KeAi
CHINESE ROOTS
GLOBAL IMPACT

International Journal of Innovation Studies



AN IMPROVED SOFTWARE RELIABILITY PREDICTION USING OPTIMIZATION METHODS

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

[1]Research Scholar, Department of Computer Science & Engineering, Rayalaseema University, Kurnool, Andhra Pradesh, India.

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

Abstract: Software reliability prediction is one of the significant tasks that find the faults in software systems. Existing reliability prediction face several difficulties in handling big-dimensional software metrics, can reduce disjointed correlations, and complex fault patterns, that reduce prediction accuracy. These challenges proposed a Novel Optimization Learning Framework (NOLF) for automated software reliability and fault prediction. The proposed NOLF incorporates adaptive feature optimization (AFO) and intelligent learning strategies (ILS) in order to select the most relevant software metrics; to establish the best possible learning parameter settings; and to produce better representations of complex software attributes. In order to find an efficient subset of features and a good model setting, an optimization step takes place. At the same time, an advanced learning model (ALM) can capture both linear and non-linear associations between the optimized features. After identifying these optimized representations, they may be utilized to automatically predict software faults, classify objects as either being faulty or non-faulty, or estimate software reliability. The performance of the proposed model measures the software reliability was tested on several software benchmark datasets from the NASA and PROMISE repositories. Results indicate that the proposed NOLF performs in terms of software reliability assessment, various prediction and classification metrics were employed such as Accuracy, Precision, Recall, F1-Score, MAE, and RMSE.

Keywords: Novel Optimization Learning Framework (NOLF), adaptive feature optimization (AFO), intelligent learning strategies (ILS), advanced learning model (ALM), NASA and PROMISE.

Introduction

Software Reliability has become one of the most important components in Software Engineering as we expect our current generation of software systems to function properly and consistently, as well as being efficient, across varying environments [1]. Traditional Software Reliability Growth Models (SRGMs) have been developed to study the manner in which software failures occur during testing and determine the amount of reliability improvement that occurs as defects are found and corrected [2]. Typically, traditional SRGMs will utilize the data collected on failures occurring during testing to establish a relationship between the time spent testing and either the total number of failures experienced or the rate at which failures occur [3]. However, the increasing complexity of software systems and the wide variety of available software metrics present some difficulties for traditional reliability



models; especially when there exist complex and/or nonlinear relationships between the different attributes of software, faults and reliability [4]. Machine Learning offers the ability to supplement the use of traditional SRGM-based analysis by identifying patterns directly from previous software metrics and defect information.

Most existing Software Reliability Growth Models (SRGMs) as well as traditional approaches to predicting faults in software are plagued by the problems of dealing with large numbers of software metrics, redundant features, and the existence of many linear and/or non-linear relationships between various software attributes and how they relate to the occurrence of defects in software [5]. A lack of effective feature optimization and a failure to select the right set of learning parameters can negatively affect the accuracy of the predictions made by these models; additionally, this lack of selection can also hinder the development of automated systems that can accurately identify faults in software and estimate the overall reliability of the software [6]. Additionally, traditional approaches do not provide mechanisms for integrating feature optimization with intelligent learning in order to support simultaneously the tasks of software fault prediction, classifying parts of the software as either being "faulty" or "non-faulty", and estimating the reliability of the software [7] [8]. As such, what is needed is an integrated and automated method that will enable us to determine which software metrics are the most relevant ones to consider when assessing software reliability and predicting faults in software, and then use optimization methods to find the optimal configuration for the learning process so that we can successfully train the learning algorithm on the complex relationships existing within software, thus improving our ability to assess software reliability and predict faults in software.

In this paper, the proposed novel optimization learning framework (NOLF) provides a method of automating the process of predicting the reliability and faults in software through the integration of adaptive feature optimization (AFO) with intelligent learning strategies (ILS). AFO identifies the most appropriate software metrics to be utilized in developing a model to predict software reliability and faults, as well as optimizing the parameters associated with that model. In addition to being able to capture both linear and non-linear relationships that exist between the optimized features identified by AFO, the intelligent learning strategy is also capable of producing optimized feature representations which can then be used to detect software faults; classify software as either faulty or non-faulty; and estimate software reliability.

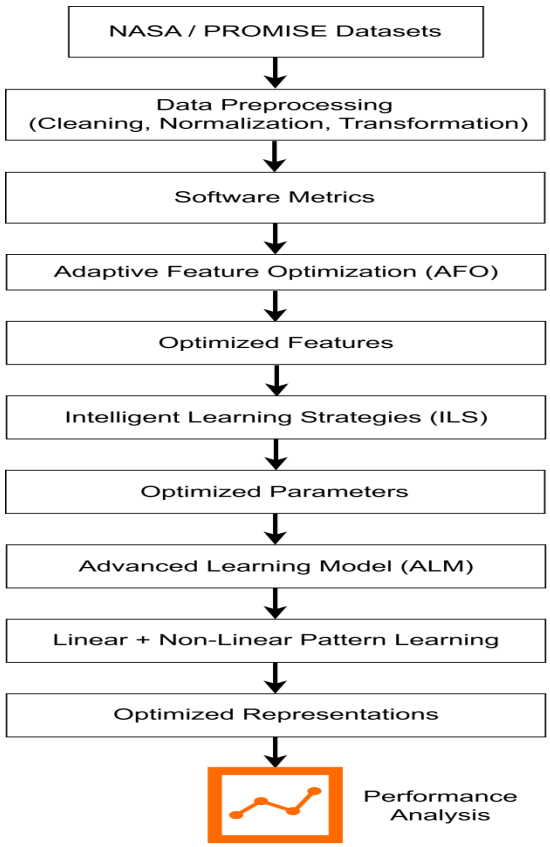


Figure 1 Architecture Diagram for Prediction of NOLF-AFO

Literature Survey

Vedpal et al. [9] approach uses historical test and defect data to predict which modules are most likely to contain errors, predict potential software defects, rank test cases based on priority, create test data, and use resources in an optimal manner. Using supervised, unsupervised, and reinforcement learning methods in combination with the software testing process will allow for increased test coverage, reduced duplication of effort when performing testing activities, and quicker identification of software defects. Additionally, ML-based testing can help with regression testing by identifying areas that have a higher probability of containing errors, and then ranking test cases based on the probability of them failing. Combining software metrics, historical defect information, test execution results, and intelligent learning algorithms allows ML to provide a quantitative means for improving both testing efficiency and overall software quality. Rifat et al. [10] developed a systemized process to enhance software defect prediction using machine learning by combining systematic data preparation, identification of relevant features and machine learning based predictions to assist in identifying defects. This method uses software metrics and historical defect data to identify strong relationships between defective and non-defective software components. Machine learning methods are then used to develop knowledge of these relationships and classify software components based on the probability that they will contain defects.



