# Machine Learning for Software Bug Prediction on the JM1 Dataset

[1] Pothuri Jyothi, [2] Ch. Naveen
[1] PG Scholar, Department of CSE, MAM WOMEN'S COLLEGE,KESANUPALLI.
[2] Associate Professor, Department of CSE, MAM WOMEN'S COLLEGE,KESANUPALLI.

## Abstract—

In order to get the most out of complicated software development processes, it is crucial to guarantee the quality of software systems. Forecasting and early detection of any software component problems or difficulties is a critical part of the overall process. This research analyzes the accuracy of prediction made by different machine learning algorithms for software defect detection using the JM1 dataset that was curated by NASA. These algorithms include Naïve Bayes, Decision Trees, Random Forest, Support Vector Machine, Logistic Regression, Artificial Neural Networks, and K-Nearest Neighbors. The paper emphasizes the need of using machine learning for early fault finding in software development.With an impressive 81% accuracy, precision, recall, and little Root-Mean-Square Error, the Random Forest model emerges as the clear winner in the trial findings. In addition, the researchers compared their findings to those of other studies, which shed light on the strengths and weaknesses of the suggested approach. Suggestions for further research are made in the study's conclusion.

## Index Terms—

Predicting software bugs, faults, and using machine learning

## I.INTRODUCTION

For the complicated software development process to be a success, it is essential that software systems be accurate and of high quality. Crucial to the process is the early identification and prediction of potential problems or errors in software components. In order to reduce the total cost of software maintenance, increase customer happiness, and improve software quality, early problem discovery is crucial. Evidence of the changing nature of software engineering data is shown by the JM1 dataset in particular. The JM1 program is a real-time predictive ground system that uses simulations to generate predictions. This dataset, which has been vetted by NASA, includes actual occurrences from that system. The complexity of software systems increases the difficulty of finding faulty components. Finding these problems is the first step in fixing software system quality. Machine learning algorithms have shown their effectiveness in this setting by reducing software errors and forecasting their occurrence. Machine learning, and more especially supervised learning algorithms, have shown tremendous promise in several domains, software engineering being only one of them. Due to their ability to detect patterns and correlations among massive datasets, these algorithms are ideal for program code that anticipates module issues. Consequently, this research will examine the effectiveness of several machine learning techniques using the JM1 dataset.This work aims to provide significantly more knowledge to the software quality prediction area by focusing on certain elements of the JM1 dataset. In what follows, we'll go over the challenges of module failure prediction, the intricacies of the JM1 dataset, and the crucial role that machine learning plays in overcoming these challenges. Section II delves into the linked literature. The datasets and assessment procedure using machine learning methods are covered in Section III. Section V presents the findings and recommendations for further research, whereas Section IV details the experimental outcomes.

## II. RELATED WORK

Predicting software defects and bugs using machine learning (ML) methods has been a hot topic in the last several years. Here we take a look at a number of seminal research articles that have added to this expanding area of study. In study [1], the researchers examined how Artificial Neural Networks, Decision Trees, and Naïve Bayes may be used to anticipate software bugs. Decision Trees, Artificial Neural Networks, and Linear Autoregression (AR) models outperformed POWM and AR in terms of root-mean-squared error (RMSE), according to the study that was based on three real-world testing and debugging datasets. With an average accuracy value over 97%, precision

