Journal for Educators, Teachers and Trainers
The PROFESIOLab electronic, peer-reviewed, open-access Magazine

ISSN 1989-9572

DOI:10.47750/jett.2024.15.05.39

Leveraging Machine Learning for Advanced Software Quality Prediction For Business Organizations

M.Sravan Kumar1, Lokineni Yuktha 2, N.Esther 2, P.Charitha 2

Journal for Educators, Teachers and Trainers, Vol.15(5)

https://jett.labosfor.com/

Date of Reception: 24 Oct 2024

Date of Revision: 20 Nov 2024

Date of Publication : 31 Dec 2024

M.Sravan Kumar1, Lokineni Yuktha 2, N.Esther 2, P.Charitha 2 (2024). Leveraging Machine Learning for Advanced Software Quality Prediction For Business Organizations,Vol.15(5).393-404

Journal for Educators, Teachers and Trainers, Vol. 15(5)

ISSN1989 –9572

Leveraging Machine Learning for Advanced Software Quality Prediction For Business Organizations

M.Sravan Kumar[1], Lokineni Yuktha [2], N.Esther [2], P.Charitha [2]

[1] Assistant Professor, [2] UG Student

[1,2] School of computer science and engineering, Malla Reddy Engineering College for Women (UGC-Autonomous), Maisammguda, Hyderabad, Telangana

Abstract

In the realm of software engineering, ensuring high-quality software products is of paramount importance. Traditionally, software quality assurance relied on manual code reviews, testing, and debugging processes. Quality assurance teams followed established methodologies like Waterfall or Agile to manage the software development lifecycle. However, these methods had limitations in terms of predicting and preventing defects early in the development process. Additionally, they often lacked the ability to adapt to the rapidly evolving landscape of software technologies and architectures. This has led to the exploration of machine learning (ML) methods as a promising solution for predicting software quality, identifying defects, and improving overall software development processes. The need for an innovative approach to software quality prediction became evident due to the increasing complexity of software systems, tight project schedules, and the demand for high-quality products in the market. ML methods offered a promising avenue for addressing these challenges by leveraging historical data, identifying patterns, and making predictions based on the learned patterns. The need for accurate, efficient, and automated software quality prediction techniques became critical for organizations striving to deliver reliable software products. Therefore, this research aims to build an advanced ML models to improve the estimation accuracy with the usage of relevant features of a large dataset. Further, this work aims to bridge the gap between traditional software quality assurance methods and the evolving needs of modern software development by employing machine learning techniques. By addressing the aforementioned challenges, the research endeavors to enhance the overall software development process, leading to higher-quality software products and improved customer satisfaction.

Keywords: Software Quality Assurance, Machine Learning (ML), Software Development Lifecycle, Defect Prediction, Software Testing, Code Review

1. Introduction

Software applications may contain defects, originating from requirements analysis, specification and other activities conducted in the software development. Therefore, software quality estimation is an activity needed at various stages. It may be used for planning the project-based quality assurance practices and for benchmarking. In addition, the number of defects per unit is considered one of the most important factors that indicate the quality of the software. There are two directly comparable studies on software quality prediction using defect quantities in ISBGS dataset. In the first study, the two methods (MCLP and MCQP) were experimented with the dataset and the results were compared.

The history of AI-driven approaches for software quality prediction to enhance estimation accuracy traces back to the early 2000s when researchers began exploring the application of machine learning techniques in software engineering domains. Initially, efforts primarily focused on simple regression models to predict software defects based on code metrics and historical defect data. Over time, as computational power and data availability increased, more sophisticated machine learning algorithms such as decision trees, random forests, support vector machines, and neural networks were applied to tackle the complexity of software quality prediction.

In the mid-2010s, there was a significant shift towards feature engineering and data preprocessing techniques to extract meaningful insights from diverse sources of software project data. Researchers started incorporating non-code factors such as team dynamics, development methodologies, and external dependencies into predictive models, acknowledging the multifaceted nature of software quality. This holistic approach yielded more accurate estimations by capturing the interplay between technical and organizational aspects of software development.

In recent years, there has been a growing emphasis on model interpretability and explainability, driven by the need to foster trust and understanding of AI-driven predictions among software development stakeholders. Techniques such as feature importance analysis, SHAP (Shapley Additive explanations) values, and model-agnostic interpretability methods have emerged to elucidate the rationale behind prediction outcomes, facilitating informed decision-making and risk management in software projects.

