International Journal of Innovation Studies

ISSN:2096-2487 | E-ISSN:2589-2975

AN ADAPTIVE GA-OPTIMIZED AUTOMATED X-LEARNING MODEL FOR INTEGRATED SOFTWARE RELIABILITY PREDICTION

Shaik Shakeer Basha[1], Dr.R.Satya Prasad[2], Dr.Syed Khasim[3]

[1]Research Scholar, Department of Computer Science, Rayalaseema University, Kurnool, Andhra Pradesh, India
bashassb4@gmail.com

[2]Professor, Department of Computer Science & Engineering, Acharya Nagarjuna University, Nagarjuna Nagar, Guntur, Andhra Pradesh, India.
profrsp@gmail.com

[3]Professor, Department of Computer Science & Engineering, Dr.Samuel George Institute of Engineering & Technology, Markapur, Prakasam District, Andhra Pradesh, India.
khasim716@gmail.com

ABSTRACT

Software reliability is a significant challenge for software companies today, enabling the development of high-quality, stable, and sustainable software systems. Many existing approaches lack accuracy, are poor in prediction, and fail to handle the highly complex software bugs. To overcome these challenges, we have presented an Adaptive Genetic Algorithm (GA)-Optimized Automated X-Learning Model (OA-XLM) for integrated software reliability prediction. The main aim of this model is to predict and detect the accuracy bugs (software threats) from the latest software datasets. The proposed approach combines an adaptive genetic algorithm (AGA) to find an accurate Learning rate, an in-depth selection of intelligent features, and hyper parameters optimization. Here, the automated X-learning model (XLM), also known as Extreme-Learning, performs bug prediction and failure rate estimation. The XLM integrates multiple learning layers to capture both linear and nonlinear relationships in software data. Finally, the XLM also classifies the software bugs with binary classification. This model was primarily developed to adapt dynamically across complex software projects. In this work, two benchmark datasets are used, namely the NASA and PROMISE repositories. The results show that the proposed approach achieved high prediction and classification rates.

Keywords: Adaptive Genetic Algorithm, Optimized Automated X-Learning Model, Software Bugs, X-Learning Model.

1. INTRODUCTION

The software reliability prediction is a very important feature of the contemporary software engineering since it guarantees the establishment of reliable and quality systems. As software applications become more complex, the old reliability measurement approaches in use tend to be less effective because the different software measurements of complexity, size, fault history, and behavior display dynamic interactions. The traditional methods normally deal with the minor issues such as prediction of defects or the estimation of failures rate which does not give

the complete picture of software reliability. Consequently, a greater significance of combined methods that are able to concomitantly examine various reliability variables, and provide precise, scalable, and evidence-based forecasts is emerging. Integrated software reliability prediction is the approach aimed at overcoming these issues and requires a combination of various analytical methods, such as machine learning, statistical model, and optimization methods, into a single framework. By using these approaches, it is possible to predict software defects and failure rates at the same time giving a whole picture of software system reliability over the life of the software. Integrated models are able to effectively cope with high-dimensional data, reduce redundancy and enhance prediction accuracy by using the high-level techniques of ensemble learning and evolutionary optimization. This does not only aid the detection of faults and risks early but also improves the decision making process in software development and maintenance which eventually results to more reliable and sound software systems.