measurements showed that all ML algorithms performed well when it came to bug prediction. In terms of average recall values, ANNs reached 99% and NB 96%; nevertheless, the DT classifier showed the best performance. When compared using the F-measure, DT was shown to be more effective than ANNs and NB. Using decision trees, logistic regression, random forests, and naive Bayes, Delphine Imaculate et al. [2] presented a methodical approach to bug prediction. Their research shows that the random forest model is the best with a 97% accuracy rate, using datasets from 15 GitHub Java and Python projects. In spite of its emphasis on thorough data preparation, the article did note several limitations, such as not delving into artificial neural networks or addressing the need for generalizability across different types of projects.In their extensive review of ML techniques, Sharma et al. [3] covers everything from neural networks and support vector machines (SVMs) to decision trees, evolutionary algorithms, ensemble learning, and more. One contribution came from Tanaka et al. [4], who tested the efficacy of the automated ML library auto-sklearn in predicting software module problems across different versions. Automated methods showed promise in defect prediction, with results that were competitive with those of the random forest model. The research highlighted the importance of data volume in automated machine learning effectiveness and the influence of changes in fault reasons between releases.To forecast software defects, X. Zhao et al. [5] compared six ML algorithms and came up with the Importance Descending Order Exclusion (IDOE) technique. According to the research, Random Forests are better for bigger datasets while LibSVM are better for smaller ones. The article provided helpful information on attribute selection techniques and classifier appropriateness for certain dataset sizes, while also recognizing that algorithm performance varies between datasets.Five ML classification methods were comprehensively assessed across NASA defect prediction datasets by Aydin and R. Samli [6]. The research highlighted the persistent high success rates of Random Forest, which provided unambiguous numerical performance measures, emphasizing its supremacy. The work failed to explicitly state its limits when making recommendations for future research, which might need more examination into issues like dataset imbalance or overfitting. Using a variety of ML algorithms—with an emphasis on feature selection and cross-validation—P and Kambli [7] presented an all-encompassing model for bug prediction. With an astounding 82.77% accuracy, Artificial Neural Networks (ANN) blew away other algorithms in the research, which was based on the PROMISE JM1 dataset. The paper's strength was in its technique for bug prediction, which took into account crucial factors like feature selection and model validation, while also recognizing the need of dataset specificity. In their hybrid model, Prabha and Shivakumar [9] combined several ML algorithms with Principal Component Analysis (PCA) to reduce features. Among the results, Support Vector Classifier (SVC) stood out for its 98.70% accuracy rate in defect prediction across all datasets. There has to be additional data for further validation, and the research highlighted how feature selection tactics affect classifier accuracy.For the purpose of software defect prediction, Manjula et al. [10] investigated a number of classification approaches, including supervised, unsupervised, and semi-supervised ones. The significance of using quantitative software measures for early defect discovery was emphasized in the article. By contrasting several methods, you might learn about their advantages and disadvantages. By using machine learning algorithms, feature selection, and K-means clustering, Khalid et al. [11] made a contribution to software fault prediction. Notable findings were obtained on the CM1 dataset by their investigation. Specifically, SVM and improved SVM models attained accuracy rates of 99% and 99.80%, respectively, surpassing other models. Although there were constraints, such as the small dataset, using ensemble approaches and Particle Swarm Optimization (PSO) helped optimize the models even more. Commonly used metrics and datasets were outlined by Malhotra [13], who also classified ML approaches. The necessity for more comparisons between ML and Logistic Regression approaches is highlighted by the consistent outperformance of Random Forest models.

## III. METHODOLOGY

Along with the names and labels of the characteristics, this section provides dataset descriptions. Next, we chose machine learning methods to forecast software bugs. Figure 1 shows the metrics used to assess the performance of the machine learning model. These metrics include F1-score, recall, accuracy, precision, and root-mean-squared error (RMSE).I. MatrixThe JM1 dataset, created for the express purpose of software failure prediction, is made accessible via the NASA Metrics Data Program [8]. Static code metrics including branch count, lines of code metrics, and McCabe's and Halstead's measures are part of the collection, which has 10,885 instances and 22 characteristics [5, 6]. A "C"-developed real-time predictive ground system, JM1 uses simulations to provide forecasts. Researchers have used this dataset to study the performance of various learners in defect prediction, drawing attention to the significance of false alarm rates, detection probabilities, and the impact of effort on