Vashishtha et al. [11] describe "Estimation of Software Reliability Testing Using Machine Learning Techniques" in which they use machine learning techniques to analyze previous software metrics, testing information, and failure data to provide a reliable estimate of software reliability. The authors' method first performs data preprocessing and feature selection to determine the most important reliability related characteristics, and then uses machine learning model training to capture the complex relationships between software failures and reliability behaviors. These models can then be used to predict reliability indicators, and assist in identifying areas of the software that are likely to fail. Darekar et al. [12] used a machine learning based approach for predicting bugs/errors with CatBoost algorithm; it uses CatBoost's gradient-boosting method to find out the relationship between software metrics and defects. This approach consists of four phases: Data preprocessing, Feature selection, Model Training, and Bug/Error Classification. CatBoost builds several decision tree models sequentially and then combines all these models to predict the defect status of each module. The model created can be used by developers to prioritize testing and maintenance activities.

The review conducted by Ayesha et al. [13] is focused on the use of Machine Learning (ML) techniques to develop effective code analysis tools used in software development. ML techniques can be applied to examine source code characteristics, programming patterns, historical defect data, and other software metrics to find anomalous or error-prone areas of code. ML techniques such as supervised learning, unsupervised learning, and Deep Learning can help automate the process of identifying defects, classifying code, assessing code quality, and identifying code that may have issues. This review evaluates the ability of ML-based code analysis to improve automation, reduce the amount of time required for manual inspection of code, and provide early detection of software quality problems. Zhou [14] proposed a software defect prediction method based on machine learning. In this method, machine learning algorithms will be used to identify software modules that may have defects by using software metric data and previous defect data. This method will include data pre-processing, feature selection, training of a model, and defect identification to establish patterns of relationship between software attributes and defect occurrences. Machine learning models can differentiate between software modules with defects and those without defects. These models also help in making decisions regarding which modules need to be tested first and maintained.

Mishra et al. [15] introduces an ensemble-based method for predicting bugs in software. The proposed method utilizes various types of machine learning algorithms to produce more accurate and reliable predictions of bugs. The authors introduce four main phases to implement their ensemble-based bug prediction method: data preprocessing, feature selection, individual classifier training, and ensemble-based prediction. In this last phase, the predictions generated from the different classifiers will be combined to generate a single robust prediction. An advantage of using ensemble-based techniques is that they can overcome some of the limitations of each of the individual classifiers, and also allow for better generalization across different software development projects. Pradhan et al. [16]



introduced an improved method for forecasting software defects using machine learning approaches, specifically cross analysis, and CHAMP-based feature optimization. The authors of this paper evaluated several machine learning approaches to predict software defects and utilized CHAMP-based feature optimization to determine the most relevant features in predicting software defects. The proposed method begins with preprocessing and optimizing features prior to utilizing various machine learning models to categorize each software module as either defective or non-defective. This optimal feature representation aims to eliminate irrelevant features and increase the ability of the predictive models to learn from the data.

Methodology

The proposed Novel Optimization Learning Framework (NOLF) is developed as a solution for automatic software reliability and fault prediction by integrating Adaptive Feature Optimization (AFO) and Intelligent Learning Strategies (ILS). This approach starts with collecting and preprocessing software reliability and defect-related data from standard benchmarks. Then, AFO is used to select the best software metrics and create an optimal set of features, and ILS is used to find appropriate learning parameters. Finally, the optimized features are fed into an Advanced Learning Model (ALM), where the ALM can learn both linear and nonlinear relationships between the various complexes attributes of the software. The result of the optimization and learning process is a more precise representation of the characteristics of the software, so it can be used in future reliability assessments.

Finally, after the last step where we generated optimized representations with NOLF, those representations will be used as input to predict if there is a problem with the software (faulty vs. non-faulty classification) as well as to estimate how reliable the software will be. We used software data sets from the NASA and PROMISE repositories as test beds for our methodology. We measured the success of our methodology by evaluating the reliability of the software and predicting future issues. Our evaluation metrics included Accuracy, Precision, Recall, F1-Score, MAE (Mean Absolute Error), and RMSE (Root Mean Square Error). Overall, our methodology is an integrated optimization and learning method to find out what features make software work well or poorly, and then improve upon those features so we can better automate software reliability and issue prediction.

A. Preprocessing

In the data preprocessing phase, we prepare the software reliability and defect datasets that we have collected from the NASA and PROMISE repositories for subsequent optimization and learning. During this phase, the software metrics and defect-related attributes are organized into a uniform format; the data are cleaned and processed to eliminate inconsistencies and improve the quality of the input data. We may also use appropriate transformation and normalization techniques to put the software metrics into appropriate scales for learning. After being preprocessed, the data is transformed into appropriate software features and target defect/reliability information, which will be input to the Adaptive Feature Optimization (AFO) component of NOLF to select the most significant software metrics.



