was 79.30% and 73.27%, respectively. The rest of the data set obtained results between 80.89% and 85.29%. Finally, it is worth to mention that this study is the only study mention to the results values of the precision and recall.

*5.5. Comparing RBF Performance with Previous Methods.* The third research question in this study is as follows: how does the RBF neural network perform when it is compared to previous methods that are used to predict software defect for the same data set?

Table 4: Number of selected feature using software.

| Data set | CM1 | JM1 | KC1 | KC2 | KC3 | KC4 | MC1 | MC2 | MW1 | PC1 | PC2 | PC3 | PC4 | PC5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Feature number | 7 | 8 | 8 | 3 | 4 | 5 | 10 | 8 | 11 | 6 | 9 | 6 | 8 | 5 |

Table 5: RBF training and testing results using selected features in the data set.

| Data set | Selected feature number | Training results | | | | Testing results | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Precision | Recall | *F*-measure | Accuracy | Precision | Recall | *F*-measure | Accuracy |
| CM1 | 7 | 97.10 | 97.77 | 0.99 | 95.30 | 94.01 | 95.78 | 0.95 | 93.99 |
| JM1 | 8 | 88.87 | 90.09 | 0.90 | 87.83 | 85.18 | 87.89 | 0.87 | 84.87 |
| KC1 | 8 | 85.78 | 86.89 | 0.86 | 84.19 | 83.24 | 86.24 | 0.83 | 83.25 |
| KC2 | 3 | 83.19 | 84.67 | 0.84 | 82.88 | 81.27 | 83.82 | 0.82 | 79.11 |
| KC3 | 4 | 84.99 | 85.39 | 0.86 | 84.19 | 79.30 | 82.10 | 0.85 | 78.25 |
| KC4 | 5 | 84.78 | 85.99 | 0.89 | 83.89 | 85.29 | 86.28 | 0.86 | 83.18 |
| MC1 | 10 | 99.79 | 100 | 1.00 | 100 | 99.89 | 100 | 0.99 | 99.01 |
| MC2 | 8 | 72.27 | 73.75 | 0.79 | 71.72 | 73.27 | 76.67 | 0.76 | 70.18 |
| MW1 | 11 | 91.28 | 94.16 | 0.99 | 90.71 | 90.90 | 92.09 | 0.95 | 88.90 |
| PC1 | 6 | 100 | 100 | 1.00 | 100 | 98.79 | 99.98 | 0.99 | 98.99 |
| PC2 | 9 | 100 | 100 | 1.00 | 100 | 100 | 100 | 0.99 | 99.80 |
| PC3 | 6 | 95.39 | 96.37 | 0.99 | 95.38 | 95.67 | 96.23 | 0.97 | 94.11 |
| PC4 | 8 | 97.96 | 98.94 | 0.98 | 96.49 | 95.12 | 95.17 | 0.95 | 94.44 |
| PC5 | 5 | 83.67 | 85.96 | 0.84 | 81.89 | 80.89 | 81.80 | 0.80 | 79.02 |

This section compares the software defect prediction results obtained by the RBF neural network for the fourteen data sets before feature selection, with methods mentioned in the previous studies in the first part of this section. It is worth to mention that the reason behind eliminating the two evaluation methods—precision and recall—is there are not many studies mentioned to the results obtained by using the precision and recall. Furthermore, every data set results will be compared separately with the other previous methods results.

Table 6 represents the result of CM1 data set. Eight classification methods are mentioned. It can be seen that the highest *F*-measure 0.96 obtained using RBF used in this study followed by RBF 0.95 and SVM 0.95 from the study [47]. Since the RBF neural network result is a black box, the reason behind the difference between the RBF result in this study and [34] cannot be explained. The lowest result obtained by SVM 0.78 in the study is proposed in [18]. Meanwhile, the best accuracy (93.99%) obtained by this study followed by SVM with 91.97% and RBF 90.33%. Meanwhile, the lowest accuracy was 75.00% obtained by SVM.

Table 7 reflected the results of JM1 data set. The F-measure results obtained from RBF in this study was 0.87, while the highest F-measure obtained was 0.90 by MLP, SVM and RBF. Naive Bayes, RF, DS, and SVM obtained 0.89, 0.76, 0.71, and 0.71, respectively. 89.97% is the highest accuracy achieved by MLP, followed with 84.87% achieved by RBF proposed in this study. The lowest accuracy was 69.00% obtained by SVM proposed in [18]. SVM [34], RBF [34], and Naive Bayes achieved 81.43%, 81.73%, and RBF. The rest of the methods achieved results lower than 80.00%.

Performance of the RBF neural network for KC1 used is presented in Table 8. The highest F-measure was 0.92 obtained by MLP, SVM, and RBF [34]. Naive Bayes obtained 0.90 followed by 0.83 obtained by RBF proposed in this study. The highest accuracy obtained using MLP was 85.51%, while the accuracy 83.25% was obtained using the RBF neural network proposed by this study. MLP [3] obtained the lowest accuracy 79.46%. Naive Bayes, SVM, and RBF [34] obtained accuracy 82.10%, 84.47%, and 84.99%, respectively.