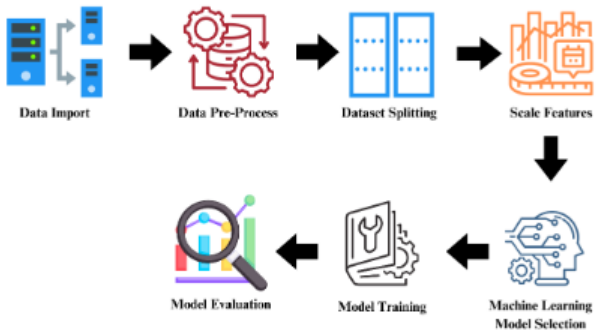


**Fig. 1. Methodology Workflow**

forecasting measures [6]. You can see the order of JM1's attributes in Table I. This study employed a two-part method to evaluate the model's efficacy: a training set included 80% of the samples, while a testing set had 20%.

TABLE I
ATTRIBUTES OF JM1 DATASET [8]

| Attribute | Labels |
|---|---|
| loc | McCabe's line count of code |
| v(g) | McCabe "cyclomatic complexity" |
| ev(g) | McCabe "essential complexity" |
| iv(g) | McCabe "design complexity" |
| n | Halstead total operators + operands |
| v | Halstead "volume" |
| l | Halstead "program length" |
| d | Halstead "difficulty" |
| i | Halstead "intelligence" |
| e | Halstead "effort" |
| b | Halstead |
| t | Halstead's time estimator |
| lOCode | Halstead's line count |
| lOComment | Halstead's count of lines of comments |
| lOBlank | Halstead's count of blank lines |
| lOCodeAndComment | Numeric |
| uniq_Op | Unique operators |
| uniq_Opnd | Unique operands |
| total_Op | Total operators |
| total_Opnd | Total operands |
| branchCount | Percentage of the flow graph |
| defects | {false, true} - Module has/has not reported defects |

Utilized Algorithms for Machine LearningPreventing software failures early in the software development life cycle (SDLC) is possible with the use of machine learning technologies [21]. These approaches identify hidden patterns in historical software data.That is because it is feature-dependent on the dataset. It is difficult to choose the optimal method to use for fault prediction [20]. Naïve Bayes, Decision Trees, Random Forest, Support Vector Machine, Artificial Neural Networks, Logistic Regression, and K-Nearest Neighbors are six popular supervised machine learning algorithms that this paper intends to analyze extensively. The purpose of this research is to assess the performance accuracy and predictive capacity of different ML algorithms as they pertain to software bug prediction. This research provides a detailed comparison of the selected ML algorithms, outlining their respective benefits and drawbacks.Bayes' Naive (NB) method: The simplicity and efficacy of Naïve Bayes make it stand out as a probabilistic classifier. The Bayes theorem is its foundation, and it operates on the premise of independence between characteristics. The acronym NB refers to a family of algorithms that share the underlying idea that there is no relationship between the presence or absence of one characteristic and any other feature [1]. cited as [18]. Pattern recognition and machine learning are only two of the numerous fields that regularly make use of decision