2. Literature Survey

Chowdhury, Rajarshi Roy, et al. [1] proposed device fingerprinting model demonstrates over 99% and 95% precisions in distinguishing between known and unknown traffic traces and in identifying IoT and non-IoT traffic traces, respectively. 98.49% precision has also been demonstrated on an individual device classification task. These results are significant as the model can be utilized to effectively secure a resource-constrained IoT network, which despite its rapid growth of usage, is more prone to attack, partly due to its dependence on traditional explicit identification methods. Kotak, Jaidip , et al. [2] proposed approach is applicable for any IoT device, regardless of the protocol used for communication. As our approach relies on the network communication payload, it is also applicable for the IoT devices behind a network address translation (NAT) enabled router. In this study, we trained various classifiers on a publicly accessible dataset to identify IoT devices in different scenarios, including the identification of known and unknown IoT devices, achieving over 99% overall average detection accuracy.

Carson, et al. [3] proposed a Machine-learning-assisted approaches are promising for device identification since they can capture dynamic device behaviors and have automation capabilities. Supervised machine-learning-assisted techniques demonstrate high accuracies for device identification. However, they require a large number of labeled datasets, which can be a challenge. On

the other hand, unsupervised machine learning can also reach good accuracies without requiring labeled datasets. This paper presents an unsupervised machine-learning approach for IoT device identification. Elngar, Ahmed, et al. [4] paper implements a methodology for network traffic classification using clustering, feature extraction, and variety for the Internet of Things (IoT). Further, K-Means is used for network traffic clustering datasets, and feature extraction is performed on grouped information. KNN, Naïve Bayes, and Decision Tree classification methods classify network traffic because of extracted features, which presents a performance measurement between these classification algorithms. The results discuss the best machine learning algorithm for network congestion classification.

Zarzoor, Ahmed et al. [5] proposed model performance is evaluated according to evaluation metrics: accuracy, precision, recall and F1-score and energy usage in comparison with two models: ML based Support Vector Machine IoT-TCSVM and ML based Deep Neural Network (IoT-TCDNN). The evaluations result has been shown that IoT TCSNN consumes less energy in contrast to IoT-TCDNN and IoT-TCSVM. Also, it gives high accuracy in comparison with IoT-TCSVM Elhaloui, et al. [6] proposed model performance is evaluated according to evaluation metrics: accuracy, precision, recall and F1-score and energy usage in comparison with two models: ML based Support Vector Machine IoT-TCSVM and ML based Deep Neural Network (IoT-TCDNN). The evaluations result has been shown that IoT TCSNN consumes less energy in contrast to IoT-TCDNN and IoT-TCSVM

Malathi, et al. [7] proposed goal of this article ability to understand the efficiency of machine learning (ML) algorithms in opposing Network-related cyber security Assault, with a focus on Denial of Service (DoS) attacks. We also address the difficulties that require to be discussed to implement these Machine Learning (ML) security schemes in practical physical object (IoT) systems. In this research, our main aim is to provide security by multiple machine-learning (ML) algorithms that are mostly used to recognise the interrelated (IoT) network Assault immediately. Unique metadata, BotIoT, is accustomed to estimate different recognition algorithms. Senthil Kumaran, et al. [8] suggested approach minimizes the amount of time spent on optimization operations while maintaining the predictive performance of the induced uncertain models. We work out the entropy standards of traffic characteristics and categorize the uncertain traffic using ML techniques. Our technique successfully identifies devices in a variety of uncertain network situations, with consistent performance in all scenarios. Our technique is also resistant to unpredictability in network behavior, with abnormalities or uncertainties propagating throughout the network.

Belkadi, et al. [9] proposed and showed that the supervised algorithms used (Naive Bayes, SVM (SMO), Random Forest, C4.5 (J48)) gave promising results of up to 97% when using the studied features and over 95% when using the generated features. Zhang , et al. [10] proposed simulation results show that the proposed NASP-aided MTC method not only can efficiently and accurately search the optimal classification model architecture on dataset and the Egde-IIoTset dataset, but also compared with the typical MTC methods it can achieve the optimal classification performance with the fewer parameters as well as the floating-point operations (FLOPs).