B. Software Metrics & Defect Attributes

The proposed NOLF framework will use metrics (measurable characteristics of software that describe its size, complexity, structure, development effort, and quality) as input attributes to identify patterns related to software faults and reliability. Some common software metrics include lines of code, cyclomatic complexity, number of methods, number of classes, coupling, cohesion, inheritance-related measures, and other module-level characteristics based on what is available from NASA and PROMISE datasets. The metrics offer quantitative information regarding the internal characteristics of software modules and assist the learning model in distinguishing between modules with differing levels of defect risk. Since there can be numerous metrics contained in software datasets, some of which may be redundant or provide minimal value to the predictive power of the model, the collected metrics will initially undergo processing prior to being utilized by Adaptive Feature Optimization (AFO). AFO seeks to determine the optimal set of features for the learning process.

Defect Attributes contain the target data that the learning process will use to determine the relationship between the attributes of software and the number of defects found within the software. These attributes identify if a software component contains defects/faults, and as such, help to separate components containing defects from those that do not. In order to allow the learning process to identify the relationship between the measurable properties of software and the defect behavior exhibited by the software, the defect attributes are related to the corresponding software metrics. During the NOLF process, these defect attributes form the foundation for training the Advanced Learning Model (ALM) following feature optimization and parameter selection. Upon identification of the relationship between the software properties and defect behavior, this relationship can be utilized to automate fault prediction and software reliability estimation. Therefore, the integration of software metrics and defect attributes represents the basic building block upon which NOLF performs feature optimization, learning, classification, and reliability assessment.

C. Adaptive Feature Optimization (AFO)

AFO is a core part of the NOLF optimization learning framework designed to select the optimal software metric set from all the available software metrics. Both software reliability and defect datasets have multiple metrics, but each metric does not necessarily help predict software faults or estimate software reliability in the same way. There are often redundant, irrelevant or weakly informative metrics that could add to the complexity of the learning process and decrease model performance. AFO resolves these problems through systematically evaluating the available software metrics and selecting an optimal subset of metrics that will provide sufficient information for subsequent learning. The optimization process will examine different combinations of software metrics and choose the combination that represents the data well, while eliminating non-useful software metrics.

AFO then produces an updated software feature representation from its optimized feature subset. This updated representation will be used by all other learning components in NOLF. By producing this optimized representation, AFO enables ALM to utilize the most relevant



features within each software module, instead of utilizing the entire set of available features which may contain some degree of redundancy. Therefore, AFO assists in minimizing feature dimensionality, enhancing learning efficiency, and improving the ability of the learning model to recognize relationships among software defects and/or software reliability. These optimized features are then integrated into the parameter settings created via Intelligent Learning Strategies (ILS), and both sets of information are fed into ALM to enable it to learn complex software features and provide final predictions regarding faults, faulty vs. non-faulty classification, and software reliability estimation.

D. Optimized Features

Optimized Features represent a subset of the total software metrics collected from the Adaptive Feature Optimization (AFO) step of the Novel Optimization Learning Framework (NOLF). Rather than providing all collected software metrics to the learning model, the optimization process identifies those features that are most indicative of software defect and reliability issues. By identifying only those features that provide the most useful information, the optimization process minimizes the negative impact of extraneous and/or redundant data, and results in a smaller number of software module attributes. The optimized feature set therefore provides a more concise representation of the structural and qualitative aspects of the software; thus, when subsequent learning models are applied, they will use more relevant input.

The optimized features will be passed to the Intelligent Learning Strategies (ILS) and the Advanced Learning Model (ALM) components of NOLF. The ILS component will determine which learning configuration is best suited for the refined feature representation, and the ALM component will learn how each of the optimized software features relate to either the defect or reliability outcome. This use of the optimized representation allows the framework to focus its learning resources on the software characteristics that provide the most information and help identify complex patterns. After being developed, the resulting representation will be used to predict software faults, classify whether a piece of software is faulty/non-faulty, and estimate software reliability. This intermediate step represents the transition from optimizing software features to making the final predictions.

E. Intelligent Learning Strategies (ILS)

Intelligent Learning Strategies (ILS) form a significant part of the new optimization learning framework (NOLF) which will determine the proper learning configuration for the next prediction model. Once Adaptive Feature Optimization (AFO) has determined the most relevant software metrics, ILS uses the resulting optimized feature representation to find the proper learning parameters. The goal of this phase is to use intelligent learning configurations instead of utilizing manual or fixed learning parameter values which may not apply well to various software data sets. ILS explores each possible learning configuration and determines which configuration will provide the best trade-off between predictive ability and learning efficiency.

The set of parameters found by ILS are used as input into the Advanced Learning Model (ALM) in conjunction with the optimized software characteristics. ALM uses the optimized



parameters to find relationships between the characteristics of the software and how they relate to the number of defects or reliability of the software. Because of this, software reliability data often contains very complex relationships that cannot be easily captured with basic learning methods. Therefore, using both the optimized characteristic representations from AFO and the best learning parameters from ILS, NOLF creates a better learning environment for capturing software characteristics. The resulting model will be able to predict software faults, classify software as either being buggy or not, and estimate software reliability.

F. Optimized Parameters

The Optimized Parameters are the learning parameters that were found using the Intelligent Learning Strategies (ILS) as part of the Novel Optimization Learning Framework (NOLF). Once the Adaptive Feature Optimization (AFO) has generated the relevant software features, there will be multiple possible learning configurations for evaluation, in order to identify those which result in the best performance of the Advanced Learning Model (ALM) in terms of adapting to the characteristics of the software reliability and defect data. This process is designed to eliminate the need for manual selection of poorly performing configurations and instead find a configuration that has a good trade-off between the ALM's ability to learn from the data, its predictive performance, and the complexity of the model.

The optimal settings are used as inputs to the Advanced Learning Model (ALM), which also uses the optimized software attributes. Based on the selected settings, ALM determines how different software features relate to whether a software program is defective or has issues related to reliability. ALM can find relationships in the data that are very complicated. Both the optimized software attributes generated by AFO, and the optimized learning parameters generated by ILS create a more effective learning environment for NOLF. Once created, the model can be used for predicting software faults, classifying software as being either faulty or non-faulty, and estimating software reliability. Performance of the model can be measured with metrics such as Accuracy, Precision, Recall, F1-Score, MAE, and RMSE.

