A Feature Selection-Based K-NN Model for Fast Software Defect Prediction

Joseph Bamidele Awotunde[1(✉)], Sanjay Misra[2], Abidemi Emmanuel Adeniyi[2], Moses Kazeem Abiodun[1,3], Manju Kaushik[4], and Morolake Oladayo Lawrence[5]

[1] Department of Computer Science, University of Ilorin, Ilorin, Nigeria
awotunde.jb@unilorin.edu.ng
[2] Department of Computer Science and Communication, Ostfold University College, Halden, Norway
Sanjay.misra@hiof.no, adeniyi.emmanuel@lmu.edu.ng
[3] Department of Computer Science, Landmark University, Omu-Aran, Nigeria
moses.abiodun@lmu.edu.ng
[4] Amity Institute of Information Technology, Amity University, Jaipur, Rajasthan, India
mkaushik@jpr.amiy.edu
[5] Department of Computer Science, Baze University, Abuja, Nigeria
morolake.lawrence@bazeuniversity.edu.ng

Abstract. Software Defect Prediction (SDP) is an advanced technological method of predicting software defects in the software development life cycle. Various research works have been previously being done on SDP but the performance of these methods varied from several datasets, hence, making them inconsistent for SDP in the unknown software project. But the hybrid technique using feature selection enabled with machine learning for SDP can be very efficient as it takes the advantage of various methods to come up with better prediction accuracy for a given dataset when compared with an individual classifier. The major issues with individual ML-based models for SDP are the long detection time, vulnerability of the software project, and high dimensionality of the feature parameters. Therefore, this study proposes a hybrid model using a feature selection enabled Extreme Gradient Boost (XGB) classifier to address these mentioned challenges. The cleaned NASA MDP datasets were used for the implementation of the proposed model, and various performance metrics like F-score, accuracy, and MCC were used to reveal the performance of the model. The results of the proposed model when compared with state-of-the-art methods without feature selection perform better in terms of the metrics used. The results reveal that the proposed model outperformed all other prediction techniques.

Keywords: Software defect prediction · Machine learning · Extreme gradient boost · Feature selection · Prediction · Software development life cycle

A. E. Adeniyi and M. K. Abiodun—Landmark University SDG 4 (Quality Education)

M. K. Abiodun—Landmark University SDG 16 (Peace and Justice, Strong Institution)

A. E. Adeniyi—Landmark University SDG 11 (Sustainable Cities and Communities)

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
O. Gervasi et al. (Eds.): ICCSA 2022 Workshops, LNCS 13380, pp. 49–61, 2022.
https://doi.org/10.1007/978-3-031-10542-5_4

1 Introduction

In recent time, the software has been the heart of the technological era where factually all organizations are in needs of one software or others [1]. But if there are frequent software defects the main aim of creating them is defeated, and reduce its credibility and reliability [2]. ally Software is the heart of this technological era. If there are frequent defects in software, it reduces reliability and credibility [3, 4]. This resulted into software dissatisfied the customers and thereby worsening the software company's images [4, 5]. If flaws are discovered beginning of the software development process, resources can be allocated to only the software components that are required. Hence, software defect prediction (SDP) is one of the key features with the continual increases in software industry for manufacturing quality software. The discovery of defects in software in time will help the industries in saving effort, money and resources.

Machine learning (ML) models have been gaining greater attention in recent time in the prediction of software defection [6]. This was done in other to give timely delivery of software products with optimal deployment of the available resources [7]. But the problems of imbalanced and high dimensionality of software datasets have hindered the performance of the ML-based models [8]. Accuracy has been identified as a good evaluation metric for comparison when consider the binary problems [9], but in a situation where the number of instances of one class is huge that another, the accuracy of such model will not be efficient. For instance, if there are 10 defects out of 100 instances and 90 instances are free of defect, the result of the validation results will be affected when ML-based model is trained on such imbalanced data. The results will of sequence be biased towards the mainstream class. The problem of imbalance is common among real-world problems [10, 11].