Machine learning (ML) is crucial to integrated software reliability prediction, which allows analyzing complex software metrics and failure patterns in a data-driven manner. The data produced by modern software systems are massive e.g. code metrics, fault history, execution logs and change records, and tend to be nonlinear and high dimensional. Decision trees (DT), support vector machines (SVM), random forests (RF), and ensemble models are an example of an ML algorithm that can use this type of data to learn the hidden pattern and relationships. By measuring historical and real-time data, these models can be useful in predicting defects in software (bug prediction) as well as estimating the failure rates and increasing the accuracy and efficacy of reliability assessment. Even with the current developments that have taken place in the field of software reliability prediction, there are still a number of challenges which cannot be solved in the existing methods. In the majority of conventional approaches, defect prediction or failure rate estimation is considered in isolation and results in piecemeal analysis and partial reliability evaluation. Also, most models use manual selection of features, and any static parameter that do not necessarily reflect the relationship complexities and non-linearities that apply in software. The fact that irrelevant, high-dimensional, and redundant features exist worsens the model performance and makes them more complex to compute. Finally, present ML algorithms failed to perform generalization when dealing with other software projects or datasets, which leads to an uneven prediction accuracy. They can also be affected by overfitting, non-interpretability and inadequate flexibility to the altering software environments. Consequently, such a framework as an integrated, adaptive, and optimized one that is capable of both doing the task of bug prediction and failure rates estimation and dealing with the high-dimensional data in the efficient manner is in need. A system like this must also make use of well-developed machine learning and optimization strategies to enhance the degree of prediction, generalization, and help in making reliable decisions when doing the software engineering processes.

2. LITERATURE SURVEY

Ghafoor Hussain et al. [9] presented an improved transformer-based system that uses the CodeBERT to predict software defects in multi classes by using fine-grained classification. The current model is a pre-trained CodeBERT architecture that learns rich semantic representations in source code and will be useful in acquiring syntax, structure, and contextual relationships. The framework uses a framework that has an improved feature refinement process and

attention-based learning to identify the various categories of defects, including logical errors, performance problems, and security vulnerabilities. Also, higher-level preprocessors, such as code normalization and tokenization, are utilized to enhance the data uniformity and model stability. Adaptive training strategies and fine-tuning are other methods of further optimization of the model to improve classification performance on multiple software datasets. A parallel ML algorithm realized by applying fine-grained execution mode of Apache Spark on big data cloud computing infrastructure based on the Apache Mesos was proposed by Xian [10]. The proposed system allows to process large-scale robots data streams effectively and efficiently to detect and classify faults accurately. The framework incorporates the distributed data ingestion, preprocessing and feature extraction mechanisms to deal with multimodal sensor input like vibration, temperature and motion signals. The approach to parallelization is a fine-grained one where calculations are broken down into smaller units such that the resource utilization process becomes efficient and the model training process can be performed quicker on the distributed nodes. There are several ML algorithms such as classification and anomaly detection models that are implemented in parallel and enhance the accuracy and resilience of fault recognition. The system also allows a dynamically allocated resource via Mesos so that it remains adaptable and capable of performing well at varying workloads.

Roy et al. [11] proposed a novel forecasting framework based on a Neighborhood Fuzzy Particle Swarm Optimization (NF-PSO) integrated neural network. The proposed approach combines the strengths of fuzzy logic, particle swarm optimization, and neural networks to enhance prediction accuracy and robustness in software reliability estimation. In the proposed model, fuzzy logic is employed to handle uncertainty and imprecision in software reliability data, while the neighborhood-based particle swarm optimization strategy improves the exploration and exploitation capabilities of the optimization process. The NF-PSO is utilized to optimize the weights and biases of the neural network, enabling efficient learning of complex nonlinear relationships between software metrics and failure patterns. The neighborhood structure allows particles to share information locally, improving convergence speed and avoiding premature stagnation. The neural network then utilizes the optimized parameters to forecast software reliability metrics such as failure rate and cumulative failures over time. Hassouneh et al. [12] presented the proposes a Boosted Whale Optimization Algorithm (BWOA) enhanced with natural selection operators for efficient and accurate software fault prediction. The proposed approach integrates the exploration and exploitation capabilities of the Whale Optimization Algorithm (WOA) with evolutionary principles inspired by natural selection, including selection, crossover, and mutation mechanisms. These operators are incorporated to improve population diversity, avoid local optima, and enhance convergence speed. Furthermore, a boosting strategy is employed to iteratively refine weak learners and improve classification performance. The optimized feature subset and model parameters are then utilized within a machine learning framework to accurately classify software modules as defective or non-defective.