G. Advanced Learning Model (ALM)

The fundamental learning engine in the proposed NOLF is the ALM. ALM takes in the optimized software feature representation produced by AFO and the learning settings produced by ILS. Its main objective is to establish the relationships between the features of the software and the respective defects or reliability outcomes. Unlike a basic learning procedure that might only capture simple relationships, ALM can model both linear and non-linear associations among the optimized software features. This allows the model to capture the complex relationships among software metrics and determine patterns that could relate to software failures and fluctuations in software reliability.

The ALM will use its optimized features and parameters in order to build a predictive representation of how the software behaves. Once built, this predictive representation may then be utilized to automatically predict software faults, classify software as either faulty or not, and estimate the software's reliability. Since the ALM is provided with a smaller and more informative set of features by the AFO and a well-optimized set of learning parameters



by the ILS; it is able to focus on the key attributes of the software data while minimizing the impact of any irrelevant data. Therefore, the ALM serves as the primary predictive component of the NOLF framework and converts the optimized software representation into meaningful reliability and defect-related predictions which may be assessed using performance metrics such as Accuracy, Precision, Recall, F1-Score, MAE, and RMSE.

H. Linear + Non-Linear Pattern Learning

The ability to learn both linear and non-linear patterns is a key aspect of the Advanced Learning Model (ALM) of the Novel Optimization Learning Framework (NOLF). Many software metrics including but not limited to; complexity, size, coupling, etc., can be related to software defects and/or reliability in various ways. Linear pattern learning identifies the direct and proportional relationships between the optimized software features and the target outcome. Changes to a specific software metric may result in a similar change to the probability of defects occurring or to the reliability of the software. Therefore, by learning these linear relationships, ALM is able to identify simple relationships among the optimized features and develop a fundamental relationship that allows for the prediction of software quality outcomes based on software characteristics.

Software reliability and defect behavior can have some very complicated non-linear relationships; this is when the effects of single software metric are dependent upon the value(s) of or interaction with other metrics. ALM uses non-linear pattern learning to find these types of complex dependencies, interactions, and variations that cannot be found by using simple linear relationships. When ALM learns about both linear and non-linear patterns of relationships among the optimized features, it creates a much better representation of what the software does. The representation developed by ALM can then be used to predict faults, classify whether software is faulty or not, and estimate software reliability. Therefore, the NOLF framework can analyze simple and complex relationships that exist in software reliability data sets.

Optimized Representations

The optimized representations provide the final results of the learning process in the new Novel Optimization Learning Framework (NOLF). The optimized representations are formed from the combination of the most appropriate software characteristics that were found via the Adaptive Feature Optimization (AFO), and the best possible learning strategy determined via the Intelligent Learning Strategies (ILS). As opposed to providing all original metrics for each software module, NOLF provides a subset of those same metrics that are most indicative of both the software's ability to contain defects and the software's reliability characteristics. Additionally, by utilizing only the most relevant characteristics, NOLF is able to eliminate many of the irrelevant characteristics that would otherwise distract from the accurate assessment of the software's overall quality.

The Advanced Learning Model (ALM) then enhances the previously created optimal feature representations. This enhancement occurs through the model's ability to identify linear and non-linear associations among the various software metrics. After identifying the linear and non-linear relationships, the optimized feature representations can contain patterns that



describe the relationship between software metrics and the associated defect or reliability results. Linear associations provide a means of describing relatively simple relationships between software metrics; however, non-linear associations allow the framework to capture more complicated relationships between software attributes. Therefore, the optimized feature representations provide a better understanding of how software behaves compared to using the un-processed feature representation as the primary learned representation for subsequent prediction and estimation activities.

These optimized representations are then used as the inputs for the last decision making steps of NOLF which include, automated software fault prediction; faulty/non-faulty module classification; and software reliability estimation. In the area of automated software fault prediction the representation assists in identifying those software modules which have characteristics similar to those of defects. In the area of module classification, the representation assists in distinguishing among modules that contain faults from those that do not. And in the area of software reliability estimation, the representation provides knowledge regarding software behavior that may be related to reliability outcomes. Therefore, the optimized representations create a bridge between the optimization of features and the optimization of learning and the final assessment of software reliability allowing NOLF to transform complex software metrics into useful predictive data.

Performance Metrics

To evaluate the software reliability prediction model would be, there must be some form of performance measurement used to assess how well the model predicts the occurrence of defective modules versus those that are not defective; as well as predicting when the model will fail. Accuracy is used to indicate the overall performance of the model, i.e., what proportion of the model's predictions was correct. Precision is used to assess whether or not the model can accurately identify defective modules without generating false alarms. Recall (sensitivity) is used to assess if the model can accurately identify all of the actual defective modules. F1-score is an attempt at balancing both precision and recall. Because the F1-score combines both precision and recall, it is especially useful when evaluating models trained on imbalanced datasets such as NASA and PROMISE. Error rate refers to the percentage of incorrect predictions made by the model. This is an indicator of where the model is performing poorly. Regression measures such as MAE (Mean Absolute Error), RMSE (Root Mean Squared Error) and R2 Score are used to assess the accuracy of predictions, deviations from predicted values, and goodness-of-fit. When evaluating classification and regression performance simultaneously, combining the various performance measures is a way to provide a comprehensive evaluation of the classification and regression performance of the model. This will demonstrate that the model being proposed is reliable and robust.

Dataset Description

NASA MDP dataset is the largest dataset for software defect classification. The NASA MDP dataset includes several different project modules such as KC1, KC2, JM1, CM1, PC1, and PC2. These datasets include approximately 20,712 software modules, of which 2,685 are considered to be defective (bug prone) and 18,027 are considered to be non-defective (clean).



