DEEP LEARNING BASED SOFTWARE RELIABILITY PREDICTION MODEL FOR COMPONENT BASED SOFTWARE

SHIVANI YADAV[1], BALKISHAN[2]

[1]Research Scholar, Department of Computer Science & Applications, Maharshi Dayanand University, India

[2]Assistant Professor, Department of Computer Science & Applications, Maharshi Dayanand University, India

E-mail: [1]shivaniyadav17@gmail.com, [2]balkishan248@gmail.com

ABSTRACT

Component Based Software Engineering (CBSE) is a software development approach used for rapid development. This involves the use of existing, mature components, which might have been developed for other software or product, along with new components to speed up the software development process. Component based development is being increasingly adopted to reduce costs, manage the increasing complexity of software products, and faster time to market. A major challenge posed by the adoption of Component Based Software (CBS) is to ensure the reliability of software products developed using this approach. The reliability of CBS is dependent upon the individual component reliability as well as software architecture. In this paper, a deep learning-based model is designed to predict the reliability of components. The results show that the designed model is well predicting the reliability with high accuracy. Comparison with other machine learning models justified the use of deep learning models in reliability prediction.

Keywords: *CBS, Component, Deep Learning, Prediction, Reliability*

1. INTRODUCTION

Software engineering can be defined as the process of designing, developing, and testing software programs and products. All the phases of software engineering are required every time a new product is developed. This means that all the activities are to be recurring, even if there is considerable overlap between product functionality. To overcome this duplication of efforts and long development cycles, concepts like code reusability were developed. This led to the development of Component-Based Software Engineering (CBSE) which involves the development of reusable components that could be used across different software products by configuring interactions and dependencies between components [[1]]. CBSE is used to fast-track the software development process by using existing software components. A component can be thought of as a part of the software that is used to apply logic and can interact with other components using interfaces for data exchange. This allows for the reusability of components resulting in low costs, higher quality, and shorter development time [**Error! Reference source not found.**]. The widespread use of components for software development led to concerns reliability of software. Software reliability is a key parameter that assists developers in planning adequate testing activities before developing new software by quantifying the failure behavior of a software system. CBSE is extremely beneficial in increasing productivity. However, to maintain the quality, the reusable software components must be reliable. The challenges faced in component selection from component repository include time, performance, fault tolerance, compatibility of components, reliability, functionality, and available component subset [3]. Among all the challenges, the main focus of this study is on the reliability of components.

The current study focuses on the use of computational intelligence techniques for predicting the reliability of software components. This research work focuses on predicting reliability of individual components using deep learning. As, it is concluded from the previous work that reliability model based on components is very rare. Several techniques like genetic algorithms, machine learning, and deep learning have been used by different researchers in various studies. This paper aims to design a model for predicting the reliability of software components using deep learning based fully connected neural network model on different datasets.

The rest of the paper is structured into four more sections. Section 2 discusses the existing studies on the software reliability prediction models using computational intelligence techniques for software reliability prediction. The fully connected neural network based on deep learning with results is presented in section 3 while the comparison of the model with other machine learning models is given in section 4 and finally, section 5 concludes the paper along with the future scope.