Kaliraj et al. [13] investigates the impact of class imbalance and generalization issues in cross-project software fault prediction and proposes a robust analytical framework to address these challenges. The approach incorporates advanced data preprocessing techniques, including resampling strategies such as SMOTE and under sampling, to balance defective and non-defective classes. Additionally, domain adaptation and feature normalization methods are

employed to reduce distributional differences between source and target projects. Multiple machine learning models, including ensemble methods and transfer learning-based approaches, are evaluated to assess their ability to generalize across projects. Javdani Gandomani et al. [14] proposes an enhanced analogy-based software effort estimation framework that integrates regression methods to improve prediction accuracy. The proposed approach first identifies similar historical projects using a similarity measurement mechanism and then applies a combination of regression techniques, such as linear regression and nonlinear regression models, to refine the effort estimation process. By leveraging regression-based learning, the model captures underlying patterns and relationships within the data, enabling more precise estimation compared to traditional analogy-based methods. Alaswad et al. [15] presented a ML-based framework for software quality prediction that leverages historical software metrics, including code complexity, size, change history, and fault data. Various supervised learning algorithms are used to learn patterns associated with software quality issues. Data preprocessing techniques, including feature selection, normalization, and handling class imbalance, are incorporated to improve model performance. The proposed approach aims to accurately classify software modules into defective and non-defective categories while minimizing false predictions. Kumar et al. [16] presented a reliability prediction analysis framework based on soft computing techniques, which are well-suited for handling uncertainty, imprecision, and nonlinear relationships inherent in software systems. The proposed approach integrates soft computing methods such as fuzzy logic, artificial neural networks, and evolutionary algorithms to model and predict the reliability of aspect-oriented applications. Fuzzy logic is employed to manage uncertainty in software metrics and aspect interactions, while neural networks learn complex nonlinear patterns from historical reliability data. Evolutionary optimization techniques are used to enhance model performance by selecting optimal features and tuning parameters. The framework considers various aspect-oriented metrics, including coupling, cohesion, and interaction complexity, to provide a comprehensive reliability assessment. Wang et al. [17] proposed a DL-based framework for software reliability prediction using a Recurrent Neural Network (RNN) encoder–decoder architecture, which is well-suited for modeling sequential and time-series data. In the proposed approach, the RNN encoder processes sequences of software execution data, such as failure occurrences and inter-failure time intervals, and encodes them into a fixed-length context vector that captures temporal dependencies and system behavior. The decoder then utilizes this context representation to predict future reliability metrics, including failure rates and cumulative failures over time. Advanced variants of RNN, such as Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRU), are employed to overcome issues related to vanishing gradients and to enhance long-term dependency learning. Additionally, data preprocessing techniques, including normalization and sequence transformation, are applied to improve model performance and stability.

3. METHODOLOGY

The presented Adaptive Genetic Algorithm (AGA)-Optimized Automated X-Learning Model (OA-XLM) is aimed at being a unified framework of software reliability prediction, both in bug detection and in the estimation of failure rates. The first stage involves gathering software datasets based on benchmark repositories, i.e. NASA and PROMISE and processing them through preprocessing to maintain data consistency, including data cleaning, normalization and

dealing with missing values. The adaptive genetic algorithm is used in the initial phase of the methodology to perform intelligent selection of features and optimize hyperparameters. The AGA adapts the important parameters crossover and mutation rates dynamically depending on the fitness values which facilitates exploration and exploitation of search space effectively. It determines the most relevant software metrics, and also identifies the best learning parameters, such as the learning rate, which minimizes redundancy, and maximizes model performance.