trees (DTs), which are robust models. Hierarchical tests compare numerical properties at each node to a threshold value in a sequential fashion [22].An ANN, or artificial neural network, is; Modeled after biological brain networks, ANNs function as massively parallel systems. Networks of artificial neurons, or ANNs, calculate weighted sums of inputs and produce outputs that are sent back into the network. One or more outputs will be generated as a result of this procedure. Artificial neural networks (ANNs) learn complex patterns from labelled data, which is useful for bug prediction [7], [15]. Various ANN designs will provide varying results when applied to certain challenges [23]. The Random Forest ensemble learning method builds a number of decision trees during the training process. It makes accurate predictions by combining the work of many tree-structured classifiers. Random forest determines a variable's significance by shuffled (or permuted) it at random [25]. Random Forest's ensemble feature enhances both the accuracy of predictions and their generalizability. [2] [5].Support Vector Machine (SVM): SVM is a supervised machine learning model that is often used in binary classification applications. Because it may build a decision boundary between classes to attain maximum margins, it works very well with datasets that have minimal data [11]. One neighbor-dependent classification method is K-Nearest Neighbors (KNN). A KNN classifier is defined by measuring the proximity of vectors that represent converted open-source code according to parameters like design complexity using a distance metric. Finding patterns in datasets is a good fit for KNN [16].Logistic Regression: This statistical method is used to model the connection between the dependent variable and the independent factors. Multinomial logistic regression is an extension of logistic regression that can deal with variables that have more than one class [17].Criteria for Assessment (C.)The target variable is subjected to label encoding during data preparation. In addition, the one-hot approach is used to encode categorical characteristics. After that, we standardize the feature set by feature scaling, and after that, we delete the features from the target variable.The research employs a range of machine-learning approaches to forecast software faults. Some examples of these models are Naïve Bayes, Decision Tree, Artificial Neural Network (ANN), K-Nearest Neighbors (KNN), Random Forest, Logistic Regression, and Support Vector Machine (SVM). The training dataset is used to train the models, and then the test set is used to generate predictions. This study assessed the efficacy of ML systems for software defect prediction using a standard set of metrics trained on the produced confusion matrices. Here are the evaluation tools and confusion matrix that were utilized:The first tool is a confusion matrix, which is used to evaluate how well the machine learning models are doing. If you want to know how well a categorization algorithm did, this is the way to go [24]. All four types of false positives, true negatives, and true positives are examined in depth in this matrix. Using the confusion matrix as a starting point, several assessment measures are constructed [1].2) Accuracy: The accuracy of a classification model is a measure of its overall correctness. Using Equation 1 [1], [7], we may calculate the overall proportion of properly identified cases, including True Positives and True Negatives.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

3)Recision (sometimes called positive predictive value) measures how accurate a model is in predicting the future. Equation 2 shows the overall number of predicted positive instances, which includes both true and false positives. To get this ratio, the number of precisely expected positive cases, or True Positives, is divided by this total [1], [19].

$$Precision = \frac{TP}{TP + FP}$$

4) Recall: The True Positive Rate, sometimes called Recall, evaluates a model's ability to identify positive instances, as shown in Equation 3. The ratio of all positive instances that were accurately predicted (True Positives) to all positive cases that actually occurred (True Positives + False Negatives) is known as the false positive rate [1], [7].

$$Recall = \frac{TP}{TP + FN}$$

5) F-measure: The F-measure is the harmonic mean of recall and accuracy. To find a happy medium between recall and accuracy, Equation 4 provides a single measure to compare models. Taking into account both false positives and false negatives is vital when using the F-measure, which might be useful when there is an uneven distribution of classes [1], [7].

$$F\text{-}measure = \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}}$$

The average size of the errors between the anticipated and actual values in Equation 5 is often calculated by regression tasks using Root-Mean-Square Error (RMSE).Serious errors are penalized more heavily than smaller ones.Use root-mean-squared error (RMSE) for regression problems where predicting the actual numerical value is important and error size is a concern [1].

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(X_i - XP_i)^2}{n}}$$

## IV. EXPERIMENTAL RESULTS

Part A: How Well Machine Learning Models WorkUsing Python's machine learning capabilities, this research evaluated six different machine learning algorithms for software bug prediction. The results for both the training and testing sets are shown in Table II. Among the tested models, the Random Forest model achieved an accuracy of 81%, while the Logistic Regression model achieved an accuracy of 80%, placing it in second place. The models were more effective than Naïve Bayes, ANN, Decision Tree, SVM, and K-Nearest Neighbors.