Table 6: Comparison of RBF results with other previous methods before feature selection for CM1 data set.

| Source | Algorithm | *F*-measure | Accuracy |
|---|---|---|---|
| [31] | RF | 0.83 | 83.00% |
| | DS | 0.79 | 78.00% |
| | SVM | 0.78 | 75.00% |
| [47] | Naive Bayes | 0.91 | 83.45% |
| | MLP | 0.94 | 89.55% |
| | SVM | 0.95 | 91.97% |
| | RBF | 0.95 | 90.33% |
| [42] | J48 | N/A | 88.60% |
| This research | RBF | 0.96 | 93.99 |

Table 7: Comparison of RBF results with other previous methods before feature selection for JM1 data set.

| Source | Algorithm | *F*-measure | Accuracy |
|---|---|---|---|
| [31] | RF | 0.76 | 77.00% |
| | DS | 0.71 | 71.00% |
| | SVM | 0.71 | 69.00% |
| [47] | Naive Bayes | 0.89 | 81.43% |
| | MLP | 0.90 | 89.97% |
| | SVM | 0.90 | 81.73% |
| | RBF | 0.90 | 81.61% |
| [42] | J48 | N/A | 79.81% |
| This research | RBF | 0.87 | 84.87 |

Table 8: Comparison of RBF results with other previous methods before feature selection for KC1 data set.

| Source | Algorithm | *F*-measure | Accuracy |
|---|---|---|---|
| [3] | MLP | N/A | 79.46% |
| [47] | Naive Bayes | 0.90 | 82.10% |
| | MLP | 0.92 | 85.51% |
| | SVM | 0.92 | 84.47% |
| | RBF | 0.92 | 84.99% |
| [42] | J48 | N/A | 84.63% |
| [48] | | | |
| This research | RBF | 0.83 | 83.25 |

KC2 *F*-measure results are shown in Table 9. Seven previous method results were used to compare with RBF obtained result in this study. Naive Bayes [34], MLP [34], SVM [34], and RBF [34] obtained the same *F*-measure 0.90. Also, RF, DS, and SVM obtained results 0.82, 0.78, and 0.80, respectively. Accuracy is the second evaluation method which was used to compare the RBF neural network used in this study. 84.78% was the highest accuracy achieved by Naive Bayes, followed by 83.64 and 83.63% accuracy obtained by MLP [34] and RBF [34]. KC2 software defect predicted accuracy was 82.34% and 82.00% for SVM [34] and RF [18], respectively. Also, MLP [3] predicts the software defect for KC2 data set 79.64%.

Table 10 reflects the result of KC3 data set. It can be seen that the best *F*-measure 0.95 was obtained using SVM [34] followed by MLP 0.94, Naive Bayes 0.91, and RBF [34] 0.90. Meanwhile, the lowest *F*-measure 0.71 obtained by RF. DS, SVM [18], and LR obtained 0.71, 0.78, and 0.78, respectively. The RBF used in this research obtained 0.85. Table 10 shows also the accuracy results where 90.04% and 90.02% was the best accuracy obtained by MLP and SVM [34], respectively. 89.78%, 86.17%, and 82.80% are the accuracy obtained by RBF [34], Naive Bayes, and RF. SVM [18], DS, and RBF in this research obtained 79.01%, 78.01%, and 78.25%, respectively. Also, LR obtained 77.01%, while the J star method obtained the lowest accuracy among the methods mentioned in Table 10.

The results of KC4 data set were compared with three previous methods SVM, RPart, and RF. The best *F*-measure 0.82 obtained by RPart followed by RF 0.80 and 0.79 for SVM. RBF proposed in this study achieves the best *F*-measure 0.86. It can be noticed from Table 11 that the accuracy did not mention in the previous method, while RBF obtained 83.18%.

Table 12 summarizes the result of previous work and RBF for MC1 data set. The highest and lowest *F*-measure were 0.99 for this research RBF and 0.10 for MLP, SVM [34], and RBF [34]. Naive Bayes and RF obtained the same *F*-measure 0.97. DS 0.95, SVM and LR [18] obtained 0.88. On the other hand, J star obtained 100% accuracy, while MLP and RBF [34] obtained the second and third accuracy with 99.40% and 99.26%, respectively. SVM [34] obtained 99.26%, RF 97%, and DS 94%, while SVM [18] and LR obtained the lowest results 81%. J48 obtained 88.60%.

Table 9: Comparison of RBF results with other previous methods before feature selection for KC2 data set.

| Source | Algorithm | *F*-measure | Accuracy |
|---|---|---|---|
| [3] | MLP | N/A | 79.64% |
| [47] | Naive Bayes | 0.90 | 84.78% |
| | MLP | 0.90 | 83.64% |
| | SVM | 0.90 | 82.34% |
| | RBF | 0.90 | 83.63% |
| [42] | J48 | N/A | 81.36% |
| [31] | RF | 0.82 | 82.00% |
| | DS | 0.78 | 78.01% |
| | SVM | 0.80 | 79.10% |
| This research | RBF | 0.82 | 79.11 |

Table 10: Comparison of RBF results with other previous methods before feature selection for KC3 data set.

| Source | Algorithm | *F*-measure | Accuracy |
|---|---|---|---|
| [43] | J star | N/A | 72.06% |
| [47] | Naive Bayes | 0.91 | 86.17% |
| | MLP | 0.94 | 90.04% |
| | SVM | 0.95 | 90.02% |
| | RBF | 0.90 | 89.78% |
| [31] | RF | 0.71 | 82.80% |
| | DS | 0.78 | 78.01% |
| | SVM | 0.78 | 79.01% |
| | LR | 0.79 | 77.01% |
| This research | RBF | 0.85 | 78.25 |

Table 11: Comparison of RBF results with other previous methods before feature selection for KC4 data set.

| Source | Algorithm | *F*-measure | Accuracy |
|---|---|---|---|
| [3] | SVM | 0.79 | N/A |
| | RPart | 0.82 | N/A |
| | RF | 0.80 | N/A |
| This research | RBF | 0.86 | 83.18 |

Table 12: Comparison of RBF results with other previous methods before feature selection for MC1 data set.