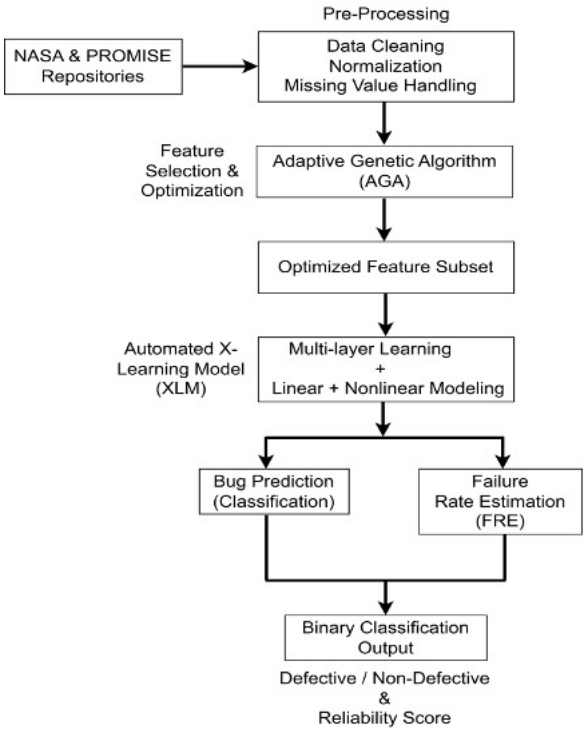

Figure 1: Architecture Diagram for Adaptive GA-Optimized Automated X-Learning Model (OA-XLM)

The second stage is the input of the optimized set of features to the Automated X-Learning Model (XLM), which is also called Extreme Learning, which aims at capturing the linear and nonlinear relationships that exist between software data. The XLM comprises of several learning layers, which increase the capability of representation and also enhance accuracy of prediction. The model is able to predict the software bugs at the same time it estimates the software failure rate giving an overall evaluation of the reliability of the software. Lastly, a binary classification system is used to classify software modules to defective and non-defective. The proposed OA-XLM framework is adaptable in nature and hence can be generalized into different and complex software projects. As seen through experimental analysis on NASA and PROMISE data sets, the proposed approach is robust, scaled to be used, and more predictive than the traditional models.

4. DATASET DESCRIPTION

The most popular datasets of software defect classification is NASA (MDP) dataset, which includes a variety of project modules, including KC1, KC2, JM1, CM1, PC1 and PC2. Together these datasets have about 20,712 software modules with 2,685 considered defective (bug-prone) and 18,027 non-defective (clean modules). Each record is a software component characterized by the metrics of static code lines of code, complexity, and Halstead features and a binary class label (1 defective and 0 non-defective). The distinctive feature of the NASA data

is the imbalance between the classes with the non-defective samples being much higher than the defective ones, which is why the data can be used to assess the strength and performance of classification models in predicting software reliability.

The PROMISE repository datasets are also offering the widest range of analysis through incorporation of data that comes through various software programs like the eclipse and apache as well as NASA derived datasets. The combined data in some common PROMISE subsets (e.g. KC1, JM1, Eclipse versions and Apache projects) is usually some 14,000 to 15,000 records, comprising of approximately 2,600-2,700 defective records and the rest 11,000+ non-defective records. These data sets consist of process and non-process data, and have a greater and more varied feature space to use in classification. Just like NASA datasets, PROMISE datasets have also the problem of class imbalance and high dimensionality, which require application of feature selection and optimization techniques. In general, both data sets represent a rich basis of training and testing machine learning models to predict software defects and integrated software reliability analysis.

5. PRE-PROCESSING

Preprocessing is one of the important aspects of software reliability prediction that makes the dataset clean, consistent, and ready to be trained in machine learning models. It starts with data cleaning, where it is possible to recognize noisy data, duplicate data and inconsistent data, and remove them to enhance data quality. Normalization is then used to put feature values into a common range (e.g. 0 to 1), which is useful to make models learn effectively and avoid having features with large values to dominate the learning process. Besides, missing value treatment is done to solve incomplete data by dropping records that lack data or filling them with the methods of mean, median, or mode replacement. Combined, these preprocessing activities increase the accuracy of the dataset, minimize errors, and increase the general performance and accuracy of predictive models.