The feature selection can be used to handle large dataset with huge amount of features that the classifiers need to select from before the prediction or classification of such dataset [12]. This is used to identify the most relevant parameters that can greatly contributed to high efficient software defect prediction using various ML-based models. The redundant within datasets can remove using feature selection techniques, the classifiers will remove non-independent features from the dataset [13]. The feature selection method will enhance the efficient handling of irrelevant and redundant parameters in the datasets used for software defect prediction [14]. Feature selection is very important in software defect prediction. The application of this classifier has a great influence on the prediction accuracy, thus become necessary to use proper feature selection classifier for dimensionality reduction.

Therefore, this paper proposes a hybrid feature selection enabled Extreme Gradient Boost model for software defect prediction. The feature selection was used to reduce the dimensionality of the dataset before applying XGBoost classifier on the software defect dataset. The case of imbalance and feature redundancy dataset was handle using data normalization mothed. The XGBoost ensemble algorithm was used because of stronger robustness to irrelevant and redundant features. The major contributions of the study are:

(i) the application of feature selection to reduce the high dimensionality of the software defect dataset.

(ii) resampling method was applied to handle the imbalanced instances within the dataset.
(iii) the prediction of the software defection dataset was perform using XGBoost algorithm to successfully classifier irrelevant and redundant data after the application of feature selection and data normalization methods.

2 Related Work

ML-based models have been applied by various researchers to solve twofold (binary) classification in various fields like in rainfall prediction [15, 16], sentiment analysis [17–19], network intrusion detection [11, 20–22], and software defect prediction [1, 2, 4, 23–25]. Some of the SDP related works are discussed in this study.

The authors in [26] reviewed the analysis of ML-based models on SDP using twelve cleaned NASA datasets for the performance of their classifiers. The algorithms employed are Support Vector Machine (SVN), K-Nearest Neighbor (K-NN), Naïve Bayes (NB), Radial Basis Function (RBF), Decision Tree (DT), Multi-Layer Perceptron (MLP), and Random Forest (RF). The performance metrics used on the analyzed models are recall, accuracy, ROC, MCC, and F1-score. The results of the experiment shown that RF performed better when compared with other classifiers followed by the SVM model. In another similar work, the authors in [27] used ensemble classification model after using feature selection to reduce the irrelevant features from the dataset used for testing the model. The proposed model was implemented in a binary dimension, using feature selection with the classifier and secondly applied the model without feature selection method. The NASA datasets was used with various performance metrics to evaluated the performance of the proposed models. The results of the model are compared with other state-of-the-art used prediction methods. The results shown great improvement on some datasets but the model was unable to perform well in class imbalanced dataset because the issue was not well resolved by the proposed model.

The authors in [28], proposed six classifiers for SDP techniques using Principal Component Analysis (PCA) for feature reduction, the algorithms used are Holographic Networks, Layered Neural network, LR, and Discriminant Analysis. The performance metrics are Misclassification Rate, Verification Cost, Predictive Validity, and Achieved Quality. The results of the proposed models revealed 100% accuracy especially the one with the application of PCA for dimensionality reduction without error. In [4], the authors provided experimental appraisal of SDP models using several boosting enabled ensemble techniques on three open source JAVA projects. Stable performance metrics like AUC, Balance, and G-Mean are used to evaluated the models. The resampling methods with four ensemble classifiers were used within the JAVA projects. The resampling methods enabled classifiers performed better when compared with classifiers without resampling techniques. This shown that the resampling has great impact on the ensemble classifiers, and has significantly improved the SDP models when compared with classic boosting classifiers. Among the resampling techniques used, RUSBoost performed greatly followed by MSMOTEBoost, and the poorest among them is the SMOTEBoost.