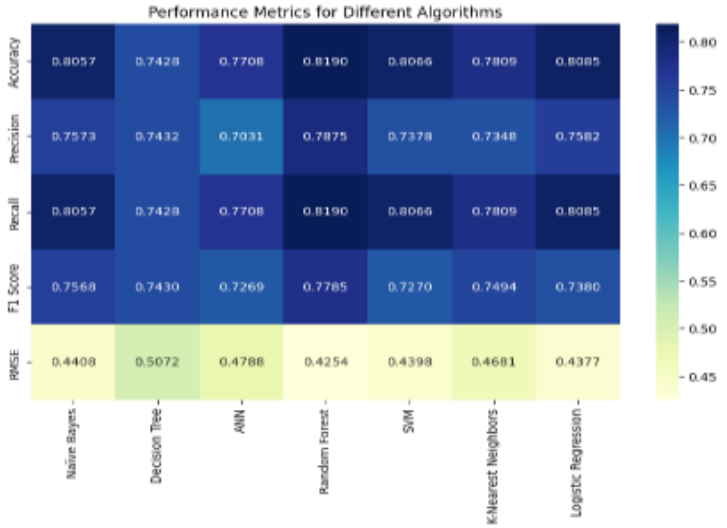


**Fig. 2. Performance Metrics for Different Algorithms on Testing Data.**

The Random Forest model then demonstrated the most accuracy in forecasting positive cases, with a precision of 79%, making it the top performer in terms of precision. With a 75% accuracy rate, the Logistic Regression model was also very well. Recall, the second assessment matrix, quantifies the percentage of true positives that the models properly detected in Figure 2. With a recall of 81%, the Random Forest model outperformed the others, showing that it successfully detected a large number of real positive events.We also used the F-measure to compare the ML models. The top F1 score was achieved by Random Forest. Thus, Random Forest and Logistic Regression seem to provide the optimal balance of accuracy and recall, according to the F1 scores. Lastly, Random Forest stands out as the best performer in analyzing machine learning algorithms using RMSE. Its lowest score of 0.4254 indicates higher accuracy in predicting outcomes.The RMSE for Logistic Regression is 0.4378, indicating good performance as well. A somewhat larger average divergence between anticipated and actual values is shown by other models with somewhat higher RMSE values, which range from 0.4408 to 0.5072.B. Talking About ItA comparison study was conducted taking into account the performance of Random Forest, Logistic Regression, SVM, and Naïve Bayes.

TABLE II
MODEL PERFORMANCE ON TRAINING AND TESTING SETS

| Model | Testing Results | | | | | Training Results | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Prec. | Rec. | F1 | Acc. | RMSE | Prec. | Rec. | F1 | Acc. | RMSE |
| Naïve Bayes | 0.757 | 0.806 | 0.757 | 0.805 | 0.441 | 0.757 | 0.805 | 0.757 | 0.805 | 0.442 |
| Decision Tree | 0.743 | 0.743 | 0.743 | 0.743 | 0.507 | 0.989 | 0.989 | 0.989 | 0.989 | 0.103 |
| ANN | 0.703 | 0.771 | 0.727 | 0.771 | 0.479 | 0.714 | 0.777 | 0.734 | 0.777 | 0.472 |
| Random Forest | 0.788 | 0.819 | 0.778 | 0.819 | 0.425 | 0.989 | 0.989 | 0.989 | 0.989 | 0.103 |
| SVM | 0.738 | 0.807 | 0.727 | 0.807 | 0.440 | 0.784 | 0.809 | 0.730 | 0.809 | 0.437 |
| k-NN | 0.735 | 0.781 | 0.749 | 0.781 | 0.468 | 0.821 | 0.839 | 0.818 | 0.839 | 0.401 |
| Logistic Reg. | 0.758 | 0.808 | 0.738 | 0.808 | 0.438 | 0.766 | 0.809 | 0.742 | 0.809 | 0.437 |

TABLE III
COMPARISON OF OUR AND PREVIOUS RESEARCH ACCURACY RESULTS

| Method | Accuracy Achieved (%) | Previous Accuracy (%) |
|---|---|---|
| Random Forest | 82.00 | 77.96 [26] |
| Logistic Regression | 81.00 | 80.94 [27] |
| SVM | 80.70 | 80.66 [27] |
| Naïve Bayes | 80.50 | 80.30 [26] |