5.1 Feature Selection & Optimization: Adaptive Genetic Algorithm (AGA)

Adaptive Genetic Algorithm (AGA) features selection and optimization is critical towards enhancing the performance of software reliability prediction models. Under this method, every possible solution (chromosome) is defined as a subset of features, and other model parameters, which are usually binary or real-valued. The AGA starts with some random population of these chromosomes. Each solution is evaluated by a fitness function - typically the classification accuracy, error rate or a combination of the performance measures. In contrast to the classical genetic algorithms, AGA dynamically changes its control parameters crossover probability, and mutation rate, according to the fitness of individuals. Those solutions which perform highly are maintained at low mutation rates to ensure stability and those with low performance are subjected to high mutation rates to promote exploration. The algorithm effectively explores the most informative combination of features and at the same time optimizes model parameters like the learning rate and hyperparameters through iterative steps of selection, crossover and mutation.

The adaptive AGA assists in the balancing of exploration and exploitation and hence prevents premature convergence, which results in a more thorough search of the solution space. The removal of redundant and irrelevant features results in the AGA of data dimensionality, reduction of the computational complexity, as well as in the increase of the generalization properties of the learning model. Also, the parameter optimization is also integrated into the

same structure, so the model can automatically adjust its internal parameters, resulting in a stronger predictive and a more robust model. This is especially useful with high-dimensional and imbalanced datasets such as NASA and PROMISE used in software. In general, the feature selection and optimization based on AGA present a smart and automated system of determining the most important inputs and the best set of settings, which has a significant impact on enhancing the efficiency and effectiveness of built software reliability prediction systems.

5.2 Optimized Feature Subset

The final feature set of the most relevant and informative features is referred to as the optimized feature subset after the optimization methods have been applied including the Adaptive Genetic Algorithm (AGA). The optimization process is used to identify a smaller set of software metrics that produces the greatest contribution to software defects and reliability prediction instead of using all available software metrics, which may contain redundant, irrelevant, or noisy features. This subset is achieved through the review of many sets of candidate features in accordance with a fitness metric (e.g., classification accuracy, error rate, or F1-score) with only features that have great predictive power being saved. Through the choice of these features, the model has the advantage of less dimensionality, less computational complexity and higher learning rates. Also, feature elimination can aid in preventing overfitting and increase the generalization of the model to other datasets as NASA and PROMISE. In general, the optimized feature subset is a whittled input space, which allows more reliable, consistent, and predictable software reliability forecasting.

5.3 Automated X-Learning Model (XLM)

Extreme Learning is also known as Automated X-Learning Model (XLM) AXLM or the Extreme Learning Model is a rapid and effective learning model that is used to analyze high-dimensional data in software reliability prediction. Also in contrast to the traditional neural networks, which perform backpropagation to update weights, XLM assigns the input weight and hidden layer bias randomly and calculates the output weight analytically, which saves a lot of training time. This feature makes XLM very appropriate to large scale software databases like NASA and PROMISE where quick learning and scaling is necessary. Within the suggested model, the optimized set of features produced by the Adaptive Genetic Algorithm is given to XLM which then acquires patterns related to software failure behavior and defects. Its capability to handle huge amounts of information with a low level of computational load makes it possible to converge faster and still have high levels of prediction accuracy.

Moreover, XLM can dynamically fit to the various software projects and data distributions without major manual tuning due to the automated nature of XLM. It is able to well represent intricate associations between software measures, such as connections among code complexity, code size and faultiness. The model combines various layers of learning and transformation processes to advance the representation of features to promote the strength and the generalization potential of the prediction system. Therefore, XLM is a central element in the unified structure and it can also be used to predict bugs and estimate failure rates more efficiently and reliably.

5.4.1 Multi-Layer Learning

XLM is further improved in the multi-layer learning mechanism, which facilitates software data in extracting hierarchical representations by processing features in multiple transformation layers. Learning patterns are learned gradually in each layer, initially of simple relationships

between the input features, to more detailed interactions that contribute to software defects. This multilevel structure enhances the capability of the model to articulate the little nuances and dependencies in the data which results in more precise predictions. With a series of learning layers, the model is more able to manipulate high-dimensional data and discover more insight regarding the features of software reliability.