In another study by [1], presented a model using several ensemble learning for prediction of faults within software modules. The proposed system combined both linear and non-linear combination rule for the ensemble models. Several software fault datasets

that are publicly available was used for the design and conduct of the experiments. The proposed system was able to predict software faults with high accuracy, and the results was reliable throughout the datasets used for performance evaluation. The prediction at level l (Pred(l)) was further used on the ensemble classifiers to measures the completeness of the results. The result shows that average relative error of the number of modules in a dataset is less than or equal to a threshold value of l. this analysis and evaluation using the metric confirmed the effectiveness of the proposed system for the prediction of software faults. The ensemble techniques performance improved when compared with single fault prediction method for software faults prediction. The main contribution of the model is the quick identification of the faults in the software products with the utilization of testing resources.

In [23], the authors presented a model for SDP contains four stages: (i) feature selection, (ii) Pre-processing (iii) classification, and (iv) Reflection of results. The feature selection model was use to remove irrelevant features from the dataset before applying ensemble learning classifiers. The cleaned NASA MDP datasets was used for the implementation of the proposed model by the authors. Various performance metrics like accuracy, F1-score, MCC, and ROC for the evaluation of the model. The model was tested on each of the dataset to compare them to know the highest scores among the six datasets. Ten well-known supervised classifiers models were used to compared with the proposed model, the search methods and the results of the proposed system revealed that the model outperformed all the other classifiers methods. In a study by [29], the authors proposed SVM for the prediction of software faults using NASA datasets for the implementation of the model. The proposed framework was compared with other models like LR, K-NN, RF, NB, RBF, and MLP. The results show that the proposed method outperformed some of the classification methods used to test the performance.

The authors in [30] shown the importance of feature selection for SDP systems by show that some parameters or features are more relevant than others. The ANN enabled with feature selection for the implementation of the framework. The selected features are used to predict the SDP using ANN classifiers. The Gaussian Kernel SVM and JM1 NASA dataset was used to tested the performance of the proposed method for experiment. From the results by the proposed model, SVM outperformed the model in the twofold defect classification. In [31], the authors used hybrid Genetic Algorithm (GA) with Deep Neural Network (DNN) SDP using several datasets from PROMISE repository. The Hybrid GA is used for feature selection in other to select the optimal features for the model, and DNN was used to performed the prediction of the system. The results of the proposed model outperformed other techniques used to test the performance of the model.

From the existing research, it was discovered that the imbalanced nature of software data can hinder the performance of models leading to incorrect interpretations of results, and high dimensionality of features in any dataset will reduce the performance of any ensemble algorithms. Therefore, this study proposed a hybrid model can consist of feature selection, data normalization, and XGBoost for the classification of software products to predict whether a software has defects or not.

3 Materials and Methods

3.1 Proposed Hybrid Software Defect Prediction Model

The proposed system aims at enhancing the performance of SDP for software development faults prediction using feature selection and XGBoost techniques on the NASA datasets. In recent years' various techniques like ML-based models and data mining techniques have been used to resolve various problems involved software defects prediction system performance. To improve the performance of SDP for software products, the proposed model reduced the number of features using for the prediction of software defects in software development. Figure 1 presents the framework for the proposed model. The model consists of the followings processes: (i) pre-processing, (ii) imbalance using resampling method, (iii) feature selection, and (iv) prediction. The stages are discussing in details in the following subsection.

3.2 The Pre-processing Stage

To provide an appropriate data for the SDP proposed framework, various pre-processing steps were performing on the NASA dataset. The following are the steps follows to reform the dataset used for the purpose of this study:

(i) Removing duplicated instances: software modules with same class labels and software metric values is referred to has duplicated instances for instance defective labels. In a real-world problem, this situation is very thinkable. Regrettably, the machine learners are going to be affect negatively if there is duplicated instances, thus cause overoptimistic if they are categorized fittingly as a part of test data. This will result to over-pessimistic presentations if they are misclassified as a part of test data. Furthermore, this can create time-consuming during training process, and reduce the improvement in the performance of the models and classifiers. Hence, it become very necessary to remove duplicated instances in the datasets;

