ISSN: 2321-2152

IJMECE

International Journal of modern electronics and communication engineering

E-Mail
editor.ijmece@gmail.com
editor@ijmece.com

www.ijmece.com

ISSN 2321-2152

www.ijmece.com

Vol 6, Issue 1, 2018

Enhancing Software Testing and Defect Prediction Using Long Short-Term Memory, Robotics, and Cloud Computing

[1]**Basava Ramanjaneyulu Gudivaka**
Cognizant Technology Solutions India
basava.gudivaka537@gmail.com

[2]**Punitha Palanisamy**
SNS College of Technology,
Coimbatore, Tamil Nadu, India.
Punithapalanisamy93@gmail.com

Abstract

Recognizing the importance of software testing as a means of assuring the quality and reliability of the software system, traditional methods mostly rely on manual processes and static forms of testing, which have always been time-consuming and error-prone. This research proposes an integrated framework establishing a synergy between Long Short-Term Memory (LSTM)-based defect prediction, robotics-based test automation, and the cloud, such that defect detection is done scalable and efficient manner. The architecture is such that it trains an LSTM classifier using the Software Defect Prediction Dataset, which contains metrics on the software code and defect histories. The data undergoes pre-processing steps, which help provide high-quality input to the model. The model evaluation is performed, and we obtained the results of 99.78% accuracy, 98.89% precision, 97.56% recall, and 98.96% F1 Score. The framework goes beyond defect prediction and utilizes cloud storage to ensure scalability, resource optimization, and prediction. The cloud infrastructure also provides for efficient utilization of resources, as shown by cloud resource utilization metrics with 80% efficiency, complemented by inference time optimized to 0.15 seconds for each prediction. Integration of robotics further automates the testing process, thus reducing manual intervention and increasing efficiency. This framework can be injected into DevOps pipelines. This allows detection of defects continuously while providing timely feedback during the process of software development. Future work will emphasize harnessing a continuous learning mechanism, increasing multi-cloud friendliness, and optimizing the usage of computational resources to further improve scalability and reduce operation costs.

Keywords: Software Defect Prediction, LSTM, Cloud Computing, Robotics, Machine Learning, Prediction

1. Introduction

Software testing constitutes an integral part of the software development process in which defects are found to guarantee that applications function correctly and meet quality standards [1]. Traditional testing methods often focus on manual inspection and static testing techniques that are time-consuming, error-prone, and non-scalable [2]. As modern software systems continue to become more complex, in recent years, there has arisen a more pressing need for more effective and automated means of testing [3]. Thus, incorporating Artificial Intelligence, Robotics, and Cloud Computing into software testing processes can radically change defect prediction and management [4]. AI-driven models can automate defect detection, robotics can serve to simulate scenarios for testing, while Cloud Computing gives scalability and flexibility when dealing with large volumes of datasets and complex applications [5]. Several penchants introduce the requirement of using increased testing and defect prediction in software. On-demand software systems, specifically those related to IoT, cloud, and AIs, have very advanced testing methodologies. Human error, inadequate coverage of tests, and the speeding up of software development are some causes of bug occurrences in software [6]. The other thing making the continuous and swift form of testing necessary is the very high demand for faster releases for production.

Methods such as artificial testing and static code analysis are becoming insufficient for new-age software development requirements [7]. AI, robotics, and cloud computing aspects may provide a solution in that they will automate the task of testing and thus enable continuous integration and real-time visibility into potential failures [8]. The impact of AI, robotics, and cloud computing on software testing creates realization barriers [9]. The very first leg-up on the progressive challenge is the heavy initial requirement for setup and training of such technologies. This becomes quite a resource-intensive exercise on the datasets upon which AI models must be trained to acquire efficiency [10]. Some robotics requires a huge investment in infrastructure and installation to make effective simulation of real conditions possible. Besides, using a cloud-based testing solution is likely to

involve some security risks, especially when sensitive data could be accessed inappropriately without ensuring proper security measures, because of insufficient isolation of the test data [11]. Over-reliance on the AI-driven testing models may bring about false positives or false negatives owing to the insufficient calibration of the models. In that case, software quality and security may be jeopardized if not adequately contained.