5.4.2 Linear and Nonlinear Modeling with Classification Layer

The XLM model has the capability of integrating both linear and nonlinear modeling to model various relationships that are evident in software data. Linear modeling can be used to find simple correlations between features and defect labels and nonlinear modeling can be used to represent complex interactions and hidden patterns that cannot be expressed linearly. The combined representations are further fed into a classification layer that is usually implemented by a decision or activation function to classify software modules as either defective (bug-prone) or non-defective. Such end classification step will provide good predictions to rely on when evaluating the reliability of software, and also the model will be able to make accurate decisions. The following equations are used to process the features:

Step 1: This step mainly captures the relationship between features and output:

$$Z_{\text{linear}} = WX + b \quad (1)$$

$X \in R^n \rightarrow$ input feature vector;
$W \in R^{1\times n} \rightarrow$ Weight vector;
$b \in R \rightarrow$ Bias; Z_{linear}- Linear output.

Step 2: Hidden Layer Transformation

$$H = f(W_1X + b_1) \quad (2)$$

Step 3: Integrated Layer: It is finally shows the transformed result after nonlinear learning:

$$Z = W_2H + b_2 \quad (3)$$

Step 4: Classification layer: This step mainly used to classify the two classes using sigmoid function:

$$\hat{y} = \frac{1}{1 + e^{-Z}} \quad (4)$$

$$\text{Class} = \begin{cases} 1 \text{ if } \hat{y} \geq 0.5 \text{ (Defective)} \\ 0 \text{ if } \hat{y} < 0.5 \text{ (Non} - \text{Defective)} \end{cases} \quad (5)$$

5.5 Performance Metrics

Performance measures are needed to determine the usefulness of software reliability prediction models by measuring the level of accuracy in detecting defective and non-defective modules and predicting the failure behavior. Accuracy is used to measure the general accuracy of the model which means the percentage of model predictions that are accurate. Precision, which measures how well the model correctly marks out defective modules without the false alarm, and recall (sensitivity), which measures how well the model marks out all the actual defected ones. F1-score has a balanced evaluation because it sums the precision and recall and is, therefore, especially handy with imbalanced datasets like NASA and PROMISE. Also, error

rate shows the percentage of inaccurate prediction, which points at weak points in the model. In the case of estimating failure rates, regression measures, which include Mean Absolute Error (MAE), root mean square error (RMSE), and R2 score are applied to determine the accuracy of prediction, deviation, and goodness of fit. Combining these measures is a way of getting an overall assessment of the classification as well as regression performance that will make the proposed model reliable and robust.

5.6 Results and Discussions

Table 1 offers a comparative analysis of ANN, XLM, and proposed OA-XLM model in terms of failure rate estimation on NASA dataset in terms of MAE, RMSE and R2 score. The ANN model has a higher error value of 0.137 and RMSE of 0.189 which implies that it has lower accurate predictions whereas the R2 score of the ANN model is 0.85 which implies moderate explanatory capacity. XLM model is more effective with a low MAE of 0.116 and RMSE of 0.169 and a higher R2 of 0.91 that proves to be more effective in learning nonlinear relationships and predicting. The suggested OA-XLM model is, however, much better than both models with the lowest MAE (0.079) and RMSE (0.138) which implies that the model does not make many errors in predicting and the highest R2 value of 0.96 which shows a very good fit and high generalization. This high quality performance can be primarily attributed to incorporation of the Adaptive Genetic Algorithm which optimizes the feature selection and hyper-parameters that helps the model to concentrate on the most appropriate information and generate more accurate and reliable predictions of the failure rate.