| Source | Algorithm | *F*-measure | Accuracy |
|---|---|---|---|
| [43] | J star | N/A | 100% |
| [47] | Naive Bayes | 0.97 | 94.57% |
| | MLP | 0.10 | 99.40% |
| | SVM | 0.10 | 99.26% |
| | RBF | 0.10 | 99.27 |
| [31] | RF | 0.97 | 97.00% |
| | DS | 0.95 | 94.00% |
| | SVM | 0.88 | 81.00% |
| | LR | 0.88 | 81.00% |
| [42] | J48 | N/A | 88.60% |
| This research | RBF | 0.99 | 99.01 |

Seven classification methods are listed in Table 13. 0.82 was the best *F*-measure obtained Naive Bayes and SVM [34] followed by 0.81 for RBF [34]. MLP and RBF (this research) obtained 0.78 and 0.76, respectively. Meanwhile, RBF [36]

Table 13: Comparison of RBF results with other previous methods before feature selection for MC2 data set.

| Source | Algorithm | *F*-measure | Accuracy |
|---|---|---|---|
| [47] | Naive bayes | 0.82 | 72.52% |
| | MLP | 0.78 | 67.97% |
| | SVM | 0.82 | 72.00% |
| | RBF | 0.81 | 69.51% |
| [31] | RF | 0.67 | 69.00% |
| | DS | 0.66 | 67.00% |
| | SVM | 0.64 | 65.00% |
| | LR | 0.63 | 63.00% |
| [49] | RBF | 0.48 | 64.68% |
| This research | RBF | 0.76 | 70.18 |

obtained the lowest *F*-measure 0.48. RF 0.67, DS 0.66, SVM [18] 0.64, and LR 0.63. Accuracy is the second performance evaluation method mentioned in Table 13. It can be noticed that 72.52 is the best accuracy obtained by Naive Bayes, 72.00% for SVM [34], 70.18% for RBF (this research), and 69.51% and 69% for RBF [34] and RF, respectively. DS 67.00%, SVM [18] 65.00%, and RBF [36] 64.68%, and LR obtained the lowest accuracy, 63.00%.

Results of MW1 data set are reflected in Table 14. 0.96 is the best *F*-measure obtained by two algorithms SVM [34] and RBF [34]; also, MLP [34] and RBF (this research) obtained 0.95 Naive Bayes 0.90. KNN and DT obtained the lowest *F*-measure 0.44 and 0.16, respectively. The best accuracy obtained to predict the software defect for MW1 data set is 92.19% for SVM [34] followed by 91.99 and 91.09% for RBF while and MLP. RBF [36] respectively. 89.33% and 88.90% for this research respectively . DT and KNN obtained the same accuracy, 86.66%, and 83.63% is the lowest accuracy obtained by Naive Bayes.

The PC1 data set results are mentioned in Table 15. RBF (this research) obtained the best F-measure 0.99, while SVM [34] obtained the lowest result 0.07. The two algorithms—MLP and Naive Bayes—obtained 0.11 for both—RF, 0.91; DS, 0.88; LR, 0.85; DT, 0.50; KNN, 0.28; and RBF [36], 0.15. RBF (this research) obtained the best accuracy—98.99% and 30.09%. The lowest accuracy obtained by MLP.94.60% for KNN, 93.13% RBF [34] and DT. 93.09% SVM [34] 91.00% RF, 88.07% Naive Bayes, 87.61 J star, DS 87.00%. 81.00% for LR, and 79.00% SVM [18].

Table 16 shows the results of PC2 data set. It can be noticed that *F*-measure obtained by MLP, SVM, and RBF [34] is 1.00 and 0.99 for Naive Bayes and RBF (this research). Also, the highest software defect prediction accuracy obtained is 99.80% RBF (this research), 99.59% SVM, 99.58% RBF [34], and 99.52% MLP. In addition, DT 97.69%, RBF [36] 97.65%, Naive Bayes 96.96%, and KNN 96.77%.

Table 17 reflects PC3 data set results. RBF (this research) obtained the highest F-measure 0.97, while KNN and DT obtained the lowest 0.35. RBF [34], SVM [34] 0.95, MLP 0.94, RF 0.84, DS 0.81, LR 0.79, SVM [18] 0.78 and Naive Bayes [34]. RBF (this research) obtained 94.11% which represents the best accuracy followed by RBF 89.76% and SVM [34] 89.33%. In addition, MLP 87.55%, RBF and DT [36] 86.39% for both. KNN 86.07%, RF [18] 84.00%, and SVM [18] 74.00%, and the lowest accuracy obtained 46.87% obtained by Naive Bayes.

Table 14: Comparison of RBF results with other previous methods before feature selection for MW1 data set.

| Source | Algorithm | *F*-measure | Accuracy |
|---|---|---|---|
| [47] | Naive Bayes | 0.90 | 83.63% |
| | MLP | 0.95 | 91.09% |
| | SVM | 0.96 | 92.19% |
| | RBF | 0.96 | 91.99% |
| [49] | RBF | N/A | 89.33% |
| | KNN | 0.44 | 86.66% |
| | DT | 0.16 | 86.66% |
| This research | RBF | 0.95 | 88.90 |

Table 15: Comparison of RBF results with other previous methods before feature selection for PC1 data set.

| Source | Algorithm | *F*-measure | Accuracy |
|---|---|---|---|
| [43] | J star | N/A | 87.61% |
| [47] | Naive bayes | 0.11 | 88.07% |
| | MLP | 0.11 | 30.09% |
| | SVM | 0.07 | 93.09% |
| | RBF | 0.12 | 93.13% |
| [31] | RF | 0.91 | 91.00% |
| | DS | 0.88 | 87.00% |
| | SVM | 0.83 | 79.00% |
| | LR | 0.85 | 81.00% |
| [49] | RBF | 0.154 | 94.60% |
| | KNN | 0.286 | 92.64% |
| | DT | 0.500 | 93.13% |
| This research | RBF | 0.99 | 98.99 |