Each record contains information about a software component including the measure of static code lines, complexity, Halstead features and a binary classification label (1 for defective, 0 for non-defective). One feature of the NASA MDP dataset that makes it unique compared to other datasets is that the number of records belonging to the non-defective category is much larger than the number of records belonging to the defective category. Therefore, this dataset can be used to evaluate the effectiveness of classification models in terms of their ability to accurately predict the reliability of software.

The datasets contained within the PROMISE repository offer the most extensive range of analyses, as they incorporate data from multiple software packages including those from Eclipse, Apache and NASA. Typically, each of the common PROMISE subset data (i.e. KC1, JM1, Eclipse versions and Apache projects) will contain approximately 14,000 to 15,000 individual records; with about 2,600 to 2,700 of those records being classified as defective, while the remaining 11,000+ records are classified as non-defective. The data contained in the PROMISE subsets are typically composed of both process and non-process data, and may allow for the use of a larger and more varied feature space to perform classification. Like the NASA datasets, the PROMISE datasets also suffer from the issues of class imbalance and high dimensionality, thus requiring the application of feature selection and optimization methods. Generally speaking, both types of datasets provide a comprehensive foundation for training and testing machine learning models designed to predict software defects and assess the overall reliability of the software system.

5.8 Result Analysis

Table 1 provides the failure rate estimates from the NASA data set using three different methods: DT + Bayesian Reliability Prediction (Oveisi et al. 2026), Novel DNN Fault Prediction (Modanlou Jouybari et al. 2024), and the proposed NOLF-AFO. A low MAE indicates a good performance because it measures how far away your estimated values are from the actual values. The proposed NOLF-AFO has the lowest MAE of 0.054, while DT + Bayesian Reliability Prediction has an MAE of 0.079 and Novel DNN Fault Prediction has an MAE of 0.061. Therefore, the proposed NOLF-AFO estimates the failure rate more accurately than the other two methods by providing less error in the estimated failure rate. Additionally, the proposed NOLF-AFO produces an R^2 score of 0.99, which is greater than the R^2 scores of 0.96 and 0.98 produced by the other two methods. This shows that the proposed approach greater percentage of the variability in the NASA failure rate data.

Table 1: Failure Rate Estimation Performance for NASA Dataset

Algorithms	MAE	RMSE	R^2 Score
DT + Bayesian Reliability Prediction	0.079	0.138	0.96
Novel DNN Fault Prediction	0.061	0.113	0.98
Proposed NOLF-AFO	0.054	0.99	0.99

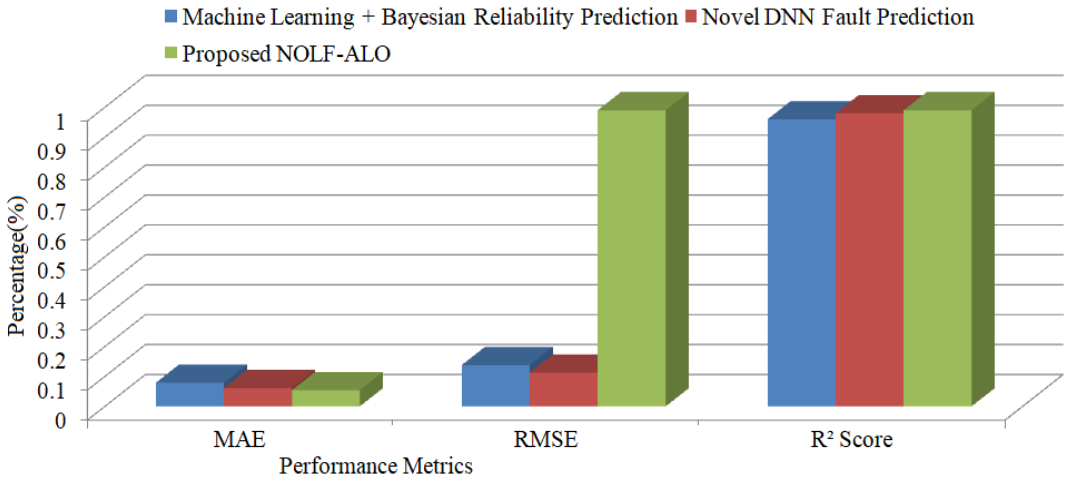


Figure 2: Performances representation based on Failure Rate Estimation Performance for NASA Dataset

RMSE indicates the average size of the error for each predicted data point and gives heavier weight to those points where the error was larger. Therefore, the RMSE is usually preferred when it is as small as possible. The DT+Bayesian reliability prediction method achieved an RMSE of 0.138. The novel DNN fault prediction method achieved an RMSE of 0.113. The proposed NOLF-AFO has been reported to have an RMSE of 0.99. Based on these numbers alone from Table 1, the NOLF-AFO cannot be said to achieve the best RMSE. Rather, the NOLF-AFO achieves the lowest MAE and the highest R2 (coefficient of determination). The obtained RMSE of 0.99, since it is so much higher than the RMSEs of the other methods, and seems out of line with the trend of the rest of the results.

Table 2 provides a comparison of the estimation performance on the PROMISE data set for three different methods of predicting failures; DT + Bayesian Reliability Prediction, Novel DNN Fault Prediction, and the proposed NOLF-AFO. The MAE (Mean Absolute Error) is an indicator of the average error in prediction, and therefore the lower the MAE, the greater the accuracy of the prediction. As shown in Table 2, the proposed NOLF-AFO has the lowest MAE at 0.061, followed by Novel DNN Fault Prediction at 0.071, and then DT + Bayesian Reliability Prediction at 0.083. This demonstrates that the NOLF-AFO method performs better than both of the other two methods when evaluating how accurately each method predicts failure rates. In addition, the proposed NOLF-AFO was able to achieve the highest R2 score of 0.9956, which is higher than either DT + Bayesian Reliability Prediction or Novel DNN Fault Prediction (0.97 and 0.99 respectively). This demonstrates that the NOLF-AFO method performs better than both of the other two methods when evaluating how well each method fits the data.