Table 1: Failure Rate Estimation Performance for NASA Dataset

Algorithms	MAE	RMSE	R^2 Score
ANN	0.137	0.189	0.85
XLM	0.116	0.169	0.91
Proposed OA-XLM	0.079	0.138	0.96

Table 2 shows the performance of ANN, XLM, and the proposed model of OA-XLM to estimate failure rate with respect to the use of MAE, RMSE, and R2 score on the PROMISE dataset. ANN model accepts greater values of errors with MAE = 0.145 and RMSE = 0.177, which indicates lower prediction accuracy, and its R2 = 0.84 which is an indicator of moderate ability to explain the variance in the data. This is because the XLM model yields better performance of 0.124 MAE and 0.158 RMSE with a higher R2 value of 0.92 showing better performance of learning complex patterns and better estimation. Nevertheless, the proposed OA-XLM model performs much better as compared to ANN and XLM, as it has the lowest MAE (0.083) and RMSE (0.121), which means it has a small error in the prediction, and the highest R2 value of 0.97, meaning that it has an excellent fit and good ability to generalize. This high performance is due to the combination of the Adaptive Genetic Algorithm that is useful in optimizing the selection of features and model parameters and thus produces more accurate, strong, and efficient failure rate forecasts on the various and high dimensional PROMISE data.

Table 2: Failure Rate Estimation Performance for PROMISE Dataset

Algorithms	MAE	RMSE	R^2 Score
ANN	0.145	0.177	0.84
XLM	0.124	0.158	0.92
Proposed OA-XLM	0.083	0.121	0.97

Table 3: Quantitative Performance Algorithms on NASA Dataset

Algorithms	ANN	XLM	Proposed OA-XLM
Precision	0.84	0.89	0.96
Recall	0. 85	0.91	0.95
Specificity	0.86	0.89	0.96
Accuracy	0.84	0.90	0.97
F1 Score	0.81	0.91	0.98

Table 4: Quantitative Performance Algorithms on PROMISE Dataset

Algorithms	ANN	XLM	Proposed OA-XLM
Precision	0.86	0.90	0.97
Recall	0. 87	0.92	0.96
Specificity	0.84	0.88	0.97
Accuracy	0.85	0.91	0.98
F1 Score	0.83	0.92	0.98

6. Conclusion

This paper described an Adaptive Genetic Algorithm (AGA)-Optimized Automated X-Learning Model (OA-XLM) of integrated software reliability prediction, that is, both bug prediction and failure rate forecasting in a single model. The proposed solution is effective in integrating the optimization potential of AGA with the rapid learning and generalization potential of the X-Learning Model to process high dimensional and complex software datasets. The model will help to eliminate redundancy, increase the speed of learning, and improve the predictive capability by conducting intelligent feature selection and hyperparameter optimization. The combination of the linear and nonlinear modeling also allows the system to present more complex relationships between the software metrics leading to more precise classification of defective and non-defective modules. Results of experiments on benchmark datasets like NASA and PROMISE indicate that the proposed OA-XLM model is significantly

better than the conventional machine learning models and baseline methods regarding their accuracy, precision, recall and error reduction. Also, the model provides better performance in terms of failure rate estimation, that is, the lower value of MAE and RMSE and high value of R2. The framework is adaptive hence strong in terms of robustness, scalability, and robust generalization in various software projects. In general, the proposed model gives an effective and stable approach to the software reliability prediction to aid in the early fault detection, better decision-making, and quality of software. Future research can be conducted on the incorporation of deep learning methods and real-time data analytics in order to increase prediction capacity and applicability in dynamic software environments.

References

[1] S. Shaikh and S. Ghosh, "Enhancing Software Reliability Through Machine Learning: Prediction Through Evaluation Metrics," 2024 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS), Pune, India, 2024, pp. 1-6, doi: 10.1109/ICBDS61829.2024.10837281.

[2] Behera, A.K., Chaudhury, P. & Dash, C.S.K. A comprehensive survey on intelligent software reliability prediction. Discov Computing 28, 90 (2025). https://doi.org/10.1007/s10791-025-09597-z.

[3] A. Jindal, A. Gupta and Rahul, "Comparative Analysis of Software Reliability Prediction Using Machine Learning and Deep Learning," 2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS), Coimbatore, India, 2022, pp. 389-394, doi: 10.1109/ICAIS53314.2022.9743129.

[4] M. Rahman, H. Sarwar, M. A. Kader, T. Gonçalves and T. T. Tin, "Review and Empirical Analysis of Machine Learning-Based Software Effort Estimation," in IEEE Access, vol. 12, pp. 85661-85680, 2024, doi: 10.1109/ACCESS.2024.3404879.