A holistic approach is what one should take to surmount the challenges revolving around the enhancement of AI, robotics, and cloud computing when it comes to software testing and defect prediction. Organizations would need to prepare the workforce involved in AI models and robotic systems by investing in training [12]. The tailor-made need for AI experts, robotics engineers, and software developers would help in collaboration efforts. Cloud computing platforms should have enhanced security protocols that could be about encryption and multi-factor authentication for protection against sensitive data [13]. This may also involve the continuous monitoring and improvement of AI models through feedback loops and performance evaluations to keep the margins of error low and ensure that defect predictions become more realistic.

1.2 Research Objectives

- Integrate LSTM-based models, robotics for automation, and cloud computing for efficient and scalable defect detection in dynamic software environments, thus enhancing software defect prediction as the overall objective of the proposed framework.
- The dataset used in this proposed framework is the Software Defect Prediction Dataset, which consists of multiple code metrics and defect histories that can be used to train and evaluate the defect prediction model.
- A defect prediction using LSTM will be carried out on the dataset to check the software components based on the past data trends.
- Automate the robotic process for testing activities as well as cloud computing in enabling an efficient storage of their time prediction, and scalable deployment of a defect prediction system.

2. Literature review

Optimizing well is a necessity for cloud-based scientific computing in solving problems for large-scale, dynamic workloads and probabilistic decision-making. Hence, a hybrid system comprising Ant Colony Optimizations, gradient descent (GD), and Bayesian decision models will be used for computational efficiency, adaptability, and scalability Ganesan. With the rising dependency on cloud-hosted applications, the cybersecurity risks have increased, therefore, requiring advanced anomaly detection systems [14]. Long Short-Term Memory networks provide efficient analysis of real-time security logs in the detection of unauthorized access, ransomware, and insider threats [15].

Predicting paediatric readmissions has been made easier with hybrid ML models integrated along with cloud-based EMR analytics. Through integrating decision trees, SVMs, and neural networks, the model yields good accuracy and scalability, thus facilitating healthcare decision-making and responsiveness [16]. The AI-Blockchain hybrid model with (SSI) achieves superior accuracy, authentication, and transaction success in its security offerings for IoT by performing real-time threat detection, decentralized authentication, and scalable blockchain transactions when compared to traditional models [17].

With the help of self-attention mechanisms for real-time structured data analysis, the Tab-Transformer-based Intrusion Detection system will improve cloud network security. It has shown an excellent accuracy rate with very low misclassification rates, thus making it a better choice compared to its predecessor models of IDS in a cloud environment [18]. The Hybrid LSTM-Attention Approach improved the disease prediction with more accuracy and efficiency in the cloud-based health system, and it assists in feature selection, minimizes computation costs, and provides for the real-time processing needed to overcome patient monitoring and diagnosis problems [19].

The use of internet-connected devices and cloud technology in healthcare facilitates very real-time monitoring of patients and prediction of risks. This particular system adopts the logic behind Decision Trees rules for performing risk classification, while it models optimizations using Gradient Descent. Therefore, it can be used in improving decision-making during surgery and patient safety [20]. Excellent signal processing coupled with enhanced resource management and better QoS metrics, enabled the hybrid IoT-Fog-Cloud architecture would perform the following functions: system reliability betterment, improved accurate data, improvement of energy efficiency, thus providing scalability and efficiency in patient systems health monitoring [21].

ISSN 2321-2152

www.ijmece.com

Vol 6, Issue 1, 2018

The components of a digital twin enhanced predictive analytics approach, which improves dependability and performance in software through digital twins and prediction modeling, with the context of real-time simulation. It benefits fault detection and resilience of the system with the efficiencies in operations and with performances in execution and dependability compared with the traditional approaches Srinivasan. Deep learning in EHR analytics supports clinical decision-making with predictive information and real-time disease progression modeling. It improves the precision of diagnosis, the ability to detect risks early, and recommendations for tailored therapies, thus improving health outcomes [22].