consistent with previous research on the JM1 dataset, as shown in Table III, promising accuracy outcomes. The authors of the research [26] analyzed the JM1 dataset using a number of machine learning programs, including SAS, WEKA, See5, and ROCKY. The Naïve Bayes model achieved an accuracy of 80.30 percent, whereas the Random Forest model averaged 77.96 percent. When we apply sophisticated machine learning methods to the same dataset, however, we find that the accuracy increases. The authors of the independent study reported in [27] used the JM1 dataset to perform feature selection; they found that SVM achieved an accuracy of 80.66% and Logistic Regression an accuracy of 80.94%.Our study makes use of a one-of-a-kind dataset and state-of-the-art models, in contrast to what we observed in section II, where various authors used diverse datasets and approaches [1] [2] [11]. It is worth mentioning that our dataset and models continuously show better accuracy, which highlights how successful our technique is compared to previous approaches.Random Forest's ensemble nature, versatility, resilience to overfitting and outliers, feature significance analysis, and performance on the provided dataset all contribute to its strong performance in software bug prediction.

## V. CONCLUSION AND FUTURE WORK

This research looked at software bug prediction using ML techniques and the JM1 dataset. The findings demonstrated that various methods were successful; the Random Forest model stood out with the highest accuracy (81%), precision (78.55%), recall (81%), F-measure, and minimum RMSE (0.4254). This is where Logistic Regression really shone. To gauge their impact on model performance, future studies may look into sophisticated feature engineering techniques. Models may be enhanced by including domain-specific knowledge and contextual data, which might lead to even more accurate predictions.

## VI. REFERENCES

[1] Hammouri, A., Hammad, M., Alnabhan, M., & Alsarayrah, F. (2018,January). Software Bug Prediction using Machine Learning Approach.International Journal of Advanced Computer Science and Applications,9.

[2] S. Delphine Immaculate, M. Farida Begam and M. Floramary, ”Soft-ware Bug Prediction Using Supervised Machine Learning Algorithms,”2019 International Conference on Data Science and Communication(IconDSC), Bangalore, India, 2019, pp. 1-7.

[3] D. Sharma, P. Chandra, ”Software Fault Prediction Using Machine-Learning Techniques,” in Smart Computing and Informatics, GuruGobind Singh Indraprastha University, 2018, pp. 541-549.

[4] K. Tanaka, A. Monden and Z. Y¨ucel, ”Prediction of Software Defects Using Automated Machine Learning,” 2019 20th IEEE/ACIS Interna-tional Conference on Software Engineering, Artificial Intelligence, Net-working and Parallel/Distributed Computing (SNPD), Toyama, Japan,2019, pp. 490-494.

[5] X. Zhao, W. Zhang and N. Ai, ”Analysis of Attribute Selection Method Based on IDOE in Software Fault Prediction,” 2019 Interna-tional Conference on Intelligent Computing, Automation and Systems(ICICAS), Chongqing, China, 2019, pp. 741-745, doi: 10.1109/ICI-CAS48597.2019.00160.

[6] Z. B. Guven Aydin and R. Samli, ”Performance Evaluation of Some Machine Learning Algorithms in NASA Defect Prediction Data Sets,”2020 5th International Conference on Computer Science and Engineer-ing (UBMK), Diyarbakir, Turkey, 2020, pp. 1-3.

[7] R. P and P. Kambli, ”Predicting Bug in a Software using ANN Based Machine Learning Techniques,” 2020 IEEE International Conference for Innovation in Technology (INOCON), Bangluru, India, 2020, pp. 1-5.

[8] PROMISE repository, http://promise.site.uottawa.ca/SERepository/datasets/jm1.arff (accessed Dec. 1, 2023).

[9] C. L. Prabha and N. Shivakumar, ”Software Defect Prediction Using Machine Learning Techniques,” 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), Tirunelveli, India, 2020, pp. 728-733.

[10] Manjula, C. M., Prasad, Lilly Florence, & Arya, Arti. (2015). A Study on Software Metrics based Software Defect Prediction using Data Mining and Machine Learning Techniques. International Journal of Database Theory and Application, 8(3), 179-190.