2. BACKGROUND STUDY

Computational Intelligence techniques are popularized by Zadeh [[4]] in 1994 for modeling imprecision and uncertainty to determine robustness and tractability with low cost for software reliability estimation. In 2005, Madsen et al. [[5]] proposed a method for supporting evolutionary computing, data mining techniques, and fuzzy approaches for software reliability modeling. Kamei et al. [[6]] performed a comparison of several computational intelligence approaches in terms of software failure prediction. The authors evaluate the performance of an SVM-based prediction approach against that of other methods like neural networks, classification trees, linear discriminant analysis, and logistic regression. The analysis reveals that the SVM-based software fault prediction model outperforms other traditional approaches. In 2010, Marcia et al. analyzed different computational intelligence techniques like fuzzy systems, neural networks, and stochastic methods which help in reliability modeling intending to improve reliability. CBS study on reliability majorly includes component interaction, dependency, reusability, failure, etc. In 2006, Pai and Hong [[7]] examined existing literature on reliability prediction and proposed the use of Support Vector Machines (SVM) for the same. The authors used Simulated Annealing (SA) technique to determine SVM constraints and compared the results with numerous other models. The proposed model showed better prediction capabilities due to the usage of SA algorithms in parameter selection. Sharma [**Error! Reference source not found.**] modeled a metric for interface complexity between different components. This was done by studying the properties of different components, including the methods used, arguments, and return types. The complexity metric was determined for components in Java Beans. Subsequently, the metric was evaluated against component metrics like readability, customizability, and execution time. The researchers concluded that components scoring higher on the complexity metric took more effort to maintain and more time to execute. Zheng [[8]] proposed a non-parametric model which used an ensemble of neural networks for reliability prediction. The model was then evaluated on two datasets and compared with a single neural network and three other parametric prediction models. Prediction accuracy was higher for non-parametric models as compared to parametric models. Further, the ensemble of neural networks performs better than a single neural network. Singh and Kumar [[9]] described a feed-forward neural network-based reliability prediction model. The proposed model used failure history as input and used back-propagation to adjust network weights. The proposed model was evaluated on seven data sets containing failure information and the results were compared with three other reliability prediction models. It was found that the proposed artificial neural network model yielded better results than conventional parametric models. Tyagi and Sharma [[10]] suggested that soft computing techniques like neural networks and fuzzy logic are better suited for the reliability prediction of Component-Based Software Systems (CBSS) since real-world reliability is difficult to predict. They proposed an Adaptive Neuro Fuzzy Inference System (ANFIS) model and compared its reliability prediction ability with a Fuzzy Inference System (FIS). The result showed that ANFIS has a higher prediction ability, based on Root Mean Square Error (RMSE) values. The proposed model is also computationally efficient as compared to the FIS model. However, the model’s limitation was the fact that there were only four factors considered since the datasets were obtained from classroom-based projects. Diwaker and Tomar [[11]] analyzed the importance of component interaction and the path used for component reuse. They used Ant Colony Optimization (ACO) for determining the optimal path between components that increases reusability and reliability. The results showed that component efficiency and dependency value increased with the application of ACO. Sharma and Gandhi [[12]] described the application of the Modified Neuro-Fuzzy Inference System (MNFIS) approach for CBS reliability prediction. The model uses the advantages of neural network and fuzzy logic and takes four factors into account, i.e., component reusability, operational profile, dependency, and fault density. The results were compared with FIS, and it was found that MNFIS had better reliability prediction based on sensitivity analysis. Kulamala et al. [[13]] studied computational intelligence techniques like

SVM, MLP, Bagging, Feed Forward Back Propagation Neural Network, Cascade Forward Back Propagation Neural Network, Adaptive Neuro Fuzzy Inference System, Generalized Regression Neural Network, Regression, Neural Network ensembles, Non-Homogeneous Poisson Process, Instance-based learning, Radial-Bias Neural Networks, etc. for software reliability prediction. These techniques are surveyed, applied, and evaluated for software reliability prediction under different criteria for different techniques. Diwaker et al. [[1]] proposed a novel mathematical model using parallel and series reliability models to estimate CBS reliability. Then, evaluation is done using computational intelligence techniques, namely Fuzzy logic, and Particle Swarm Optimization. The result concluded that the Fuzzy logic-based model predicts reliability better than PSO. Bharathi et al. [[14]] described a new, logic error-based approach for reliability prediction. The framework is called Software Failure Estimation with Logic Error. The research was focused on studying the impact of a logic error on a critical system like ABS in automotive. They used a Hidden Markov Model to estimate failures across the system due to the introduction of a logic error. The authors then studied the different error states, types of failures that occurred in the system, and the worst-case outcome. This approach can be used to understand the impact of a single error at the design stage itself. However, the proposed framework is very selective in its approach and may not apply to other systems. Similarly, Chau Pattnaik et al. [[15]] proposed the use of the Discrete Time Markov Chain for designing a reliability estimation framework. The model is then applied to a queue management application under different execution scenarios with average execution time and component reliabilities as input parameters. The mean and variance of visits to each component from the initial component are calculated to determine component reliability. It was found that the component reliability decreases with an increase in mean and variance. The system reliability is estimated using component reliability, component dependency graph, and component transition probabilities. Shivani et al. [16][17][18][19][20] studied various quality models related to components and role of computational intelligence in improving the reliability of components. Omdev et al. [21][22][23][24][25] research concludes how important is testing during the development phase of software using computational techniques to improve the quality of software.

The literature reveals various computational techniques used by researchers in reliability prediction which are listed in table 1 along with their tasks.