3. Problem statement

Cloud-Intelligent Systems and Pre-existing Frameworks Encounter several Challenges: one being limited scalability to meet the demands of a dynamic workload [23]. Moreover, most models lack adaptation to changing data patterns and possess security limitations derived from their inherent centralized architectures [24]. Additionally, it provides an adaptive learning component for real-time updates of the model during inference and some efficiency gained in terms of privacy, security, and performance from decentralized techniques such as federated learning [25]. This can be applied to dynamic environments such as healthcare, IoT, and network security.

4. Proposed Software Defect Prediction Using LSTM Methodology

The illustration titled Proposed Framework for Software Defect Prediction Using LSTM shows the scheme for software defects with the help of the LSTM model. The first step is creating the Software Defect Prediction Dataset, which consists of features related to a software code and its defect history. The dataset proceeds for Data Pre-processing, which involves handling missing values and Normalization. After pre-processing, the data is fed into the LSTM Classification model, which classifies the software components.

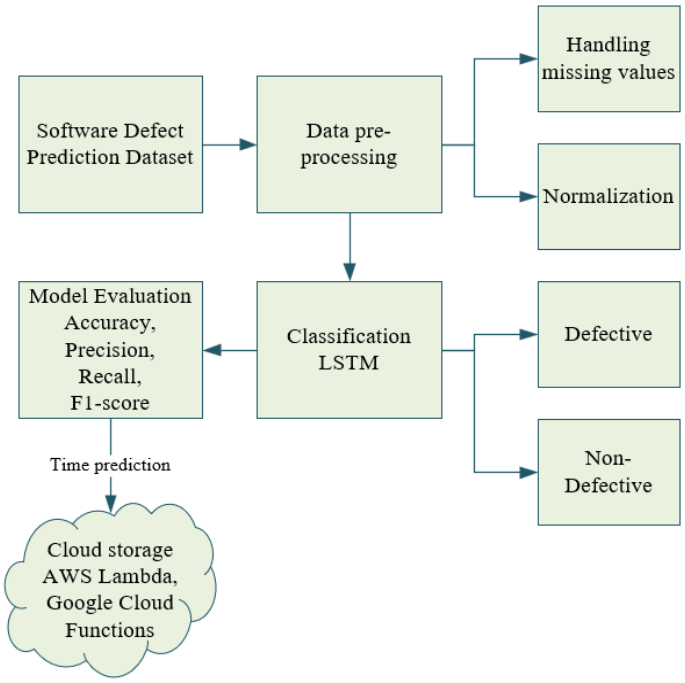

Figure 1: Architecture diagram of proposed methodology

The LSTM-based software defect prediction framework is proposed for the cloud environment, and it is shown in Figure 1. The model performance is assessed based on accuracy, precision, recall, and F1-score. Besides, the system is capable of predicting when the defect occurred. Once the model is trained, results are uploaded to Cloud Storage via platforms such as AWS Lambda and Google Cloud Functions for easy deployment and accessibility for time predictions.

4.1 Dataset description

The dataset utilized in this framework is the Software Defect Prediction Dataset, which relates to the historical information of all the software development projects[26]. It contains attributes that define codes, cyclomatic

ISSN 2321-2152
www.ijmece.com
Vol 6, Issue 1, 2018

complexity, number of functions, and defect status history, as well as project attributes such as size, team size, development timelines, etc. The dependent variable indicates whether a particular component of software is defective (1) or not (0). This dataset bears great importance to train and evaluate an LSTM model intended to predict defects based on code characteristics and previously discovered defect patterns. This dataset is stored in cloud storage holdings so that perceived advantages of scaling and access may be offered for cloud process applications. The feature space is sufficiently rich for accurate modeling of temporal dependencies regarding the occurrence of defects, thus making the dataset relevant for AI-based defect prediction. Pre-processing as well as feature engineering are to be the principal game changers regarding obtaining better accuracy as well as performance of the model with this dataset in defect identification.