Table 16: Comparison of RBF results with other previous methods before feature selection for PC2 data set.

| Source | Algorithm | *F*-measure | Accuracy |
|---|---|---|---|
| [47] | Naive Bayes | 0.99 | 96.96% |
| | MLP | 1.00 | 99.52% |
| | SVM | 1.00 | 99.59% |
| | RBF | 1.00 | 99.58% |
| [49] | RBF | N/A | 97.65% |
| | KNN | N/A | 96.77% |
| | DT | N/A | 97.69% |
| This research | RBF | 0.99 | 99.80 |

The results of PC4 are summarized in Table 18. The highest F-measure obtained by RF (this research) is 0.95, and 0.25, 0.28, and 0.58 are the lowest for RBF, KNN, and DT, respectively. MLP and SVM [34] obtained 0.94. RBF [34] obtained 0.93. Naive Bayes and RF [18] obtained 0.92 and 0.90. Also DS 0.86 0.86, 0.84 for SVM [18] and LR. Moreover, RF (this research) obtained the highest accuracy (94.44%), while RBF [34] obtained the lowest accuracy (27.27%). RF is 90.00% followed by 89.115 for MLP, 88.45% for SVM, 87.40% for RBF [36], 86.87% for DT, 85.82% for KNN, 85.51% for Naive Bayes, 85.00% for DS, 82.00% for LR, and 81.00% for SVM.

Table 17: Comparison of RBF results with other previous methods before feature selection for PC3 data set.

| Source | Algorithm | *F*-measure | Accuracy |
|---|---|---|---|
| [47] | Naive Bayes | 0.60 | 46.87% |
| | MLP | 0.94 | 87.55 |
| | SVM | 0.95 | 89.33% |
| | RBF | 0.95 | 89.76% |
| [31] | RF | 0.84 | 84.00% |
| | DS | 0.81 | 80.00% |
| | SVM | 0.78 | 74.00% |
| | LR | 0.79 | 84.00% |
| [49] | RBF | N/A | 86.39% |
| | KNN | 0.35 | 86.07% |
| | DT | 0.35 | 86.39% |
| This research | RBF | 0.97 | 94.11% |

Table 18: Comparison of RBF results with other previous methods before feature selection for PC4 data set.

| Source | Algorithm | *F*-measure | Accuracy (%) |
|---|---|---|---|
| [47] | Naive Bayes | 0.92 | 85.51 |
| | MLP | 0.94 | 89.11 |
| | SVM | 0.94 | 88.45 |
| | RBF | 0.93 | 27.27 |
| [31] | RF | 0.90 | 90.00 |
| | DS | 0.86 | 85.00 |
| | SVM | 0.84 | 81.00 |
| | LR | 0.84 | 82.00 |
| [49] | RBF | 0.25 | 87.40 |
| | KNN | 0.28 | 85.82 |
| | DT | 0.58 | 86.87 |
| This research | RBF | 0.95 | 94.44 |

Table 19, shows the result of PC5 data set, where the best *F*-measure 0.99 is obtained by MLP, SVM [34], and RBF [34]. Naive Bayes obtained 0.98, and RBF (this research) obtained 0.80 followed by RF, DS, and SVM [18] (0.76, 0.72, and 0.69, respectively). LR obtained 0.70. Meanwhile, DT, KNN, and RBF [36] obtained the lowest results with 0.53, 0.49, and 0.23, respectively. The accuracy obtained by SVM, MLP, and Naive Bayes was 97.23%, 97.03%, and 96.03%, respectively. 87.27% RBF [34], 79.02% RBF (this research), 76.00% RF, 75.79% RBF [36] and 75.00% DT, KNN 73.03%, 71.01% DS. 68.01% SVM and 68.00% for LR [18], respectively.

The fourth research question in this study is as follows: "how does the RBF neural network perform when it is compared to previous methods that are used to predict the software defect for fourteen NASA data sets after eliminating the nonrelated feature?" Table 20 reflects the answer for question four. As seen in Table 20, two things can be noticed that this study is the first study mentioned for software defect prediction for KC4 data set. In order to compare the result obtained in this study with results obtained in the previous work, two parameters were selected—*F*-measure and accuracy. Nevertheless, few studies mentioned the F-measure results, precisely if feature selection techniques performed before predecting the software defect.

Table 19: Comparison of RBF results with other previous methods before feature selection for PC5 data set.

| Source | Algorithm | *F*-measure | Accuracy |
|---|---|---|---|
| [47] | Naive Bayes | 0.98 | 96.03% |
| | MLP | 0.99 | 97.03% |
| | SVM | 0.99 | 97.23% |
| | RBF | 0.99 | 87.27% |
| [31] | RF | 0.76 | 76.00% |
| | DS | 0.72 | 71.01% |
| | SVM | 0.69 | 68.01% |
| | LR | 0.70 | 68.00% |
| [49] | RBF | 0.23 | 75.79% |
| | KNN | 0.49 | 73.03% |
| | DT | 0.53 | 75.00% |
| This research | RBF | 0.80 | 79.02 |

CM1 results in Table 20 show the best *F*-measure obtained by RBF (this research), and SVM is 0.95, followed by DT 0.94 and MLP 0.94. While 93.99% is the best accuracy obtained by RBF (this research) followed by 90.69% obtained by SVM, MLP [36], Boost-RF, Bag-RF-FS, DT, and MLP [38] which obtained accuracy between 89.32% and 89.97%. In addition, RF obtained 84.60% and J star obtained the lowest accuracy 81.69% among all listed classification methods.

For the JM1 data set, RF (this research) obtained the highest F-measure (0.87), while MLP obtained the lowest (0.17). RF (this research) obtained the highest accuracy (84.87%) followed by J star (81.90%), MLP, Boost-RF, and Bag-RF-FS which obtained 80.44%, 80.56%, and 80.61%, respectively. RF obtained the lowest accuracy (73.90%).