Table 1: Computational Intelligence Techniques With Their Implication

Technique	Implication
Artificial Neural Network	Helps in prediction, classification, and non parametric model
Fuzzy Logic	Helps in prediction and classification, apprehends abstract data of software metrics since the initial phase of software development
Genetic Programming	Helps in optimization and prediction for computational intelligence technique
Naïve Bayes	Helps in the classification of problems
Particle Swarm Optimization	Helps in optimization and prediction for computational intelligence technique
Regression Analysis	Helps in estimating problems and resolving the relationship between variables
Statistical Modeling Technique	Used for resolving estimation issues, parametric models, faults in software are treated as a random aspect
Support Vector Machine	Used for the classification problem, Regression problem
Deep Learning	Helps in automatic feature engineering, the classification problem

It has been observed from the literature review that a lot of work has been done by various researchers in the fields of reliability prediction as software reliability is one of the major attributes for improving software quality, but it is also observed that very less work has been done to predict the reliability of software components. Therefore, there is much scope in designing models for predicting the reliability of components. Another observation from the literature review is that the deep learning model is becoming the preferred choice for reliability prediction because of the high potential in prediction with more accuracy.

3. DESIGN AND IMPLEMENTATION OF MODEL

3.1 Workflow of the model

The model for component reliability prediction is designed using an artificial neural network to implement a deep learning approach.

Many types of neural networks, consisting of different numbers of hidden layers, exist in the deep learning domain. Each neuron in a layer act as a computational unit, applying an activation function on the input and passing the output to the next layer. Every input is provided weight and the product of input, and its weight is fed to the activation function. The major parameters for a neural network are batch size, number of neurons, number of hidden layers, learning rate, and epoch. These parameters need to be optimized to improve the neural network's ability to find patterns in the data. A smaller number of neurons and hidden layers can result in poor prediction ability whereas a high number of neurons and hidden layers can result in excessive computation effort and time depending upon the dataset. Generally, it is observed that the number of neurons in each hidden layer should be the same or decreased towards the direction of the output. Batch size refers to the number of training examples processed in one pass of the network. It is directly proportional to the time taken for training the neural network. However, the batch size should not be too small either as it might reduce the estimation accuracy of the network.

Learning rate is a hyper-parameter that controls the step size for adjusting weights in the network to adjust the loss gradient for optimization. If the step sizes for the learning rate are big, it will yield a sub-optimal result whereas a very small step size can extend the training time. An epoch is said to be completed when a dataset has completed both a forward and backward pass. For large datasets, this is usually done by dividing the dataset into batches and running iterations. More epochs can lead to improved performance at the cost of increased training time.

The model is validated to determine its performance on validation data. The model design can then be tweaked for finding improved models by changing the hyper-parameters if needed. The workflow for a neural network is shown in fig 1

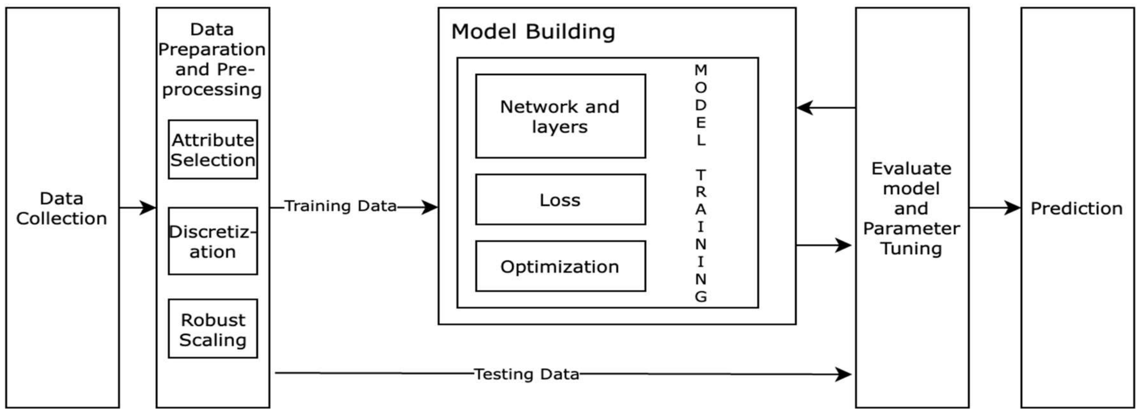
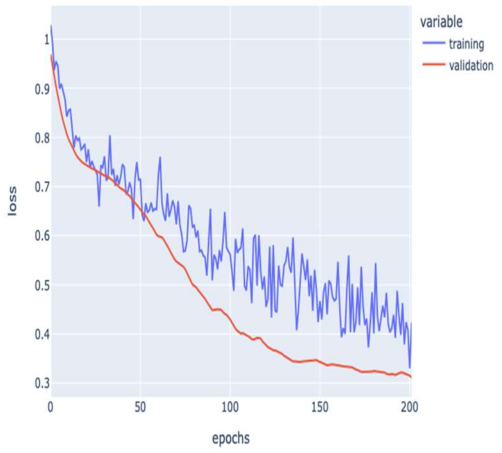