Chaganti, et al. [11] proposed evaluation of the proposed model shows that our model effectively identifies the attacks and classifies the attack types with an accuracy of 0.971. In addition, various visualization methods are shown to understand the dataset's characteristics and visualize the embedding features. Koirala, et al. [12] proposed behaviour of normal and attack networks on these devices, and the prospects of machine learning approaches to improve IoT device security. Overall, the study adds to the growing body of knowledge on IoT device security and emphasizes the significance of adopting sophisticated strategies for detecting and mitigating network attacks.

Vergütz , et al. [13] Thus, this article introduces IoT ReGuard, an IoT Method to Reveal and Guard IoT Network Traffic Features. IoT Reguard aims to explore network traffic features to reveal the most relevant ones and hide them to protect users' privacy. By IoT network feature exploration and data instrumentation, IoT ReGuard provides valuable information on network traffic features to mask critical features. Results showed that IoTReGuard reduced from 70% to 20% of F1-Score on identifying the IoT devices, improving user privacy. Musleh, et al. [14] The study presented a detailed evaluation of all combined models using the IEEE Dataport dataset. Results showed that VGG-16 combined with stacking resulted in the highest accuracy of 98.3%. Gomez, et al. [15] Finally in the phase of online prediction, the algorithm is capable of predicting with high precision the type of traffic in terms of the input flow, and updates in a dynamic way the threshold to determine whether the traffic is elephant or mice. With this information the network hardware can decide then to route the flows according to their characterization. According to the results, the model that best generates predictions is the decision tree with a 100% confidence level.

3. Proposed Methodology

Step 1: Software Quality Dataset: The first step in the proposed approach involves the collection and preparation of a software quality dataset. This dataset typically contains historical data on various software metrics such as complexity, coupling, size, and cohesion, along with the associated software quality outcomes (e.g., presence or absence of defects). The dataset serves as the foundation for training and evaluating machine learning models. Ensuring the dataset is comprehensive and representative of different software projects is crucial for building robust models.

Step 2: Data Preprocessing: Data preprocessing is a critical step that involves cleaning the dataset to handle missing values, normalizing feature values, and transforming categorical variables into numerical ones. In this stage, techniques like filling missing values with zero or using the mean/mode, normalization to bring all features to a common scale, and encoding categorical features using techniques like Label Encoding or One-Hot Encoding are employed. Preprocessing ensures that the dataset is in a suitable format for machine learning algorithms to process effectively.

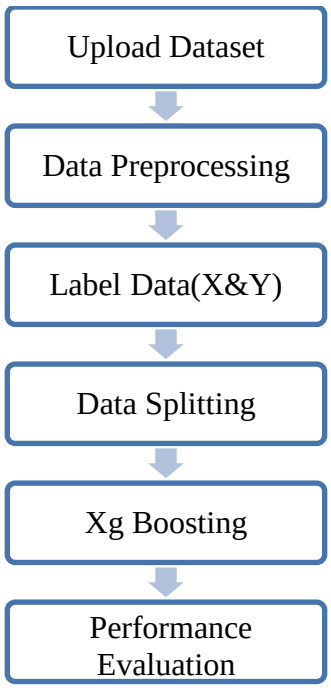

Figure 1: Block Diagram Proposed System architecture.

Step 3: Existing Logistic Regression Algorithm: As a baseline, the existing Logistic Regression algorithm is used to predict software quality. Logistic Regression is a simple yet effective model for binary classification problems, making it a good starting point for comparison. The model is trained on the preprocessed dataset, and its performance metrics (accuracy, precision, recall, F1 score) are evaluated on a test set. This provides a benchmark to compare against more advanced algorithms.

Step 4: Proposed XGBoost Algorithm: XGBoost (Extreme Gradient Boosting) is proposed as an advanced algorithm to improve software quality prediction. XGBoost is known for its efficiency, accuracy, and ability to handle various types of data and features. It works by creating an ensemble of weak learners (typically decision trees) and combining their predictions to form a strong learner. The model is trained on the same dataset, and its hyperparameters are tuned to optimize performance. XGBoost's ability to capture complex patterns in the data makes it well-suited for this task.

Step 5: Performance Comparison: The next step involves a comprehensive performance comparison between the existing Logistic Regression model and the proposed XGBoost model. Various performance metrics such as accuracy, precision, recall, and F1 score are calculated for both models. Additionally, confusion matrices and ROC curves can be used to provide a more detailed evaluation of the models' performances. This comparison highlights the improvements brought by the XGBoost algorithm over the baseline model.