(ii) Replacement of Misplaced labels: There many instance values with multiple software metrics generally. Due to some reasons, there can be more than one values of an instance that can be missing especially when data were collected carelessly. It became very often that the instance cannot satisfy input requirement of the proposed model since this missing values has to be processed. For any missing value in this study, we replace it with then average of the equivalent metric. For instance, suppose both mt_{99} *and* mt_{100} are missed i.e. NaN, a metric mt and it clarifications $\{mt_1, \ldots.., mt_{100}\}$. the two missing values can be replaced using Eq. (1):

$$mt_{99} = mt_{100} = \frac{1}{98} \sum_{i=1}^{98} mt, \tag{1}$$

(iii) Data Standardization: the values of various software metrics are of several order of magnitude, thus we need to perform data normalization on these metrics. The min-max-normalization the most commonly used method was used in this study to transform in the interval [0, 1] of all the values, and normalize the data [32, 33].

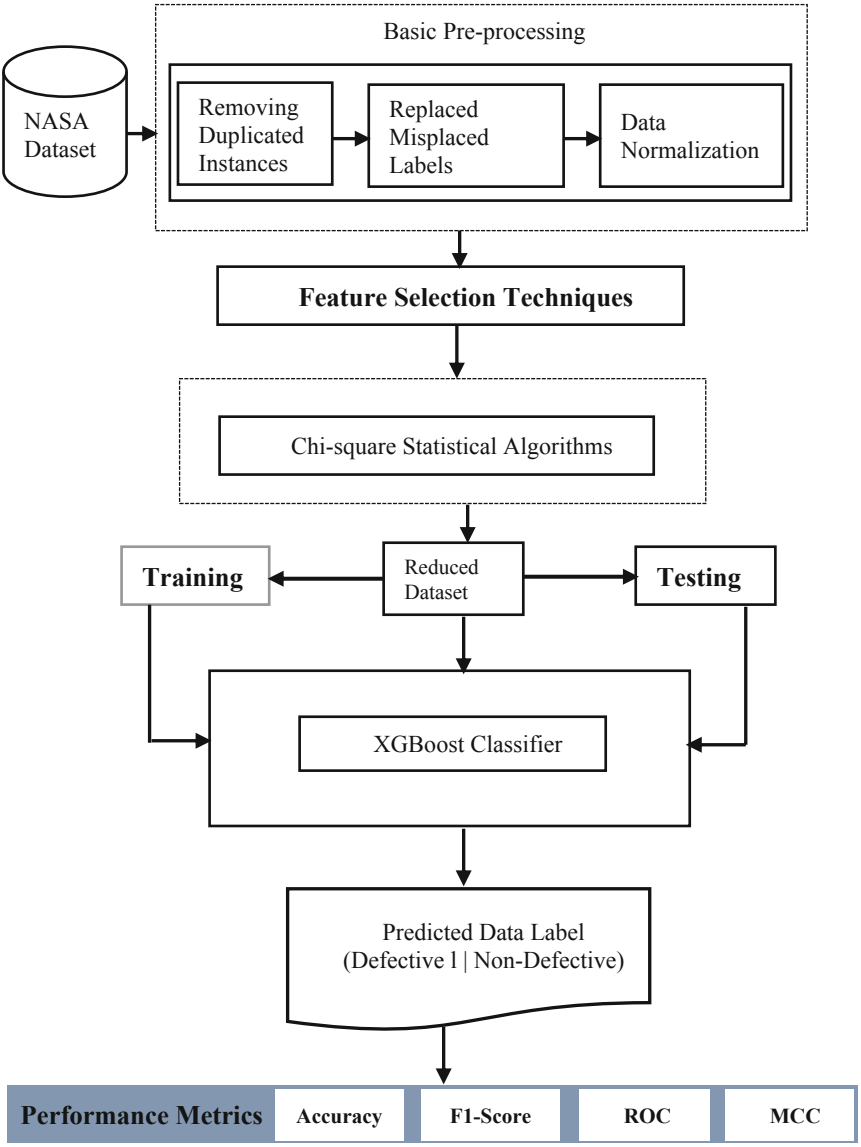

Fig. 1. The proposed Software Defects Prediction Architecture

the minimum and maximum values of a given metric x are $\max(x)$ *and* $min(x)$, one-to-one. The value $\widetilde{x_i}$ for each value of x_i of metric x, can be computed as