4.2 Data preprocessing

Data preprocessing is a vital process that enables the transformation of raw data into a correctly formatted entity for downstream analysis and ML modeling. The steps of this process include managing missing data through imputation or elimination, normalizing or scaling numeric data to ensure that features are in consistent ranges, and encoding categorical variables in numerical formats using algorithms such as Encoding.

Handling Missing Data

The missing values in the dataset are handled using imputation methods to ensure that the model can work with a complete dataset. The formula for calculating the Mean Imputation is given in Eqn (1):

$$\hat{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_j \tag{1}$$

where $\boldsymbol{x_j}$ represents the non-missing values in the feature column.

Normalization

Normalization standardizes the features within a common range, for instance, 0 to 1, so that all features can give their full share of contribution to the model during times of use, such as during computation in an LSTM. The formula for Min-Max Normalization is given in (2):

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{2}$$

where x is the original value and x' is the normalized value.

Standardization

Standardization refers to the conversion of any data attribute into values that will have a mean equal to zero and a variance equal to one. The formula for Standardization is given in (3):

$$x' = \frac{x-\mu}{\sigma} \tag{3}$$

where μ is the mean of the feature and σ is the standard deviation.

4.3 Working of LSTM in Software Defect Prediction

The Long Short-Term Memory is one type of RNN model, which is more specifically applied in sequential data processing for software defect prediction. There are three types of gates: an input gate, a forget gate, and an output gate. The LSTM learns how previous changes in code or previous project milestones play a role in defect occurrences. Because of its capacity to be updated in terms of memory, LSTM responds to all yelling patterns, making itself one of the strongest models in a dynamic software development environment.

Forget Gate: It decides how much of the previous cell state to retain. The formula for calculating the forget gate is given in Eqn (4)

$$f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big) \tag{4}$$

where σ is the sigmoid activation, W_f is the weight matrix, h_{t-1} is the previous hidden state, and x_t is the input at time step t.

Input Gate: It determines how much new information to store in the cell state. The formula for calculating the input gate is given in Eqn (5):

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{5}$$

where i_t is the input gate output.

Output Gate: It decides what part of the cell state should be output to the next layer. The formula for calculating the Forget gate is given in Eqn (6):

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{6}$$

and the final output is shown in Eqn (7):

$$h_t = o_t \cdot \tanh(C_t) \tag{7}$$

where h_t is the output of the LSTM at time step t.

LSTM is a class of RNNs, Recurrent Neural Networks, best suited for sequential data such as software defect prediction. In this case, LSTMs treat sequences of software development data, such as code changes or defect histories, while learning their long-term dependencies. LSTMs solve the traditional RNNs' vanishing gradients through the use of gates controlling the information flow. The input gate controls the new information to be stored, the forget gate determines what past information to keep, and the output gate decides which information to pass on to the next level. The temporal modeling ability of LSTMs enables the prediction of future defects based on historical trends. Based on attributes such as Lines of Code and Cyclomatic Complexity, the LSTM model is trained to classify software components as defective or non-defective. This allows for early detection of potential problems during the software development lifecycle, thereby enhancing software quality.

4.4 Working of Robotics in Software Testing and Defect Prediction

Robotics seems to play an eminent role today in software testing and defect prediction, which generally addresses structural repetitive activities such as test case execution and defect classification. Robotic Process Automation (RPA) tools are typically seen in the software development life cycle, automating manual testing jobs in a short time through user interaction simulation or executing predefined test cases for very early detection of defects. The test cases may well be generated from code changes by the robot, executed, and scanned for possible defects. The robot may document these test results and offer an instant notification on defect presence. Using robots with intelligent models such as LSTM enriches this process with informed decisions.The formula for Defect Density is given in formula (8):

$$\text{Defect Density} = \frac{\text{Number of Defects}}{\text{Lines of Code (LOC)}} \tag{8}$$

Using such parameters, the robotic system can dynamically adapt to testing needs, focusing on areas with the highest likelihood of defects.