[5] G. E. d. P. Rodrigues, A. M. Braga and R. Dahab, "Detecting Cryptography Misuses With Machine Learning: Graph Embeddings, Transfer Learning and Data Augmentation in Source Code Related Tasks," in IEEE Transactions on Reliability, vol. 72, no. 4, pp. 1678-1689, Dec. 2023, doi: 10.1109/TR.2023.3237849.

[6] P. Y. P. Chan and J. Keung, "Validating Unsupervised Machine Learning Techniques for Software Defect Prediction With Generic Metamorphic Testing," in IEEE Access, vol. 12, pp. 165155-165172, 2024, doi: 10.1109/ACCESS.2024.3494044.

[7] M. Ali et al., "Software Defect Prediction Using an Intelligent Ensemble-Based Model," in IEEE Access, vol. 12, pp. 20376-20395, 2024, doi: 10.1109/ACCESS.2024.3358201

[8] R. He, Y. Li and C. Sun, "Understanding Software Defect Prediction Through eXplainable Neural Additive Models," in IEEE Access, vol. 13, pp. 82860-82873, 2025, doi: 10.1109/ACCESS.2025.3567140.

[9] R. Ghafoor Hussain, K. -C. Yow and M. Gori, "Leveraging an Enhanced CodeBERT-Based Model for Multiclass Software Defect Prediction via Defect Classification," in IEEE Access, vol. 13, pp. 24383-24397, 2025, doi: 10.1109/ACCESS.2024.3525069.

[10] G. Xian, "Parallel Machine Learning Algorithm Using Fine-Grained-Mode Spark on a Mesos Big Data Cloud Computing Software Framework for Mobile Robotic Intelligent Fault Recognition," in IEEE Access, vol. 8, pp. 131885-131900, 2020, doi: 10.1109/ACCESS.2020.3007499.

[11] P. Roy, G. S. Mahapatra and K. N. Dey, "Forecasting of software reliability using neighborhood fuzzy particle swarm optimization based novel neural network," in IEEE/CAA Journal of Automatica Sinica, vol. 6, no. 6, pp. 1365-1383, November 2019, doi: 10.1109/JAS.2019.1911753.
[12] Y. Hassouneh, H. Turabieh, T. Thaher, I. Tumar, H. Chantar and J. Too, "Boosted Whale Optimization Algorithm With Natural Selection Operators for Software Fault Prediction," in IEEE Access, vol. 9, pp. 14239-14258, 2021, doi: 10.1109/ACCESS.2021.3052149.
[13] S. Kaliraj, A. M. Kishoore and V. Sivakumar, "Software Fault Prediction Using Cross-Project Analysis: A Study on Class Imbalance and Model Generalization," in IEEE Access, vol. 12, pp. 64212-64227, 2024, doi: 10.1109/ACCESS.2024.3397494.
[14] T. Javdani Gandomani, M. Dashti, H. Zulzalil and A. B. M. Sultan, "Enhancing Software Effort Estimation in the Analogy-Based Approach Through the Combination of Regression Methods," in IEEE Access, vol. 12, pp. 152122-152137, 2024, doi: 10.1109/ACCESS.2024.3480829.
[15] F. Alaswad and E. Poovammal, "Software quality prediction using machine learning," Materials Today: Proceedings, Mar. 2022, doi: https://doi.org/10.1016/j.matpr.2022.03.165.
[16] P. Kumar, S. K. Singh, and S. Deo Choudhary, "Reliability prediction analysis of aspect-oriented application using soft computing techniques," Materials Today: Proceedings, vol. 45, pp. 2660–2665, 2021, doi: https://doi.org/10.1016/j.matpr.2020.11.518.
[17] J. Wang and C. Zhang, "Software reliability prediction using a deep learning model based on the RNN encoder–decoder," Reliability Engineering & System Safety, vol. 170, pp. 73–82, Feb. 2018, doi: https://doi.org/10.1016/j.ress.2017.10.019.