Nine prediction methods used to predict the software defect in KC1 data set results are listed in Table 20. The MLP [38], SVM [38], and DT obtained the best *F*-measure with 0.92, 0.91, and 0.91, respectively. RBF (this research) obtained 0.83, and MLP [36] obtained the lowest *F*-measure 0.43. J star and DT obtained the best accuracy of 85.97% and 85.68%, respectively. The lowest accuracy was 71.08% obtained by J48. Algorithms like RF [29] obtained 71.08%, Boost-RF and Bag-RF-FS obtained 78.51%, MLP [36] obtained 77.65%, DT obtained 84.56%, RF obtained 84.16%, RBF (this research) obtained 83.25%, SVM [38] obtained 84.59%, and MLP [38] obtained 85.68%.

Only one previous study was available to compare this study result with it for KC2 data set. 0.82 RBF (this research) was obtained for *F*-measure, while the LSTSVM did not mention this value. RBF (this research) obtained 79.11%, while LSTSVM obtained a better accuracy with 86.12%.

For KC3 data set, the highest *F*-measure was 0.96 attained by using three algorithms MLP [36], DT, and SVM [38], while the lowest was obtained using MLP 0.28. RBF (this research) obtained 0.85. The highest and lowest accuracy was 93.50% and 77.58% attained using MLP [38] and Bag-RF-FS, respectively. SVM [38] and DT obtained 93.28% and 93.11%, respectively while MLP 82.75%, Jstar and RF 80.50% for both, 79.13% for Boost-RF. The RBF (this research) obtained 78.25%.

Table 20: Comparison of RBF results with other previous methods after feature selection.

| Data set | Source | Algorithm | *F*-measure | Accuracy |
|---|---|---|---|---|
| CM1 | [43] | J star | N/A | 81.69% |
| | | RF | N/A | 84.60% |
| | [49] | MLP | N/A | 89.79% |
| | [50] | Boost-RF | N/A | 89.97% |
| | | Bag-RF-FS | N/A | 89.97% |
| | [51] | DT | 0.94 | 89.92% |
| | | MLP | 0.94 | 89.32% |
| | | SVM | 0.95 | 90.69% |
| | This research | RBF | 0.95 | 93.99 |
| JM1 | [43] | J star | N/A | 81.90% |
| | | RF | N/A | 73.90% |
| | [49] | MLP | 0.17 | 80.44% |
| | [50] | Boost-RF | N/A | 80.56% |
| | | Bag-RF-FS | N/A | 80.61% |
| | This research | RBF | 0.87 | 84.87 |
| KC1 | [42] | J48 | N/A | 68.78% |
| | | RF | N/A | 71.08 |
| | [43] | RF | N/A | 84.16% |
| | | J star | N/A | 85.97% |
| | [50] | Boost-RF | N/A | 78.51% |
| | | Bag-RF-FS | N/A | 78.51% |
| | [49] | MLP | 0.43 | 77.65% |
| | [51] | DT | 0.91 | 84.56% |
| | | MLP | 0.92 | 85.68% |
| | | SVM | 0.91 | 84.59% |
| | This research | RBF | 0.83 | 83.25% |
| KC2 | [52] | LSTSVM | N/A | 86.12% |
| | This research | RBF | 0.82 | 79.11% |
| KC3 | [43] | J star | N/A | 80.50% |
| | | RF | N/A | 80.50% |
| | [50] | Boost-RF | N/A | 79.13% |
| | | Bag-RF-FS | N/A | 77.58% |
| | [49] | MLP | 0.28 | 82.75% |
| | [51] | DT | 0.96 | 93.11% |
| | | MLP | 0.96 | 93.50% |
| | | SVM | 0.96 | 93.28% |
| | This research | RBF | 0.85 | 78.25% |
| KC4 | This research | RBF | 0.86 | 83.18% |
| MC1 | [50] | Boost-RF | N/A | 97.61% |
| | | Bag-RF-FS | N/A | 97.61% |
| | [49] | MLP | N/A | 97.61% |
| | This research | RBF | 0.99 | 99.01% |
| MC2 | [50] | Boost-RF | N/A | 64.86% |
| | | Bag-RF-FS | N/A | 62.16% |
| | [49] | MLP | N/A | 97.60% |
| | This research | RBF | 0.76 | 70.18 |
| MW1 | [50] | Boost-RF | N/A | 89.33% |
| | | Bag-RF-FS | N/A | 89.33% |
| | [49] | MLP | 0.40 | 92.00% |
| | This research | RBF | 0.95 | 88.90% |

Table 20: Continued.

| Data set | Source | Algorithm | *F*-measure | Accuracy |
|---|---|---|---|---|
| PC1 | [43] | J star | N/A | 91.17% |
| | | RF | N/A | 91.17% |
| | [50] | Boost-RF | N/A | 96.07% |
| | | Bag-RF-FS | N/A | 96.07% |
| | [49] | MLP | 0.42 | 96.56% |
| | [51] | DT | 0.96 | 93.58% |
| | | MLP | 0.96 | 93.59% |
| | | SVM | 0.96 | 93.10% |
| | This research | RBF | 0.99 | 98.99% |
| PC2 | [50] | Boost-RF | N/A | 97.23% |
| | | Bag-RF-FS | N/A | 97.69% |
| | [49] | MLP | N/A | 97.69% |
| | This research | RBF | 0.99 | 99.80% |
| PC3 | [50] | Boost-RF | N/A | 87.34% |
| | | Bag-RF-FS | N/A | 87.34% |
| | [49] | MLP | 0.14 | 85.12% |
| | This research | RBF | 0.97 | 94.11 |
| PC4 | [50] | Boost-RF | N/A | 91.60% |
| | | Bag-RF-FS | N/A | 90.81% |
| | [49] | MLP | 0.44 | 88.97% |
| | This research | RBF | 0.95 | 94.44 |
| PC5 | [50] | Boost-RF | N/A | 75.78% |
| | | Bag-RF-FS | N/A | 76.96% |
| | [49] | MLP | 0.24 | 74.80% |
| | This research | RBF | 0.80 | 79.02% |

The result of software defect prediction, obtained using the RBF neural network For KC4 data set, after feature selection can be seen in Table 20, where 0.86 is for *F*-measure and 83.18% is for accuracy. Moreover, KC4 was not a subject of interest for researchers; therefore, no previous work was found.