4.5 Working of Cloud Storage in Software Defect Prediction

The cloud storage is a scalable and efficient data management in defect prediction systems. As data from software projects becomes larger, platforms of cloud storage, such as Google Cloud Storage, and Azure Blob storage, provide a flexible and secure environment for storing large datasets. With this availability and scalability, these cloud platforms simplify the easy retrieval of data and the processes necessary for obtaining defect predictions by software teams. In the collaborative nature of software development, cloud storage provides a centralized source of defect prediction models and data available to all teams regardless of location. How the cloud infrastructure is utilized to maintain optimal performance and minimize costs is shown in the following formula (9):

$$\text{Cost Efficiency} = \frac{\text{Cost of Cloud Resources}}{\text{Performance Output (Accuracy, Latency)}} \tag{9}$$

ISSN 2321-2152
www.ijmece.com
Vol 6, Issue 1, 2018

5. Result and Discussion

The framework for the prediction of software defects has been successfully realized in the programming language Python. It has imported a technique popularly known as LSTM to classify software defects more accurately. The framework utilizes a Software Defect Prediction Dataset after some preprocessing steps, which include missing value treatment and normalization. The parameters that evaluate the model are, accuracy, precision, recall, and F1-score, giving solid results. Bringing together both Google Cloud Functions and AWS Lambda gave space for scaling, moreover, to make on-time predictions.

5.1 Performance Metrics of The Proposed Framework

Accuracy: Accuracy quantifies the average percentage of correct predictions made by the LSTM model. A high accuracy measure denotes that the model predicts defective and nondetectable instances accurately. It provides a good summary of performance measures in a model; however, it is not sufficient to examine very imbalanced datasets. The formula for Accuracy is shown in Eqn (10):

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

where TP - True Positives, TN - True Negatives, FP - False Positives, FN - False Negatives.

Precision: It refers to the ratio of accurate positive predictions made by a prediction system. The formula for precision is shown in Eqn (11):

$$\text{Precision} = \frac{TP}{TP + FP} \tag{11}$$

where TP - True Positives, FP - False Positives.

Recall: It refers to how well the model detects actual defects. The higher the recall, the less defects are missed; therefore, recall is crucial in ensuring complete software quality assessment and no critical issues are missed. The formula for Recall is shown in Eqn (12):

$$\text{Recall} = \frac{TP}{TP + FN} \tag{12}$$

Where TP - True Positives, FW - False Negatives.

F1-Score: In the case of imbalanced datasets, it gives a response for evaluating performance and balances the trade-off between precision and recall. The formula for F1-Score is shown in Eqn (13):

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{13}$$

where, precision and recall are calculated as above.

5.2 Model Evaluation

The performance metrics evaluation of the proposed Software Defect Prediction Framework via LSTM is shown in Figure 2. The graphs indicate high results for all performance metrics, signifying high model performance concerning the important evaluation criteria.