$$\widetilde{x_i} = \frac{x_i - min(x)}{max(x) - min(x)}, \tag{2}$$

3.3 The Chi-Square Statistical Model for Feature Selection

The Chi-square statistical model was use for feature selection in this study. There are always two variables in feature selection that normally refer to incidence frequency of feature t and the category C is the likelihood of incidence respectively. In software defects prediction, in the proposed model it is consider whether features t and C are autonomous. It means if features t and C are independent, then feature t cannot be used to decide whether a label fit in to category C. In training, to find out the extent of t and C whether they are correlated is very difficult especially when not independent. Hence, Chi-square test can be used to define their significance. The relationship between feature t and category C is measures using the Chi-square statistical techniques. The label feature t and a category C was represented using bidirectional queue.

If feature t and the category C assume, and accords with the chi-square distribution with first-order degree of freedom. The more category label the feature carries is the higher chi-square score for the category c, then there is great relevance between t *and* C_j. Then the chi-square score of feature t for category C can be define as follows:

$$X^2 = (t, c)\frac{N(AD - BC)^2}{(A + C)(A + B)(B + BD)(C + D)} \tag{3}$$

The result in Eq. 3 shown the relevance between feature t and category C_j. The relevant of feature t will be the lager the class category C_j are autonomous, when $CHI\,(t, C_i)$, then the feature t and label class C_j is independent. Equation 3 can be used to calculated the value for a single class denoted by $X^2(t, c)$, but by combining all the value of $X^2(t, c)$ for feature label t for all classes, then we first calculate the $X^2(t, c)$ for all the feature of instance t for all classes. Then testing feature t for all the distinct $X^2(t, c)$ score, the number of the classes is then be m:

$$XAVG^2(t) = \sum\nolimits_{i-1}^{m} p(c)X^2(t, c) \tag{4}$$

The average of $X^2(t, c)$ score of the feature label t for all classes is been calculated by Eq. 4.

$$\aleph MAX^2(t) = \max_{1 \le i \le m} \left\{ X^2(t, c) \right\} \tag{5}$$

Equation (5) is used to calculates the maximum $X^2(t, c)$ of feature label t for all classes. The feature label is finally sorted by the X scores, and a suitable quantity of feature label are designated permitting to the given threshold value.

3.4 The Extreme Gradient Boost

This is a family of GB ensemble that uses second-order products of the forfeiture function to find the finest base classifier more perfectly and proficiently. The XGB utilizes the second-order gradients, but the GB uses gradients to fit a new base classifier. Gradient Boosting algorithm is a supervised machine learning technique which tries to predict the output by combining the predictions of multiple weaker or low performing models. This ML-based method uses ensemble of DT with GB framework [34]. The error generated from the previous models is been used by the ML-based methods using boosting to minimized the errors and influence the performance of the model for enhancement. The GB minimize the errors of the ensemble efficiently when compare with Boosting. The Gradient Descent model is used by the GB to reduce errors in successive algorithms. The XGBoost is used for additional improvement to the tree-based models uses the benefits of GB model remove overfitting and offer ability to escape bias circumstances. tree-based algorithms are added by the use of XGBoost which uses the benefits of Gradient Boosting algorithm to provide facility to avoid bias or overfitting conditions, tree thinning and matching learning.

3.5 The NASA Datasets

Four widely used datasets from NASA MDP repository was used in this study, the datasets were used because selection an appropriate dataset is the most important aspect of classification and prediction of SDP for the proposed model. The NASA MDP datasets used were: PC4, PC5, JM1, and KC1 and this was presented in Table 1. Each dataset is in the form of an attribute set called module with various records where each record represents a module. Each attribute is generated during the dataset capturing and development called the software metrics (Table 2).