Different algorithms were used to predict the software defect for MC1 data set, such as Boost-RF, Bag-RF-FS, MLP, and RBF (this research). The latter is the only algorithm to calculate the *F*-measure, and it was 0.99. Also, the RBF neural network attained the highest accuracy with 99.01%, while the rest algorithms obtained 97.06%.

The same algorithms used with the MC1 data set were used with the MC2 data set. *F*-measure obtained using RBF (this research) is the highest with 0.76, while the lowest is 0.33 for Boost-RF and 0.36 for Bag-RF-FS. MLP obtained the best accuracy (97.60%) and RBF (this research) obtained 70.18%, while Boost-RF and Bag-RF-FS obtained 64.86% and 62.16%, respectively.

The highest F-measure was result of MW1 was 0.95 and the lowest was 0.40 obtained using the RBF [this research] and MLP. Moreover, 92.00% was the best accuracy obtained using MLP and followed 89.33% obtained using Boost-RF and Bag–RF-FS. RBF obtained 88.90% accuracy.

The result of Software defect prediction for PC1 data set was summarized in Table 20. The best and lowest *F*-measure obtained were 0.99 and 0.42 obtained using RBF (this research) and MLP [38] respectively. The algorithms DT, MLP [36], and SVM obtained 0.96. RBF obtained the highest accuracy 98.99% followed by MLP [36] (96.56%) and Boost-RF Bag-RF-FS with 96.07%. DT and MLP [36] obtained 93.58% and 93.59%, respectively. SVM obtained 93.10%, and J star and RF obtained the lowest accuracy (91.17%).

Four algorithms' results for PC2 are mentioned in Table 20. RBF (in this research) with 0.99 is the only study which mentioned the *F*-measure. In addition, the highest accuracy obtained is 99.80% using RBF and lowest accuracy is 97.23% using Boost-RF. Meanwhile, Bag-RF-FS and MLP obtained 97.69%.

The best F-measure result obtained for PC3 data set was 0.97 using RBF (this research), and the lowest was 0.14 using MLP. The accuracy (87.34%) obtained using the algorithms Boost-RF and Bag-RF-FS.MLP obtained the lowest accuracy (85.12%).

The lowest and highest *F*-measure for PC4 data set was obtained using MLP and RBF (this research), respectively. The lowest and highest accuracy for PC4 was obtained using RBF (this research) (94.44%) and MLP obtained 88.97%. Boost-RF obtained 91.60%, and Bag-RF-FS obtained 90.81.

The last data set in this study is PC5, where MLP obtained the lowest *F*-measure (0.24) and RBF (this research) obtained the best. The highest and lowest accuracy was obtained using RBF (this research) (79.02%), and the lowest was 74.80% using MLP. Also, Boost-RF obtained 75.78%, and Bag-RF-FS obtained 76.96%.

As seen in Table 20 and discussed in this section. RBF and feature selection proposed in this research has shown better F-measure results than the previous methods for the data sets CM1, JM1, KC2, KC4, MC1, MC2, MW1, PC1, PC2, PC3, PC4, and PC5. However, RBF and feature selection proposed in this research has shown the lowest F-measure results than previous methods for two data

sets—KC1 and KC3. Furthermore, RBF and feature selection proposed in this research has shown better accuracy results than previous methods for the data sets CM1, KC4, MC1, PC1, PC2, PC3, PC4, and PC5. Previous method has shown better accuracy results for data sets including KC1, KC2, KC3, MC1, and MW1.

## 6. Conclusion

Software defect is one of the major causes of software failure. This study introduced a model to predict the software defect. The model consists of the correlation base method for feature selection and RBF for prediction. The study focused on predicting the software defect in fourteen well-known NASA data sets, namely, CM1, JM1, KC1, KC2, KC3, KC4, MC1, MC2, MW1, PC1, PC2, PC3, PC4, and PC5. RBF neural network was used to test the fourteen data sets in two stages: first, using the data set before performing feature selection. In the second stage, the correlation base method was used to select the important features. K-cross-validation methods were performed to divided the data set that is used to train and test the neural network. Furthermore, precision, recall, *F*-measure, and accuracy were performed to evaluate the RBF neural network performance. Results obtained in this study was compared with the results obtained in other methods. RBF neural network has shown a better performance when it is compared with other methods in data such as CM1, KC4, MC1, PC1, PC2, PC3, PC4, and PC5. The main goal of the study was to introduce a model that can detect the software defect with high accuracy; therefore, four questions mentioned earlier in this study were answered. The limitation of the proposed method is that it could not predict the software defect with higher performance for the KC1, KC2, KC3, MC1, and MW1 data sets as some previous methods do. The feature work must focus on these data sets. In conclusion, the proposed method in this study can be used to detect the software defects.

## Data Availability

The data used to support the findings of this study are available in http://openscience.us, https://data.mendeley.com/datasets/923xvkk5mm/1, and http://promise.site.uottawa.ca/SERepository/datasets-page.html.

## Conflicts of Interest

The author declares no conflicts of interest.