Figure 1: Workflow Of Deep Learning Based Model

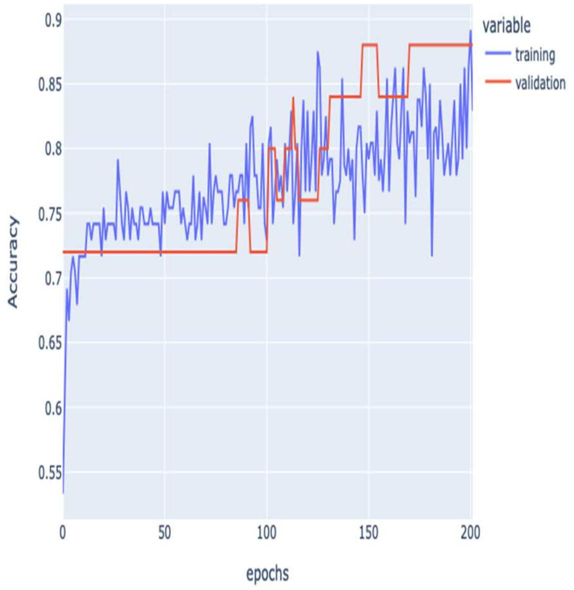
3.2 Configuration and Results

In this paper, a dense neural network based on deep learning is used which is appropriate for taking numeric data as input. The model is designed for predicting the reliability of components on two different datasets. The first dataset (dataset1) used in this study consist of five metrics i.e., complexity, component interaction, component dependency, failure, and reusability [[1]]. The given reliability in the dataset is used as a target feature. The reliability values are given in the range from 0 to 1 which are further bifurcated as low, medium, and high [0, 0.35], (0.35, 0.6], (0.6 ,1] respectively [[1]]. Multiclass classification is applied to predict reliability. Dataset is divided among training, validation, and testing in the ratio of 70%, 10%, and 20% respectively. Hyper parameters are fine-tuned, after many attempts' best configuration with the highest accuracy is chosen. The best accuracy configuration is specified in table 2 where the model is trained with four hidden layers with a different number of neurons. Tanh, adam, cross-entropy function is used as activation function, optimizer, loss function respectively for evaluating the model. Fig.1&2 shows the accuracy and loss graph of dataset1 over the number of epochs. The results in terms of accuracy, precision, recall, and f1-score are achieved as 96%, 95%, 96%, and 95% respectively. Table 4 shows the result and comparison with the other models on dataset1.

The second dataset (dataset2) [26] is built using 15 Java projects for bug prediction. The total instances used in this dataset are 1,82,671 and 103 attributes are used for predicting the bugs. Fig. 3 and 4 represents the loss and accuracy graph for dataset2. Dataset2 in table 3 shows different configurations, and table 5 shows results for all performance metrics along with other models. The result shows accuracy, precision, recall, and f1-score as 93%, 92%, 93%, and 92% respectively with three numbers of hidden



layers, tanh as activation function, adam as an optimizer, and the number of bugs is taken as a target



feature where data is predicted as faulty/non-faulty for predicting reliability.