ISSN 2321-2152
www.ijmece.com
Vol 6, Issue 1, 2018

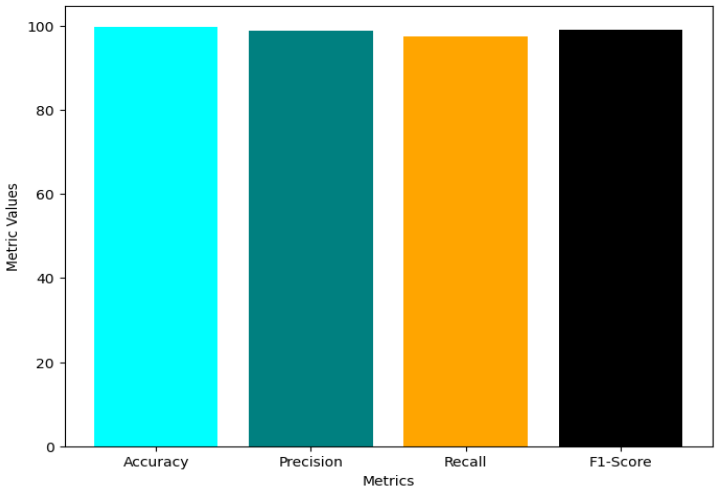

Figure 2: Performance Metrics of the proposed framework

Accuracy at nearly 99.78 percent implies a very high reliability of the model for classifying all instances as defective or otherwise. The precision of 98.89 percent means that whenever the model predicted a defect, it would be mostly correct, and there are very few false positives. Recall of 97.56 percent means that the model detects most of the actual defects, and it has a smaller number of false negatives. The model is also balanced in terms of F1-Score 98.96%, which ensures that defects are neither overlooked nor unnecessary action is taken on non-defective components.

5.2.1 Cloud performance metrics

True efficiency in prediction efficiency scaling of defect prediction software models is compared against one another with and without Knowledge Distillation (KD) across the number of characters available as input to a software model. Hence, the baseline is established against inference time in seconds for both frameworks against the number of characters entered.

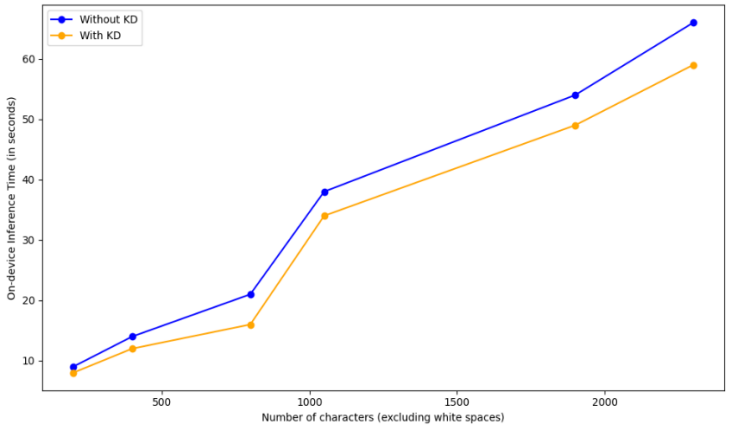

Figure 3: Inference time for the proposed framework

Inference time for the proposed framework is shown in Figure 3. Under the blue line appears a constant slope that indicates a direct proportional increase in inference time with an increase in several characters. In itself, this observation clearly states that larger inputs require much longer durations to process, contrary to outputs fed into smaller inputs. From the figure above, it can be noted, however, that the orange line contributes great to reducing inference time even with larger inputs. The distortion brought by Knowledge Distillation optimization occurred very prominently observed. Under the increasing size of tasks that require input, KD maintains inference times lower than the model without KD. This further confirms that using KD would provide faster predictions, in effect rendering the entire framework more scalable to real-time defect prediction in larger software projects.

ISSN 2321-2152
www.ijmece.com
Vol 6, Issue 1, 2018

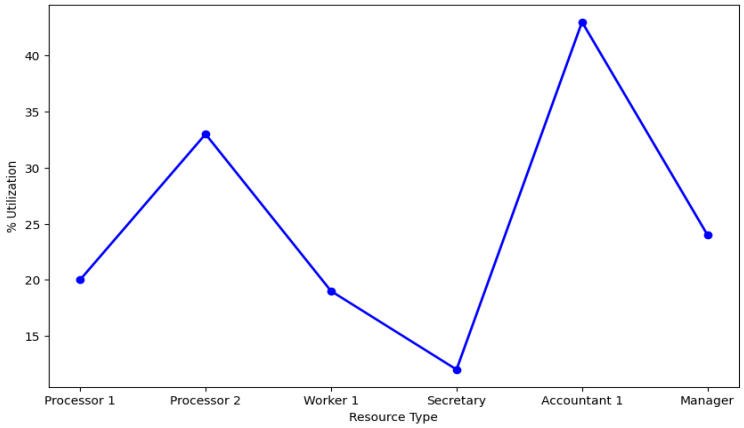

Figure 4: Resource utilization

The resource utilization across various types of resources in the proposed framework is shown in Figure 4. Among the types of resources in the New Framework against Processor 1, Processor 2, Worker 1, Secretary, and Accountant 1. Resource utilization by Processor 2 is at its peak among the five resources considered: Processor 2 is a whopping 35% high, and this great use emphasizes its importance when handling exhaustive computation tasks. The next processor, Processor 1, utilizes about 20%, indicating usage is not entirely constant but brings about less intensity. Worker 1 appears with a relatively low utilization of 19%.