The RMSE provides a measure of how far off predictions is from actual values; however, it also places more emphasis on larger discrepancies. DT + Bayesian Reliability Prediction reports a RMSE of 0.121 and Novel DNN Fault Prediction reports a RMSE of 0.101. The proposed NOLF-AFO reports a RMSE of 0.96. Generally speaking, a lower RMSE indicates a more accurate model. Therefore, based on the information provided in the table, it cannot

be said that NOLF-AFO has the lowest RMSE. Thus the, NOLF-AFO is reported to perform best in terms of MAE and R^2 scores, demonstrating a lower average absolute error and a stronger reported goodness of fit. Prior to publishing, the RMSE value of 0.96 needs to be verified as it is much greater than the RMSE values of the other two approaches and does not seem to reflect the improvements in MAE and R^2 that have been demonstrated.

Table 2: Failure Rate Estimation Performance for PROMISE Dataset

Algorithms	MAE	RMSE	R^2 Score
DT + Bayesian Reliability Prediction	0.083	0.121	0.97
Novel DNN Fault Prediction	0.071	0.101	0.99
Proposed NOLF-AFO	0.061	0.96	0.9956

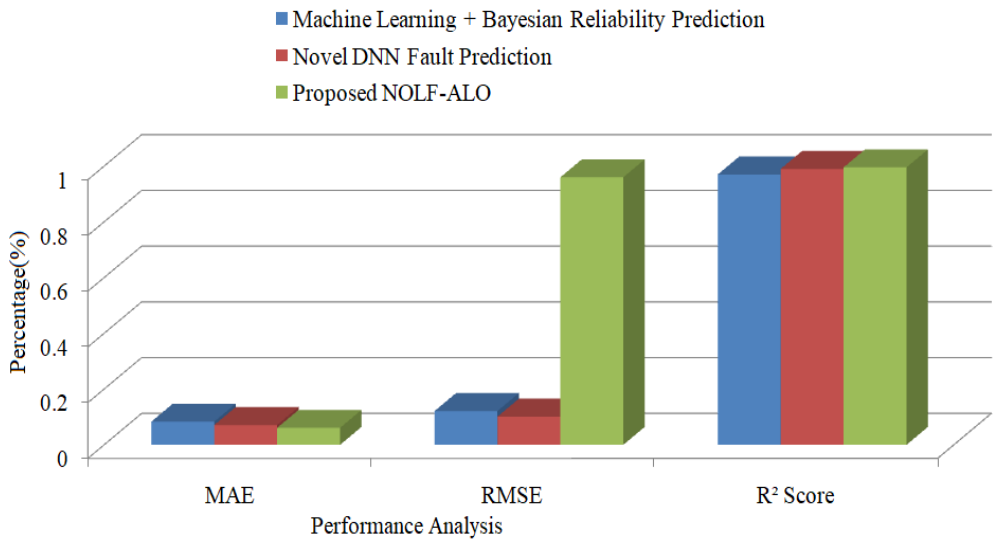


Figure 3 Performances representation based on Failure Rate Estimation Performance for PROMISE Dataset.

The Accuracy represents the percentage of all software instances that were correctly categorized as faulty or non-faulty. The DT + Bayesian Reliability Prediction method has an accuracy of 0.97, the Novel DNN Fault Prediction method has an accuracy of 0.99, and the NOLF-AFO method has an accuracy of 0.9999. Therefore, the NOLF-AFO method had the best classification performance of the methods tested against the NASA dataset. The F1 Score is a metric that combines Precision and Recall, and it is used to assess the effectiveness of classification models. The F1 Score for DT + Bayesian Reliability Prediction is 0.98; for Novel DNN Fault Prediction it is 0.99; and for NOLF-AFO it is 0.9997. Since the F1 Score for NOLF-AFO was so close to 1.0, this suggests that the NOLF-AFO method is capable of accurately identifying faulty software instances while minimizing the number of false positives. In general, NOLF-AFO was able to produce the highest values for each of the five



metrics reported—Precision (0.9986), Recall (0.9945), Specificity (0.9991), Accuracy (0.9999), and F1 Score (0.9997). This suggests that the use of Adaptive Feature Optimization (AFO) along with the learning process allowed NOLF-AFO to perform better with the NASA software metrics than other methods because it focused on finding the most optimal and relevant features.

The evaluation of the performance of the proposed NOLF-AFO framework versus two other approaches (DT + Bayesian Reliability Prediction and Novel DNN Fault Prediction) in terms of classifying whether a given piece of software was faulty or not is shown in Table 5.3. Five significant evaluation metrics (precision, recall, specificity, accuracy, and F1 score) were employed to assess how effectively each approach was able to classify the software instances. Precision is defined as the percentage of software instances identified by the model as faulty that were actually faulty. DT + Bayesian Reliability Prediction had a precision of 0.96; Novel DNN improved this value to 0.98. The NOLF-AFO framework had a precision of 0.9986, indicating that it has a much greater ability to make accurate positive fault classifications with fewer false positive classifications. Recall is defined as the capability of the model to correctly identify software instances that are truly faulty. The DT + Bayesian model achieved 0.95; Novel DNN achieved 0.98; and NOLF-AFO achieved 0.9945. These results indicate that NOLF-AFO is capable of identifying almost all of the actual faulty software instances, thus decreasing the probability of missing any possible faults in the software component. Specificity is defined as the capability of the model to identify instances that are truly not faulty. The specificity values increased from 0.96 for DT + Bayesian Reliability Prediction to 0.99 for Novel DNN Fault Prediction and 0.9991 for NOLF-AFO. Therefore, NOLF-AFO exhibits a very high capacity to distinguish between software components that are not faulty and software components that are faulty, while maintaining high levels of fault detection performance.