[11] A. Khalid, G. Badshah, N. Ayub, M. Shiraz, and M. Ghouse, ”Software Defect Prediction Analysis Using Machine Learning Techniques,” Sus- tainability, vol. 15, pp. 5517, Mar. 21, 2023. DOI: 10.3390/su15065517.

[12] M. D’Ambros, M. Lanza and R. Robbes, ”An extensive comparison of bug prediction approaches,” 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010), Cape Town, South Africa, 2010, pp. 31-41.

[13] R. Malhotra, A Systematic Review of Machine Learning Techniques forSoftware Fault Prediction, Applied Soft Computing Journal (2014). [14] Jijo, B. T., & Abdulazeez, A. M. (2021). ”Classification Based on Decision Tree Algorithm for Machine Learning.” Journal of Applied Science and Technology Trends, 2(1), 20-28.

[15] I. F. B. Tronto, J. D. S. da Silva, and N. Sant’Anna, ”An investigation of artificial neural networks based prediction systems in software project management,” Journal of Systems and Software, vol. 81, no. 3, pp. 356-367, March 2008.

[16] M. Hammad, A. Alqaddoumi, H. Al-Obaidy, and K. Almseidein, ”Pre-dicting Software Faults Based on K-Nearest Neighbors Classification,”Int. J. Com. Dig. Sys., vol. 8, no. 5, Sep. 2019.

[17] M. A. Ibraigheeth and S. A. Fadzli, ”Software project failures predictionusing logistic regression modeling,” 2020 2nd International Conferenceon Computer and Information Sciences (ICCIS), Sakaka, Saudi Arabia,2020, pp. 1-5.

[18] O. F. Arar and K. Ayan, ”A feature-dependent Naive Bayes approachand its application to the software defect prediction problem,” AppliedSoft Computing, vol. 59, pp. 197-209, Oct. 2017.

[19] M. Efendioglu, A. Sen and Y. Koroglu, ”Bug Prediction of SystemCModels Using Machine Learning,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 38, no. 3, pp.419-429, March 2019.

[20] I. Mehmood et al., ”A Novel Approach to Improve Software Defect Prediction Accuracy Using Machine Learning,” in IEEE Access, vol.11, pp. 63579-63597, 2023.

[21] Bilal Khan, Rashid Naseem, Muhammad Arif Shah, Karzan Wakil, Atif Khan, M. Irfan Uddin, Marwan Mahmoud, ”Software Defect Prediction for Healthcare Big Data: An Empirical Evaluation of Machine Learning Techniques”, Journal of Healthcare Engineering, vol. 2021, Article ID 8899263, 16 pages, 2021.

[22] S. Fletcher and M. Z. Islam, ”Decision Tree Classification with Dif- ferential Privacy: A Survey,” ACM Computing Surveys, vol. 52, no. 4, Article 83, pp. 1–33, August 2019.

[23] M. Saritas and A. Yasar, ”Performance Analysis of ANN and Naïve Bayes Classification Algorithm for Data Classification,” International Journal of Intelligent Systems and Applications, vol. 7, 2019.

[24] N. Ali, D. Neagu, and P. Trundle, ”Evaluation of k-nearest neighbor classifier performance for heterogeneous data sets,” SN Applied Sci-ences, vol. 1, 2019.

[25] L. He, R. A. Levine, J. Fan, J. Beemer, and J. Stronach, ”Random forest as a predictive analytics alternative to regression in institutional research,” Practical Assessment, Research and Evaluation, vol. 23, pp. 1-16, 2018.

[26] L. Guo, Y. Ma, B. Cukic and Harshinder Singh, ”Robust prediction of fault-proneness by random forests,” 15th International Symposium on Software Reliability Engineering, Saint-Malo, France, 2004, pp. 417-428.

[27] Azam, Muhammad & Nouman, Muhammad & Gill, Ahsan, ”Comparative Analysis of Machine Learning Techniques to Improve Software Defect Prediction”, vol 5, pp 41-66, 2022