Table 1. D" NASA cleaned datasets

Dataset	Features	Elements	Defects	Non-Defects	Defects (%)
PC4	37	1270	176	1094	13.8
PC5	39	1694	458	1236	27.0
JM1	22	7720	1612	6108	20.8
KC1	22	1162	294	868	25.3

Table 2. Confusion matrix.

		Predicted	
		Class a	Class b
Real	Class a	TP	FN
	Class b	FP	TN

3.6 Performance Evaluation Metrics

Accuracy is the percentage of correctly classified instances as presented by Eq. (6)

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{6}$$

The F1-Score is the harmonic mean between the recall and precision as illustrated by Eq. (7)

$$F1_{Score} = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{7}$$

The ratio of predicted and the observed classifications is used to computed the MCC as shown by Eq. (8)

$$MCC = \frac{TN * TP - FN * FP}{\sqrt{(FP + TP)(FN + TP)(TN + FP)(TN + FN)}} \tag{8}$$

4 Results and Discussion

The experiment of the study was done using Core™ i5-4200U CPU @ 1.60 GHz 2.30 Intel® GHz 8.0 GB RAM laptop running on Windows 10.1 (64bit). And WEKA programing. The proposed system used the benchmark datasets with experiment setup respectively. the experiment was perform using various performance metrics for the proposed system evaluation.

The proposed model used four NASA software that is publicly available software defects datasets. As say earlier from this study, the study uses feature selection to remove irrelevant features to enable the model to performance efficiently. This was done due to the missing values, duplicate instances, and conflicting features that the NASA datasets suffer from. Table 3 shows the original features and the selected features using Chi-square statistical model on the four NASA datasets.

Table 3. The number of features before and after features selection algorithm

	Dataset	PC4	PC5	JM1	KC1
No of Features	Before	37	39	21	21
	After	17	19	8	10
	Retention ratio (%)	45.95	48.72	38.10	47.62

Table 4 displayed the performance evaluation of the XGBoost model in terms of accuracy, F1-score, and MCC scores over the 4 software defects datasets. This show the results when the feature selection algorithm was not used on the datasets. The XGBoost model noticeable performance with PC4 dataset, and follow by the PC5 and the KC1 was the worst of all the dataset.

Table 4. The performance of the XGBoost model without feature selection

Dataset	Accuracy	F1-score	MCC
PC4	95.324	0.597	0.517
PC5	93.021	0.473	0.421
JM1	89.457	0.376	0.345
KC1	87.872	0.321	0.296

Table 5 displayed the performance evaluation of the proposed model in terms of accuracy, F1-score, and MCC scores over the 4 software defects datasets. This show the results XGBoost when the feature selection algorithm was not used on the datasets. The model noticeable performance with PC4 dataset, and follow by the PC5 and the KC1 was the worst of all the dataset. The feature selection really improved the prediction model on the datasets with an average of 4.197%.

Table 5. The performance of the XGBoost model with feature selection

Dataset	Accuracy	F1-score	MCC
PC4	99.521	0.743	0.765
PC5	96.386	0.532	0.553
JM1	91.035	0.412	0.394
KC1	88.278	0.395	0.353

5 Conclusion

The study proposed a model for the prediction of software defects prone to software modules. Early detection of defect in software modules can really reduce the cost of software development life cycle process. The proposed system used Chi-square statistical algorithm for feature selection in order to eliminate features that are irrelevant in the classification process of the model. Data normalization was used to reduce the high dimensionality of the datasets used to the performance of the proposed model. XGBoost was use for the prediction of the software metrics module on the four NASA datasets. Consider the performance of the model on the four datasets, the XGBoost with feature

selection performed better on the PC4 dataset, and the worst of all the dataset is KC1. The results of the proposed framework show that the feature selection enabled model greatly increase the performance of the prediction model when compare with the model without the feature selection. Nevertheless, future research still need to optimize various classifiers with extensive set of features for more variants to be selected for ML-based models. And the applications of Deep learning models with feature selection will surely improve the classification of software modules for defects.