## Acknowledgments

The author would like to thank Ajloun National University for the facilities used in this work.

## References

[1] G. Reports, "Business software and services market size, share & trends analysis report by software, by service, by deployment, by end-use, by enterprise size, by region, and segment forecasts, 2021-2028," 2021.

[2] TechRepublic, *Report: Software Failure Caused $1.7 Trillion in Financial Losses in 2017*, TechRepublic, San Francisco, CA, USA, 2018.

[3] Ö. F. Arar and K. Ayan, "Software defect prediction using cost-sensitive neural network," *Applied Soft Computing*, vol. 33, pp. 263–277, 2015.

[4] R. Patton, *Software Testing*, Sams, Bentonville, AR, USA, 2000.

[5] J. Li, P. He, J. Zhu, and M. R. Lyu, "Software defect prediction via convolutional neural network," in *Proceedings of the 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, Prague, Czech Republic, 2017.

[6] M. S. Alkhasawneh, "Hybrid cascade forward neural network with elman neural network for disease prediction," *Arabian Journal for Science and Engineering*, vol. 44, no. 11, pp. 9209–9220, 2019.

[7] J. Lu, L. Tan, and H. Jiang, "Review on convolutional neural network (CNN) applied to plant leaf disease classification," *Agriculture*, vol. 11, no. 8, p. 707, 2021.

[8] M. Abdel-Sattar, A. M. Aboukarima, and B. M. Alnahdi, "Application of artificial neural network and support vector regression in predicting mass of ber fruits (Ziziphus mauritiana Lamk.) based on fruit axial dimensions," *PLoS One*, vol. 16, no. 1, Article ID e0245228, 2021.

[9] M. S. Alkhasawneh, "Olive oil ripping time prediction model based on image processing and neural network," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 1, 2021.

[10] P. Hewage, A. Behera, M. Trovati et al., "Temporal convolutional neural (TCN) network for an effective weather forecasting using time-series data from the local weather station," *Soft Computing*, vol. 24, no. 21, pp. 16453–16482, 2020.

[11] B. Z. Aljunaeidia, M. S. Alkhasawneh, and M. A. BaniYounes, "Isolated Arabic hand written letters recognition based on contour matching and neural network," *Journal of Computer Science*, vol. 14, no. 11, pp. 1565–1576, 2018.

[12] L. J. Cao, K. Chua, W. Chong, H. Lee, and Q. Gu, "A comparison of PCA, KPCA and ICA for dimensionality reduction in support vector machine," *Neurocomputing*, vol. 55, no. 1-2, pp. 321–336, 2003.

[13] M. S. Alkhasawneh, L. T. Tay, U. K. Ngah, M. S. Al-batah, and N. A. Mat Isa, "Intelligent landslide system based on discriminant analysis and cascade-forward back-propagation network," *Arabian Journal for Science and Engineering*, vol. 39, no. 7, pp. 5575–5584, 2014.

[14] M. S. Alkhasawneh, U. K. Ngah, L. T. Tay, N. A. Mat Isa, and M. S. Al-batah, "Determination of important topographic factors for landslide mapping analysis using MLP network," *The Scientific World Journal*, vol. 2013, pp. 1–12, Article ID 415023, 2013.

[15] M. S. Alkhasawneh, U. K. Ngah, L. T. Tay, N. A. Mat Isa, and M. S. Al-batah, "Determination of important topographic factors for landslide mapping analysis using MLP network," *The Scientific World Journal*, vol. 2013, pp. 1–12, Article ID 415023, 2013.

[16] M. Habib, I. Aljarah, H. Faris, and S. Mirjalili, "Multi-objective particle swarm optimization: theory, literature review, and application in feature selection for medical diagnosis," *Evolutionary Machine Learning Techniques*, pp. 175–201, Springer, Singapore, 2020.

[17] M. Rostami, S. Forouzandeh, K. Berahmand, and M. Soltani, "Integration of multi-objective PSO based feature selection and node centrality for medical datasets," *Genomics*, vol. 112, no. 6, pp. 4370–4384, 2020.

[18] B. Nakisa, M. N. Rastgoo, D. Tjondronegoro, and V. Chandran, "Evolutionary computation algorithms for feature selection of EEG-based emotion recognition using mobile sensors," *Expert Systems with Applications*, vol. 93, pp. 143–155, 2018.

[19] M. Sharif, M. A. Khan, Z. Iqbal, M. F. Azam, M. I. U. Lali, and M. Y. Javed, "Detection and classification of citrus diseases in agriculture based on optimized weighted segmentation and feature selection," *Computers and Electronics in Agriculture*, vol. 150, pp. 220–234, 2018.

[20] H. M. Zawbaa, E. Emary, C. Grosan, and V. Snasel, "Large-dimensionality small-instance set feature selection: a hybrid bio-inspired heuristic approach," *Swarm and Evolutionary Computation*, vol. 42, pp. 29–42, 2018.

[21] A. Sukumaran and T. Brindha, "Optimal feature selection with hybrid classification for automatic face shape classification using fitness sorted Grey wolf update," *Multimedia Tools and Applications*, vol. 80, no. 17, pp. 1–22, 2021.

[22] S. Jadhav, H. He, and K. Jenkins, "Information gain directed genetic algorithm wrapper feature selection for credit rating," *Applied Soft Computing*, vol. 69, pp. 541–553, 2018.

[23] D. Kapila and N. Bhagat, "Efficient feature selection technique for brain tumor classification utilizing hybrid fruit fly based abc and ann algorithm," *Materials Today Proceedings*, vol. 51, pp. 12–20, 2022.

[24] X. Yan, C. Wang, D. Hao, and M. Chen, "License plate detection using bayesian method based on edge features," in *Proceedings of the 2021 IEEE 5th International Conference on Cryptography, Security and Privacy (CSP)*, IEEE, Zhuhai, China, 2021.