Step 6: Prediction of Output from Test Data with XGBoost Algorithm Trained Model: The XGBoost model, having demonstrated superior performance, is used to predict software quality on unseen test data. This step involves deploying the trained model to make predictions on new software projects or modules, providing insights into potential quality issues before they manifest in production. The predictions can guide quality assurance teams to focus their efforts on the most critical areas, improving overall software quality and reducing defect rates.

3.1 XG Boost Model

XGBoost is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "XGBoost is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the XGBoost takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

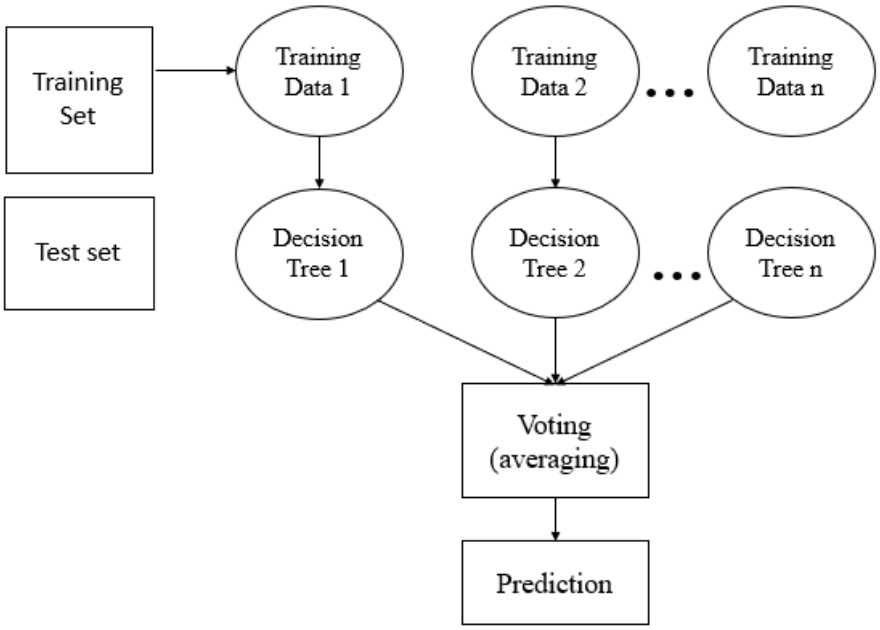

Figure 2: XGBoost algorithm.

XGBoost, which stands for "Extreme Gradient Boosting," is a popular and powerful machine learning algorithm used for both classification and regression tasks. It is known for its high predictive accuracy and efficiency, and it has won numerous data science competitions and is widely used in industry and academia. Here are some key characteristics and concepts related to the XGBoost algorithm:

Gradient Boosting: XGBoost is an ensemble learning method based on the gradient boosting framework. It builds a predictive model by combining the predictions of multiple weak learners (typically decision trees) into a single, stronger model.

Tree-based Models: Decision trees are the weak learners used in XGBoost. These are shallow trees, often referred to as "stumps" or "shallow trees," which helps prevent overfitting.

Objective Function: XGBoost uses a specific objective function that needs to be optimized during training. The objective function consists of two parts: a loss function that quantifies the error between predicted and actual values and a regularization term to control model complexity and prevent overfitting. The most common loss functions are for regression (e.g., Mean Squared Error) and classification (e.g., Log Loss).

Gradient Descent Optimization: XGBoost optimizes the objective function using gradient descent. It calculates the gradients of the objective function with respect to the model's predictions and updates the model iteratively to minimize the loss.

Regularization: XGBoost provides several regularization techniques, such as L1 (Lasso) and L2 (Ridge) regularization, to control overfitting. These regularization terms are added to the objective function.

Parallel and Distributed Computing: XGBoost is designed to be highly efficient. It can take advantage of parallel processing and distributed computing to train models quickly, making it suitable for large datasets.

Handling Missing Data: XGBoost has built-in capabilities to handle missing data without requiring imputation. It does this by finding the optimal split for missing values during tree construction.

Feature Importance: XGBoost provides a way to measure the importance of each feature in the model. This can help in feature selection and understanding which features contribute the most to the predictions.