References

1. Rathore, S.S., Kumar, S.: Towards an ensemble based system for predicting the number of software faults. Expert Syst. Appl. **82**, 357–382 (2017)
2. Laradji, I.H., Alshayeb, M., Ghouti, L.: Software defect prediction using ensemble learning on selected features. Inf. Softw. Technol. **58**, 388–402 (2015)
3. Abisoye, O.A., Akanji, O.S., Abisoye, B.O., Awotunde, J.: Slow hypertext transfer protocol mitigation model in software defined networks. In: 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy, ICDABI 2020, 9325601 (2020)
4. Malhotra, R., Jain, J.: Handling imbalanced data using ensemble learning in software defect prediction. In: 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), pp. 300–304. IEEE (2020)
5. Awotunde, J.B., Ayo, F.E., Ogundokun, R.O., Matiluko, O.E., Adeniyi, E.A.: Investigating the roles of effective communication among stakeholders in collaborative software development projects. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2020, 12254 LNCS, pp. 311–319 (2020)
6. Awotunde, J.B., Folorunso, S.O., Bhoi, A.K., Adebayo, P.O., Ijaz, M.F.: Disease diagnosis system for IoT-based wearable body sensors with machine learning algorithm. Intelligent Systems Reference Library **2021**(209), 201–222 (2021)
7. Awotunde, J.B., Misra, S.: Feature extraction and artificial intelligence-based intrusion detection model for a secure internet of things networks. Lecture Notes Data Eng. .ications Technol. **2022**(109), 21–44 (2022)
8. Behera, R.K., Shukla, S., Rath, S.K., Misra, S.: Software reliability assessment using machine learning technique. In: Gervasi, O., et al. (eds.) Computational Science and Its Applications – ICCSA 2018: 18th International Conference, Melbourne, VIC, Australia, July 2-5, 2018, Proceedings, Part V, pp. 403–411. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-95174-4_32
9. Chicco, D., Jurman, G.: The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC Genomics **21**(1), 1–13 (2020)
10. Shukla, S., Behera, R.K., Misra, S., Rath, S.K.: Software reliability assessment using deep learning technique. In: Chakraverty, S., Goel, A., Misra, S. (eds.) Towards Extensible and Adaptable Methods in Computing, pp. 57–68. Springer, Singapore (2018). https://doi.org/10.1007/978-981-13-2348-5_5
11. Awotunde, J.B., Chakraborty, C., Adeniyi, A.E.: Intrusion detection in industrial internet of things network-based on deep learning model with rule-based feature selection. Wirel. Commun. Mob. Comput. **2021**(2021), 7154587 (2021)
12. Ogundokun, R.O., Awotunde, J.B., Sadiku, P., Adeniyi, E.A., Abiodun, M., Dauda, O.I.: An enhanced intrusion detection system using particle swarm optimization feature extraction technique. Procedia Computer Science **193**, 504–512 (2021)