[25] A. Okutan and O. T. Yıldız, "Software defect prediction using Bayesian networks," *Empirical Software Engineering*, vol. 19, no. 1, pp. 154–181, 2014.

[26] S. Wang, T. Liu, and L. Tan, "Automatically learning semantic features for defect prediction," in *Proceedings of the 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, Austin, TX, USA, 2016.

[27] H. K. D. Kim, "A deep tree-based model for software defect prediction," arXiv, 2018.

[28] D. Bowes, T. Hall, and J. Petrić, "Software defect prediction: do different classifiers find the same defects?" *Software Quality Journal*, vol. 26, no. 2, pp. 525–552, 2018.

[29] Ö. F. Arar and K. Ayan, "A feature dependent Naive Bayes approach and its application to the software defect prediction problem," *Applied Soft Computing*, vol. 59, pp. 197–209, 2017.

[30] Q. Yu, S. Jiang, R. Wang, and H. Y. Wang, "A feature selection approach based on a similarity measure for software defect prediction," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 11, pp. 1744–1753, 2017.

[31] A. Alsaeedi and M. Z. Khan, "Software defect prediction using supervised machine learning and ensemble techniques: a comparative study," *Journal of Software Engineering and Applications*, vol. 12, no. 5, pp. 85–100, 2019.

[32] V. Vashisht, S. Kamya, and M. Vashisht, "Defect prediction framework using neural networks for business intelligence technology based projects," in *Proceedings of the 2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*, Gunupur, India, 2020.

[33] S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," *IEEE Transactions on Reliability*, vol. 62, no. 2, pp. 434–443, 2013.

[34] M. J. Siers and M. Z. Islam, "Software defect prediction using a cost sensitive decision forest and voting, and a potential solution to the class imbalance problem," *Information Systems*, vol. 51, pp. 62–71, 2015.

[35] R. Jindal, R. Malhotra, and A. Jain, "Software defect prediction using neural networks," in *Proceedings of the 3rd International Conference on Reliability, Infocom Technologies and Optimization*, Noida, India, 2014.

[36] L. Zhao, Z. Shang, L. Zhao, T. Zhang, and Y. Y. Tang, "Software defect prediction via cost-sensitive Siamese parallel fully-connected neural networks," *Neurocomputing*, vol. 352, pp. 64–74, 2019.

[37] T. Wang, Z. Zhang, X. Jing, and L. Zhang, "Multiple kernel ensemble learning for software defect prediction," *Automated Software Engineering*, vol. 23, no. 4, pp. 569–590, 2016.

[38] D. R. Ibrahim, R. Ghnemat, and A. Hudaib, "Software defect prediction using feature selection and random forest algorithm," in *Proceedings of the 2017 International Conference on New Trends in Computing Sciences (ICTCS)*, Amman, Jordan, 2017.

[39] I. Arora, V. Tetarwal, and A. Saha, "Open issues in software defect prediction," *Procedia Computer Science*, vol. 46, pp. 906–912, 2015.

[40] M. S. Alkhasawneh, "A comparison between seven heuristic methods to estimate the number of hidden layer neurons in a multi layer perceptron neural network," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 10, no. 2, p. 11, 2021.

[41] Y. Jung and J. Hu, "A K-fold averaging cross-validation procedure," *Journal of Nonparametric Statistics*, vol. 27, no. 2, pp. 167–179, 2015.

[42] C. Catal and B. Diri, "Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem," *Information Sciences*, vol. 179, no. 8, pp. 1040–1058, 2009.

[43] M. K. Jain, "Feature selection in software defect prediction: a comparative study," in *Proceedings of the 6th International Conference-Cloud System and Big Data Engineering*, pp. 658–663, Noida, India, 2016.

[44] P. Venkatesan and S. Anitha, "Application of a radial basis function neural network for diagnosis of diabetes mellitus," *Current Science*, vol. 91, no. 9, pp. 1195–1199, 2006.

[45] G. A. Montazer and D. Giveki, "An improved radial basis function neural network for object image retrieval," *Neurocomputing*, vol. 168, pp. 221–233, 2015.

[46] P. Evans, K. C. Persaud, A. S. McNeish, R. W. Sneath, N. Hobson, and N. Magan, "Evaluation of a radial basis function neural network for the determination of wheat quality from electronic nose data," *Sensors and Actuators B: Chemical*, vol. 69, no. 3, pp. 348–358, 2000.

[47] S. Aleem, L. F. Capretz, and F. Ahmed, "Benchmarking machine learning techniques for software defect detection," *International Journal of Science and Engineering Applications*, vol. 6, no. 3, pp. 11–23, 2015.

[48] R. B. Bahaweres, F. Agustian, I. Hermadi, A. Imam Suroso, and Y. Arkeman, "Software defect prediction using neural network based SMOTE," in *Proceedings of the 2020 7th International Conference on Electrical Engineering, Computer Sciences and Informatics (EECSI)*, IEEE, Yogyakarta, Indonesia, 2020.

[49] 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.

[50] A. Iqbal, S. Aftab, I. Ullah, M. Salman Bashir, and M. Anwaar Saeed, "A feature selection based ensemble classification framework for software defect prediction," *International Journal of Modern Education and Computer Science*, vol. 11, no. 9, pp. 54–64, 2019.

[51] K. O. Elish and M. O. Elish, "Predicting defect-prone software modules using support vector machines," *Journal of Systems and Software*, vol. 81, no. 5, pp. 649–660, 2008.

[52] S. Agarwal and D. Tomar, "A feature selection based model for software defect prediction," *International Journal of Advanced Science and Technology*, vol. 65, pp. 39–58, 2014.