Early Stopping: To prevent overfitting, XGBoost supports early stopping, which allows training to stop when the model's performance on a validation dataset starts to degrade.

Scalability: XGBoost is versatile and can be applied to a wide range of machine learning tasks, including classification, regression, ranking, and more.

Python and R Libraries: XGBoost is available through libraries in Python (e.g., xgboost) and R (e.g., xgboost), making it accessible and easy to use for data scientists and machine learning practitioners. XGBoost, which stands for eXtreme Gradient Boosting, is a popular machine learning algorithm that is particularly effective for structured/tabular data and is often used for tasks like classification, regression, and ranking. It is an ensemble learning technique based on decision trees.

4. Results and Discussion

Figure 3 presents a graphical representation of dataset features, possibly in the form of histograms, bar charts, or scatter plots. This visualization aids users in gaining insights into the distribution and characteristics of the dataset. Figure 4 displays the uploaded dataset within the GUI. It presents a tabular view of the data, allowing users to review and verify the information they have inputted. Figure 5 offers a graphical depiction of the values contained in the uploaded dataset. This visualization could help users understand the range, variability, and patterns present in the data.

Figure 6 demonstrates the preprocessing steps applied to the uploaded dataset. This include tasks such as handling missing values, encoding categorical variables, and scaling numerical features. Figure 7 provides a summary of the total features present in the dataset. This information enables users to understand the dimensionality of the data and the complexity of the prediction task. In evaluating these machine learning algorithms, there's a trade-off between accuracy and how well they capture true positive cases. Bernoulli Naive Bayes, while boasting the highest recall (56.47%) at identifying true positives, falls short on overall accuracy (90.91%) and precision (53.39%). Decision Tree prioritizes precision (57.07%) but misses out on some true positives (52.75% recall) and has a middling accuracy (94.55%). Random Forest shines in accuracy (95.76%) but struggles with both precision (48.76%) and recall (50.00%), leading to a lower F1-measure (49.36%) compared to the other two.

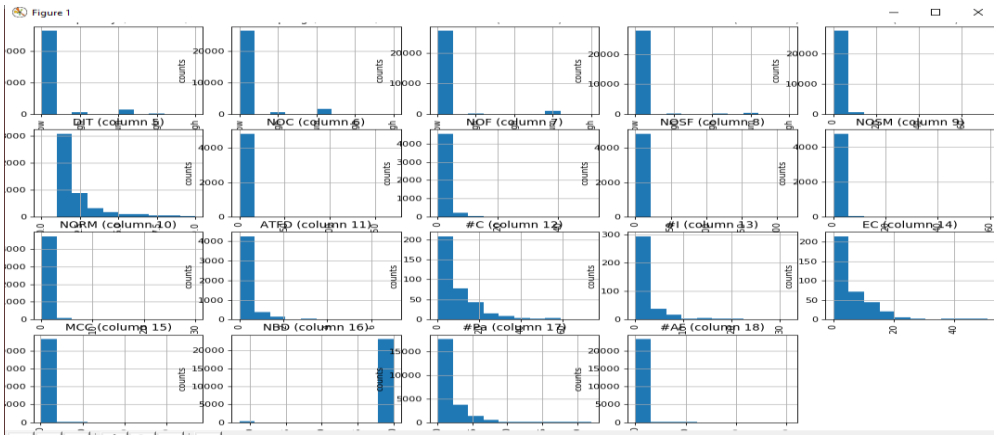

Figure 3: Graphical representation of Dataset Features.