13. Jagdhuber, R., Lang, M., Stenzl, A., Neuhaus, J., Rahnenführer, J.: Cost-Constrained feature selection in binary classification: adaptations for greedy forward selection and genetic algorithms. BMC Bioinformatics **21**(1), 1–21 (2020)
14. Kumari, A., Behera, R.K., Sahoo, B., Sahoo, S.P.: Prediction of link evolution using community detection in social network. Computing, 1–22 (2022)
15. Mishra, N., Soni, H.K., Sharma, S., Upadhyay, A.K.: Development and analysis of artificial neural network models for rainfall prediction by using time-series data. International Journal of Intelligent Systems Applications, **10**(1) (2018)
16. Zhang, X., Mohanty, S.N., Parida, A.K., Pani, S.K., Dong, B., Cheng, X.: Annual and non-monsoon rainfall prediction modelling using SVR-MLP: an empirical study from Odisha. IEEE Access **8**, 30223–30233 (2020)
17. Jagdale, R.S., Shirsat, V.S., Deshmukh, S.N.: Sentiment analysis on product reviews using machine learning techniques. In: Mallick, P.K., Balas, V.E., Bhoi, A.K., Zobaa, A.F. (eds.) Cognitive Informatics and Soft Computing. AISC, vol. 768, pp. 639–647. Springer, Singapore (2019). https://doi.org/10.1007/978-981-13-0617-4_61
18. Hassonah, M.A., Al-Sayyed, R., Rodan, A., Ala'M, A.Z., Aljarah, I., Faris, H.: An efficient hybrid filter and evolutionary wrapper approach for sentiment analysis of various topics on Twitter. Knowledge-Based Syst.**192**, 105353 (2020)
19. Rehman, A.U., Malik, A.K., Raza, B., Ali, W.: A hybrid CNN-LSTM model for improving accuracy of movie reviews sentiment analysis. Multimedia Tools and Applications **78**(18), 26597–26613 (2019)
20. Awotunde, J.B., Abiodun, K.M., Adeniyi, E.A., Folorunso, S.O., Jimoh, R.G.: A deep learning-based intrusion detection technique for a secured IoMT system. Communications in Computer and Information Science, 2022, 1547 CCIS, pp. 50–62 (2021)
21. Verma, A., Ranga, V.: Machine learning based intrusion detection systems for IoT applications. Wireless Pers. Commun. **111**(4), 2287–2310 (2020)
22. Amouri, A., Alaparthy, V.T., Morgera, S.D.: A machine learning based intrusion detection system for mobile Internet of Things. Sensors **20**(2), 461 (2020)
23. Matloob, F., Aftab, S., Iqbal, A.: A framework for software defect prediction using feature selection and ensemble learning techniques. International Journal of Modern Education Computer Sci. **11**(12) (2019)
24. Yalçıner, B., Özdeş, M.: Software defect estimation using machine learning algorithms. In: 2019 4th International Conference on Computer Science and Engineering (UBMK), pp. 487–491. IEEE (2019)
25. Arar, Ö.F., Ayan, K.: Software defect prediction using cost-sensitive neural network. Appl. Soft Comput. **33**, 263–277 (2015)
26. Iqbal, A., et al.: Performance analysis of machine learning techniques on software defect prediction using NASA datasets. Int. J. Adv. Comput. Sci. Appl **10**(5), 300–308 (2019)
27. Iqbal, A., Aftab, S., Ullah, I., Bashir, M.S., Saeed, M.A.: A feature selection based ensemble classification framework for software defect prediction. Int. J. Modern Education Comput. Sci. **11**(9), 54 (2019)
28. Lanubile, F., Lonigro, A., Vissagio, G.: Comparing models for identifying fault-prone software components. In: SEKE, pp. 312–319 (1995)
29. Elish, K.O., Elish, M.O.: Predicting defect-prone software modules using support vector machines. J. Syst. Softw. **81**(5), 649–660 (2008)
30. Gondra, I.: Applying machine learning to software fault-proneness prediction. J. Syst. Softw. **81**(2), 186–195 (2008)
31. Manjula, C., Florence, L.: Deep neural network based hybrid approach for software defect prediction using software metrics. Clust. Comput. **22**(4), 9847–9863 (2018). https://doi.org/10.1007/s10586-018-1696-z

32. Witten, I.H., Frank, E.: Data mining: practical machine learning tools and techniques with Java implementations. ACM SIGMOD Rec. **31**(1), 76–77 (2002)
33. Dai, H., Hwang, H.G., Tseng, V.S.: Convolutional neural network based automatic screening tool for cardiovascular diseases using different intervals of ECG signals. Comput. Methods Programs Biomed. **203**, 106035 (2021)
34. Awotunde, J.B., et al.: An improved machine learnings diagnosis technique for COVID-19 pandemic using chest X-ray images. Communications in Computer and Information Science, 2021, 1455 CCIS, pp. 319–330 (2021)