5.3 Discussion

The suggested framework is all-encompassing and spearheads LSTM-based fault prediction with automation robotics and cloud computing scalability, so it meshes easily within the bounds of software defect prediction. The KD, which Knowledge Distillation uses, can reduce the inference time of the model and fast-track the output of these figures, especially for larger inputs. For effective storage and management of data, it eliminates repetitive test processes through robotic process automation, reducing the human cost. The proposed performance metrics of the framework present a balanced scorecard, having a high level of accuracy and precision without compromising on recall or resource utilization efficiency. The framework will generally provide a solution that could be flexible, scalable, and affordable for real-time software fault detection in the presence of dynamic development environments.

6. Conclusion and future works

The proposed framework for software defect prediction employing LSTM, robotics, and cloud computing has recorded a strong performance of accuracy 99.78%, precision 98.89%, recall 97.56%, and F1-score 98.96%, indicating its efficacy at identifying defects with minimum error. The use of Knowledge Distillation (KD) improves inference time, thus enabling the system to scale well for predictions. The cloud infrastructure ensures optimal resource utilization and increases its adaptability for larger datasets. Future work includes improving the adaptability of models by continuous learning, improving inter-cloud compatibility, integrating this model with DevOps pipelines, extending support for multiple programming languages, and optimizing cloud resource management for cost reduction and scalability. With these improvements, the framework will be made more flexible and effective in dynamic software development environments.

References

[1] Zein, S., Salleh, N., & Grundy, J. (2016). A systematic mapping study of mobile application testing techniques. *Journal of Systems and Software*, *117*, 334-356.
[2] Wei, J., & Li, G. (2014). Automated lung segmentation and image quality assessment for clinical 3-D/4-D-computed tomography. *IEEE journal of translational engineering in health and medicine*, *2*, 1-10.
[3] Wiklund, K., Eldh, S., Sundmark, D., & Lundqvist, K. (2017). Impediments for software test automation: A systematic literature review. *Software Testing, Verification and Reliability*, *27*(8), e1639.

[4] Xia, M., Li, T., Zhang, Y., & De Silva, C. W. (2016). Closed-loop design evolution of engineering system using condition monitoring through internet of things and cloud computing. *Computer Networks*, *101*, 5-18.

[5] Ren, L., Zhang, L., Wang, L., Tao, F., & Chai, X. (2017). Cloud manufacturing: key characteristics and applications. *International journal of computer integrated manufacturing*, *30*(6), 501-515.

[6] Vahabzadeh, A., Fard, A. M., & Mesbah, A. (2015, September). An empirical study of bugs in test code. In *2015 IEEE international conference on software maintenance and evolution (ICSME)* (pp. 101-110). IEEE.

[7] Hui, T. K., Sherratt, R. S., & Sánchez, D. D. (2017). Major requirements for building Smart Homes in Smart Cities based on Internet of Things technologies. *Future Generation Computer Systems*, *76*, 358-369.

[8] Ståhl, D., & Bosch, J. (2014). Modeling continuous integration practice differences in industry software development. *Journal of Systems and Software*, *87*, 48-59.

[9] Wu, D., Thames, J. L., Rosen, D. W., & Schaefer, D. (2013). Enhancing the product realization process with cloud-based design and manufacturing systems. *Journal of Computing and Information Science in Engineering*, *13*(4), 041004.