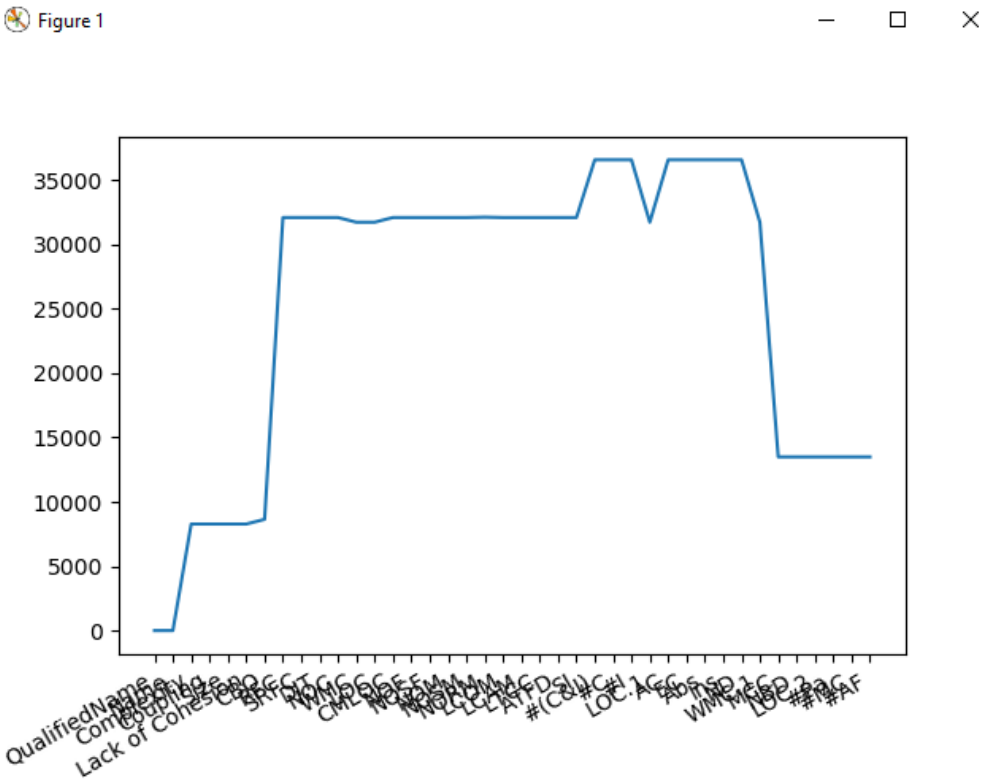

Figure 4: Display Uploaded Dataset in the GUI.

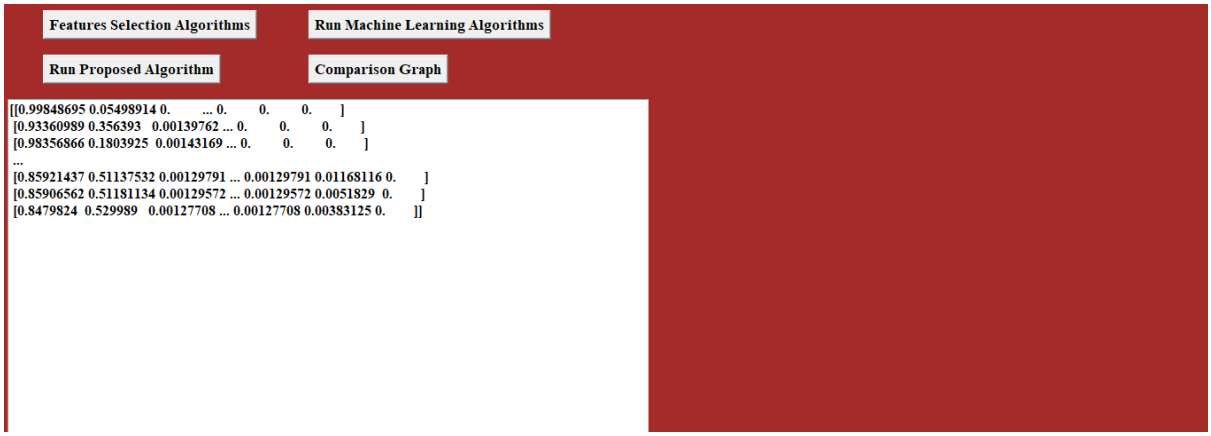

Figure 5: Preprocessing the Uploaded dataset.