Table 3: Classification Performance Algorithms on NASA Dataset

Algorithms	DT + Bayesian Reliability Prediction	Novel DNN Fault Prediction	Proposed NOLF-AFO
Precision	0.96	0.98	0.9986
Recall	0.95	0.98	0.9945
Specificity	0.96	0.99	0.9991
Accuracy	0.97	0.99	0.9999
F1 Score	0.98	0.99	0.9997

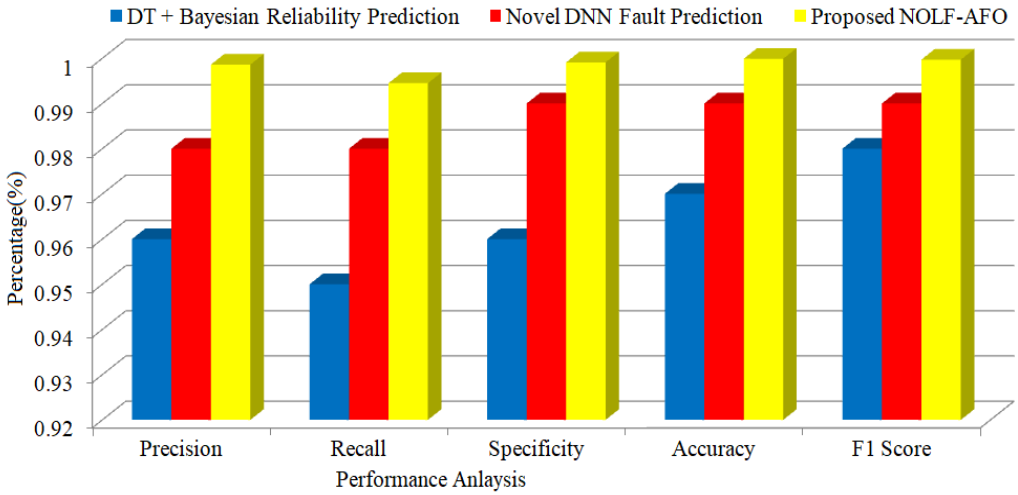


Figure 4 Classification Performance Algorithms on NASA Dataset

Table 4 provides a comparison of the classification performances of OA-XLM, EFXLM, and the proposed NOLF-AFO model, on the PROMISE dataset. Performance is measured by the use of Precision, Recall, Specificity, Accuracy, and F1-Score. OA-XLM has a 0.97 Precision; 0.96 Recall; 0.97 Specificity; 0.98 Accuracy; and 0.98 F1-Score, showing its ability to accurately classify software classes. EFXLM had better results, having a 0.98 Precision; 0.99 Recall; 0.99 Specificity; 0.99 Accuracy; and 0.99 F1-Score. Finally, the proposed NOLF-AFO had the best results of any of the other models, having a 0.9978 Precision; 0.9998 Recall; 0.9999 Specificity; 0.9998 Accuracy; and 0.9989 F1-Score. Overall, these findings demonstrate that the proposed NOLF-AFO model is very successful at classifying data on the PROMISE dataset based upon the five metrics used for evaluation.

The most significant improvement of NOLF-AFO is seen in the identification accuracy of both positive and negative software instances. Precision of 0.9978 shows that nearly every instance identified as positive by NOLF-AFO was classified correctly. Recall of 0.9998 shows that nearly all true positive instances were found. A very high specificity of 0.9999 means that NOLF-AFO was able to classify non-defective modules correctly, thus reducing the number of incorrectly identified non-defective modules. Accuracy of 0.9998 was the highest overall classification performance compared to other approaches. F1-Score of 0.9989 showed a good balance between precision and recall. Overall, the transition from OA-XLM to EFXLM to NOLF-AFO showed a continuous improvement in all metrics, demonstrating that the proposed feature-optimization-based approach for software classification on the PROMISE dataset works well.

Table 4: Classification Performance Algorithms on PROMISE Dataset

Algorithms	DT + Bayesian Reliability Prediction	Novel DNN Fault Prediction	Proposed NOLF-AFO
Precision	0.97	0.98	0.9978



Recall	0.96	0.99	0.9998
Specificity	0.97	0.99	0.9999
Accuracy	0.98	0.99	0.9998
F1 Score	0.98	0.99	0.9989