[10] Laniak, G. F., Olchin, G., Goodall, J., Voinov, A., Hill, M., Glynn, P., ... & Hughes, A. (2013). Integrated environmental modeling: a vision and roadmap for the future. *Environmental modelling & software*, *39*, 3-23.

[11] Beckers, K., Côté, I., Faßbender, S., Heisel, M., & Hofbauer, S. (2013). A pattern-based method for establishing a cloud-specific information security management system: Establishing information security management systems for clouds considering security, privacy, and legal compliance. *Requirements Engineering*, *18*, 343-395.

[12] Yarlagadda, R. T. (2017). AI Automation and it's Future in the UnitedStates. *International Journal of Creative Research Thoughts (IJCRT), ISSN*, 2320-2882.

[13] Banyal, R. K., Jain, P., & Jain, V. K. (2013, September). Multi-factor authentication framework for cloud computing. In *2013 fifth international conference on computational intelligence, modelling and simulation* (pp. 105-110). IEEE.

[14] Freet, D., Agrawal, R., John, S., & Walker, J. J. (2015, October). Cloud forensics challenges from a service model standpoint: IaaS, PaaS and SaaS. In *Proceedings of the 7th International Conference on Management of computational and collective intElligence in Digital EcoSystems* (pp. 148-155).

[15] Haus, M., Waqas, M., Ding, A. Y., Li, Y., Tarkoma, S., & Ott, J. (2017). Security and privacy in device-to-device (D2D) communication: A review. *IEEE Communications Surveys & Tutorials*, *19*(2), 1054-1079.

[16] Yin, H., & Jha, N. K. (2017). A health decision support system for disease diagnosis based on wearable medical sensors and machine learning ensembles. *IEEE Transactions on Multi-Scale Computing Systems*, *3*(4), 228-241.

[17] Younas, M., Awan, I., & Mecella, M. (2016, August). Mobile Web and Intelligent Information Systems. In *13th International Conference, MobiWIS 2016 Vienna, Austria, August 22–24*.

[18] Mishra, P., Pilli, E. S., Varadharajan, V., & Tupakula, U. (2017). Intrusion detection techniques in cloud environment: A survey. *Journal of Network and Computer Applications*, *77*, 18-47.

[19] Cai, R., Zhu, B., Ji, L., Hao, T., Yan, J., & Liu, W. (2017, November). An CNN-LSTM attention approach to understanding user query intent from online health communities. In *2017 ieee international conference on data mining workshops (icdmw)* (pp. 430-437). IEEE.

[20] Murji, A., Luketic, L., Sobel, M. L., Kulasegaram, K. M., Leyland, N., & Posner, G. (2016). Evaluating the effect of distractions in the operating room on clinical decision-making and patient safety. *Surgical endoscopy*, *30*, 4499-4504.

[21] Shahane, S., & Kulkarni, R. (2014). Cloud Auditing: An Approach for Betterment of Data Integrity. *International Journal of Soft Computing and Engineering*, *3*(6), 107-112.

[22] Castaneda, C., Nalley, K., Mannion, C., Bhattacharyya, P., Blake, P., Pecora, A., ... & Suh, K. S. (2015). Clinical decision support systems for improving diagnostic accuracy and achieving precision medicine. *Journal of clinical bioinformatics*, *5*, 1-16.

[23] Belmonte, D. L., Linhares, R. R., Stadzisz, P. C., & Simao, J. M. (2016). A new method for dynamic balancing of workload and scalability in multicore systems. *IEEE Latin America Transactions*, *14*(7), 3335-3344.

[24] Gharaibeh, A., Salahuddin, M. A., Hussini, S. J., Khreishah, A., Khalil, I., Guizani, M., & Al-Fuqaha, A. (2017). Smart cities: A survey on data management, security, and enabling technologies. *IEEE Communications Surveys & Tutorials*, *19*(4), 2456-2501.

[25] Sharma, S. K., & Wang, X. (2017). Live data analytics with collaborative edge and cloud processing in wireless IoT networks. *IEEE Access*, *5*, 4621-4635.