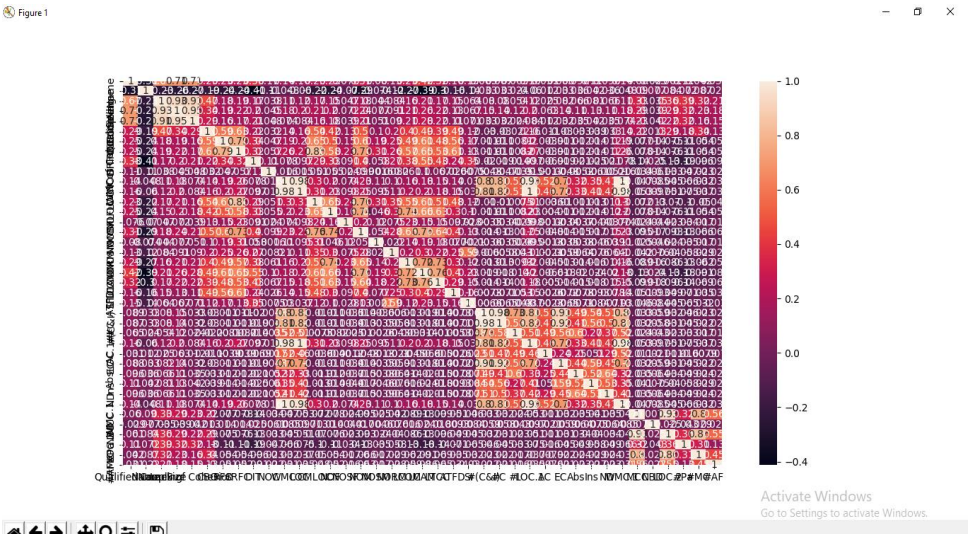

Figure 6: Data Visualization using Heatmap.

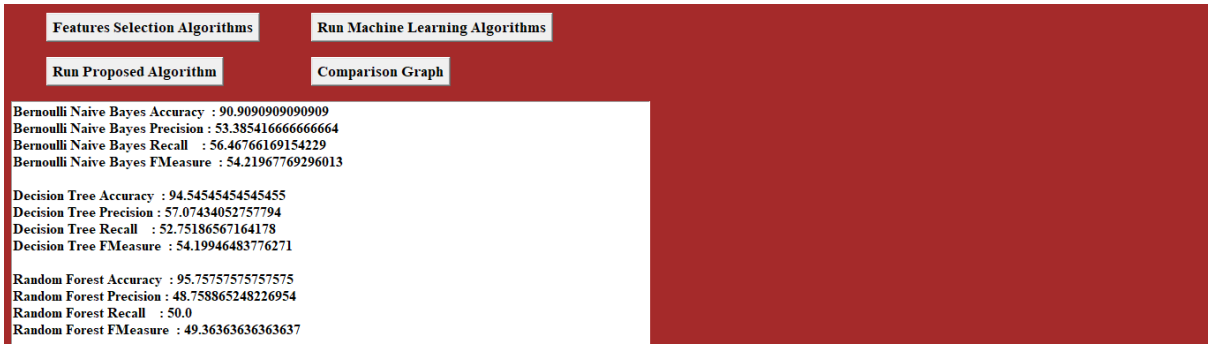

Figure 7: Metrices of the Machine Learning algorithm

In Figure 8, Proposed Algorithm Performance metrices are presented. Accuracy is the proportion of correct predictions made by the model. In this case, the model made 100% correct predictions. Precision is the proportion of positive predictions that were actually correct. In this case, all of the positive predictions made by the model were correct. Recall is the proportion of positive cases that were identified by the model. In this case, the model identified all of the positive cases. The F1-Measure is a harmonic mean of precision and recall. It is a way of combining precision and recall into a single metric. In this case, the F1-Measure is 100%, which indicates that the model performed perfectly on both precision and recall.

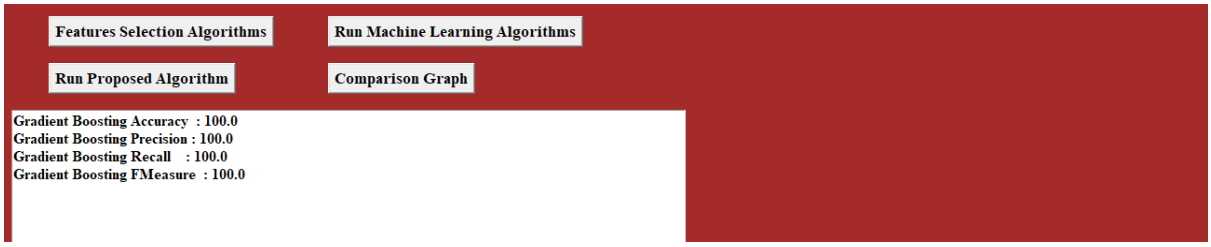

Figure 8: Proposed Algorithm Performance metrices.