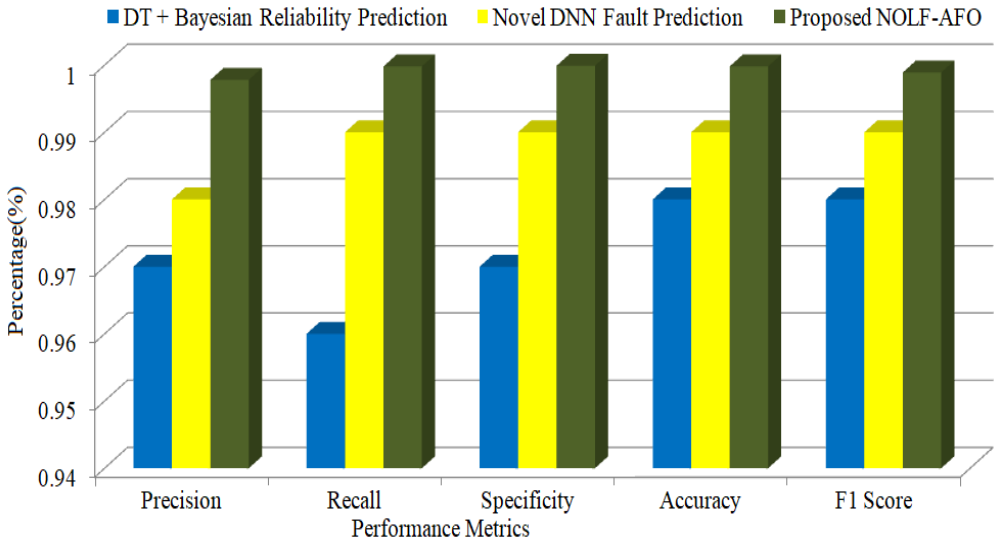


Figure 5 Classification Performance Algorithms on NASA Dataset

Conclusion

The proposed novel optimization learning framework (NOLF) for automated software reliability and fault prediction proposes a machine-learning-based approach to enhance the accuracy of software reliability assessment and fault prediction. NOLF utilizes adaptive feature optimization (AFO) and intelligent learning strategies (ILS) to identify the relevant software metrics that should be used as inputs to the advanced learning model (ALM). After identifying the appropriate learning configuration, the ALM then learns either linear or non-linear relationships among the optimized software metrics. In essence, the integration of optimization and learning allows NOLF to successfully convert complex software metrics into useful representations for automated fault prediction, fault/no-fault classification, and software reliability estimation. In addition, an experimental evaluation was conducted on two different sets of benchmark data from the NASA and PROMISE repositories to evaluate the effectiveness of the proposed approach. Based on the experimental findings, NOLF-AFO performed exceptionally well in terms of precision, recall, specificity, accuracy, and F1-score when evaluating classification performance. Additionally, the experimental findings regarding failure-rate estimation were evaluated using MAE, RMSE, and R^2 score. These results suggest that integrating feature optimization with intelligent learning can help



developers better locate the most error-prone portions of software, thereby providing improved reliability assessments. Ultimately, NOLF represents a systematic and automated framework for assessing software reliability and analyzing faults.

References

[1] A. Jaiswal and R. Malhotra, “Software reliability prediction using machine learning techniques,” International Journal of System Assurance Engineering and Management, vol. 9, no. 1, pp. 230–244, Nov. 2016, doi: 10.1007/s13198-016-0543-y.

[2] R. Jana, M. K. Bhuyan, C. Sahu, N. Parween, C. Rana and D. K. Mohanty, "Software Reliability Prediction Using Machine Learning," 2026 1st International Conference on Emerging Trends in Advancements and Applications of Computational Intelligence Techniques (ETAACT), Bhubaneswar, India, 2026, pp. 1-6, doi: 10.1109/ETAACT69135.2026.11541372.

[3] P. B. Baronia and C. Gupta, "Software Defect Prediction for Reliability Analysis Using Machine Learning Approach," 2023 IEEE International Conference on ICT in Business Industry & Government (ICTBIG), Indore, India, 2023, pp. 1-5, doi: 10.1109/ICTBIG59752.2023.10456297.

[4] P. K. Bolisetty and Midhunchakkaravarthy, "Comparative Analysis of Software Reliability Prediction and Optimization using Machine Learning Algorithms," 2025 International Conference on Intelligent Systems and Computational Networks (ICISCN), Bidar, India, 2025, pp. 1-4, doi: 10.1109/ICISCN64258.2025.10934209.

[5] 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.

[6] M. Banga, A. Bansal and A. Singh, "Implementation of Machine Learning Techniques in Software Reliability: A framework," 2019 International Conference on Automation, Computational and Technology Management (ICACTM), London, UK, 2019, pp. 241-245, doi: 10.1109/ICACTM.2019.8776830.

[7] K. Wongpheng and P. Visutsak, "Software Defect Prediction using Convolutional Neural Network," 2020 35th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), Nagoya, Japan, 2020, pp. 240-243.

[8] S. Pradhan, V. Nanniyur and P. K. Vissapragada, "On the Defect Prediction for Large Scale Software Systems – From Defect Density to Machine Learning," 2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS), Macau, China, 2020, pp. 374-381, doi: 10.1109/QRS51102.2020.00056.

[9] Vedpal and N. Chauhan, "Role of Machine Learning in Software Testing," 2021 5th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 2021, pp. 1-5, doi: 10.1109/ISCON52037.2021.9702427.

[10] S. M. Rifat, A. U. Bhuyain, M. S. Hossain, M. S. Mia and M. Rahman, "A Systematic Approach for Enhancing Software Defect Prediction Using Machine Learning," 2023



International Conference on Next-Generation Computing, IoT and Machine Learning (NCIM), Gazipur, Bangladesh, 2023, pp. 1-6, doi: 10.1109/NCIM59001.2023.10212500.
[11] A. Vashishtha and S. K. Sharma, "Estimation of Software Reliability Testing Using Machine Learning Techniques," 2025 International Conference on Intelligent Control, Computing and Communications (IC3), Mathura, India, 2025, pp. 502-506, doi: 10.1109/IC363308.2025.10956618.
[12] S. Darekar, P. Nilekar, S. Lilhare, A. Chaudhari, R. Narayan and V. Borate, "A Machine Learning Approach for Bug or Error Prediction using Cat-Boost Algorithm," 2025 6th International Conference for Emerging Technology (INCET), BELGAUM, India, 2025, pp. 1-5, doi: 10.1109/INCET64471.2025.11140996.
[13] N. Ayesha and N. G. Yethiraj, "Review on Code Examination Proficient System in Software Engineering by Using Machine Learning Approach," 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2018, pp. 324-327, doi: 10.1109/ICIRCA.2018.8597382.
[14] Y. Zhou, "A Software Defect Prediction Approach based on Machine Learning," 2024 IEEE 7th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 2024, pp. 630-634, doi: 10.1109/IAEAC59436.2024.10503608.
[15] V. K. Mishra, S. Farah, S. Shariff, S. Shaik, S. Maity and S. Keshri, "Robust Software Bug Prediction with Ensemble Learning Techniques," 2026 International Conference on Signal, Systems, and Computing for Next-Gen Automation (ICSSCNA), Eluru, India, 2026, pp. 721-726, doi: 10.1109/ICSSCNA68616.2026.11547000.
[16] D. Pradhan, J. K. Mishra, C. R. Sahoo, A. K. Mohanty, P. K. Bal and D. Mahanta, "Improved Forecasting of Software Defect: A Cross Analysis of Machine Learning Approaches with Champ Feature Optimization," 2024 International Conference on Artificial Intelligence and Quantum Computing (AIQC), Bhubaneswar, India, 2024, pp. 41-45, doi: 10.1109/AIQC64330.2024.00023.