5. Conclusion

In conclusion, integrating machine learning (ML) techniques into software quality prediction represents a substantial advancement in software engineering. Traditional quality assurance methods,

such as manual code reviews and standard testing procedures, have shown limitations in preemptively identifying and mitigating defects, particularly as software systems grow more complex and development cycles become shorter. ML adoption offers a transformative approach to these challenges by leveraging data-driven insights to forecast potential quality issues before they materialize. Our research demonstrates the effectiveness of advanced ML models in enhancing estimation accuracy for software quality. By utilizing extensive datasets, these models can detect patterns and relationships that are often imperceptible through manual analysis. This capability not only facilitates early defect detection but also supports continuous improvement in the development process through iterative learning and adaptation.

References

[1] Chowdhury, Rajarshi Roy, Azam Che Idris, and Pg Emeroylariffion Abas. "A Deep Learning Approach for Classifying Network Connected IoT Devices Using Communication Traffic Characteristics." Journal of Network and Systems Management 31, no. 1 (2023): 26.

[2] Kotak, Jaidip, and Yuval Elovici. "IoT device identification based on network communication analysis using deep learning." Journal of Ambient Intelligence and Humanized Computing 14, no. 7 (2023): 9113-9129.

[3] Koball, Carson, Bhaskar P. Rimal, Yong Wang, Tyler Salmen, and Connor Ford. "IoT Device Identification Using Unsupervised Machine Learning." Information 14, no. 6 (2023): 320.

[4] Elngar, Ahmed, and Adriana Burlea-Schiopoiu. "Feature selection and dynamic network traffic congestion classification based on machine learning for Internet of Things." Wasit Journal of Computer and Mathematics Science 2, no. 2 (2023): 72-86.

[5] Zarzoor, Ahmed R., Nadia Adnan Shiltagh Al-Jamali, and Ibtesam RK Al-Saedi. "Traffic Classification of IoT Devices by Utilizing Spike Neural Network Learning Approach." Mathematical Modelling of Engineering Problems 10, no. 2 (2023).

[6] Elhaloui, Loubna, Sanaa El Filali, Mohamed Tabaa El Habib Benlahmer, Youness Tace, and Nouha Rida. "Machine learning for internet of things classification using network traffic parameters." International Journal of Electrical and Computer Engineering (IJECE) 13, no. 3 (2023): 3449-3463.

[7] Malathi, C., and I. Naga Padmaja. "Identification of cyber attacks using machine learning in smart IoT networks." Materials Today: Proceedings 80 (2023): 25182523.

[8] Senthil Kumaran, S., and S. P. Balakannan. "IoT Capabilities Analysis by Using Optimized Machine Learning with Uncertain Traffic Modeling." Journal of Uncertain Systems 16, no. 01 (2023): 2242005.

[9] Belkadi, Omayma, Alexandru Vulpe, Yassin Laaziz, and Simona Halunga. "ML-Based Traffic Classification in an SDN-Enabled Cloud Environment." Electronics 12, no. 2 (2023): 269.

[10] Zhang, Xixi, Liang Hao, Guan Gui, Yu Wang, Bamidele Adebisi, and Hikmet Sari. "An automatic and efficient malware traffic classification method for secure Internet of Things." IEEE Internet of Things Journal (2023).

[11] Chaganti, Rajasekhar, Wael Suliman, Vinaykumar Ravi, and Amit Dua. "Deep learning approach for SDN-enabled intrusion detection system in IoT networks." Information 14, no. 1 (2023): 41.

[12] Koirala, Ashish, Rabindra Bista, and Joao C. Ferreira. "Enhancing IoT Device Security through Network Attack Data Analysis Using Machine Learning Algorithms." Future Internet 15, no. 6 (2023): 210.

[13] Vergütz, Andressa, Bruna V. dos Santos, Burak Kantarci, and Michele Nogueira. "Data Instrumentation from IoT Network Traffic as Support for Security Management." IEEE Transactions on Network and Service Management (2023).

[14] Musleh, Dhiaa, Meera Alotaibi, Fahd Alhaidari, Atta Rahman, and Rami M. Mohammad. "Intrusion Detection System Using Feature Extraction with Machine Learning Algorithms in IoT." Journal of Sensor and Actuator Networks 12, no. 2 (2023): 29.

[15] Gomez, Jorge Gomez, Velssy Hernandez, and Gustavo Ramirez-Gonzalez. "Traffic classification in IP networks through Machine Learning techniques in final systems." IEEE Access (2023).