Figure 2: Accuracy Graph For Dataset1

Figure 3: Loss Graph For Dataset1

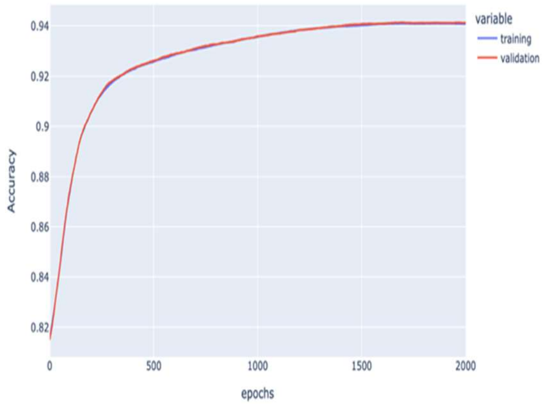


Figure 4: Accuracy Graph For Dataset2

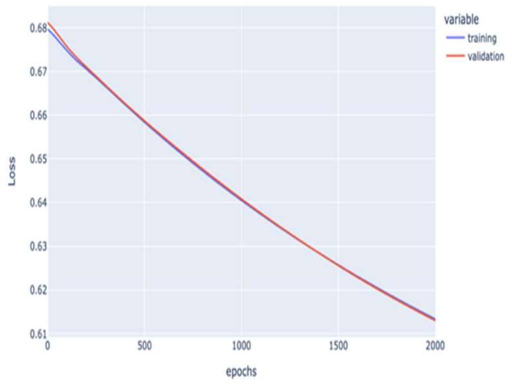


Figure 5: Accuracy Graph For Dataset2

Table 2: Different Configurations Of Deep Learning Based Model For Dataset1

Number of Layers	Activation Function	Accuracy (%)	Precision (%)	Recall (%)
[5,32,64,8,3]	Tanh + Dropout	96	96	96
[5,10,15,3]	Tanh+ Dropout	94	95	94
[5,10,15,3]	ReLu + Dropout	74	55	74
[5,15,3]	Tanh + Dropout	91	92	91
[5,32,16,8,3]	ReLu	74	55	74

Table 3: Different Configurations Of Deep Learning Based Model For Dataset2

Number of Layers	Activation Function	Accuracy (%)	Precision (%)	Recall (%)
[103,512,1024,128,2]	ReLu	92	96	96
[103,256,512,64,2]	ReLu	90	93	90
[103,256,512,64,2]	ReLu + Dropout	90	93	90
[103,512,128,2]	ReLu	83	93	83

4. COMPARISON WITH EXISTING MODELS

The designed model is compared with other machine learning models such as K- Nearest Neighbor (KNN), SVM, Gaussian Naive Bayes, Component Naive Bayes, Decision Tree, Ensemble method, Random Forest, and deep learning-based model. The comparison is done by applying all the models on the same dataset using performance metrics such as accuracy, precision, recall, and f1-score. A brief description of the models used for comparison is given below:

4.1 KNN

The KNN is a non-parametric approach for classifying data points based on their distance and correlation with other available data [27]. After experimenting with different values of k, finally, k is taken as 8 for dataset1 and five hundred for dataset2.

4.2 Naïve Bayes

For multi-class prediction issues, Naive Bayes is the option made by developers. It requires less training data if the assumption of feature independence is correct. For categorical input variables, Naive Bayes is a better fit and makes real-time predictions [27]. In this study, the two different types of Naive Bayes are used:

4.2.1 Complement NB (CNB)

As the dataset is slightly imbalanced that's why CNB, a variation of the Multinomial Naïve Bayes (MNB) algorithm, is used. CNB is especially suited for application in imbalanced data sets, as is our case. In this study, the value of the additive (Laplace/ lidstone) smoothing parameter for dataset1 is five and for dataset2 is .0005.

4.2.2 Gaussian NB (CNB)

This algorithm follows Gaussian normal distribution and supports continuous data.

4.3 SVM

Support Vector Machines (SVMs) are a group of supervised machine learning models that can be used for classification or regression.

4.3.1 SVC

Support Vector Classification (SVC) implementation in scikit-learn is based on libsvm. It is essentially a wrapper around SVM which support multiple kernels and can be used for binary or multi-class classification.

4.3.2 Linear Support Vector Classification

Linear SVC only supports a linear kernel. It is faster and easier to scale due to which it can be used when the number of samples is large [27].

When applying the dataset for SVM model (Linear SVC and SVC) the configuration results were not satisfactory due to the imbalanced nature of the dataset, for better results we added class weights to the model.

4.4 Random Forest

Random Forest algorithms are widely used for classification and regression. It is a type of ensemble technique since it combines results from multiple decision trees to arrive at the final classification [27]. In this study, for dataset1 maximum depth of the tree is limited to four and the maximum number of leaf nodes of the grown trees has been limited to three without using the bootstrap sampling method. For dataset2 maximum depth is fifteen, the number of trees in the forest is fifty, and class weight is balanced.

4.5 Decision Tree

Decision Trees (DTs) are supervised learning algorithms that work by successively applying decision rules based on data features [27]. The outcome is reflected in the leaf nodes, whereas the decision nodes reflect the feature on which splits are made. In this study, maximum depth is limited to four for dataset1 and the minimum number of samples required to split an internal node has been set to 8. For dataset2 maximum depth is twenty, and class weight is balanced.

Table 3: Dataset1 model results with other models.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Deep Learning Based Model (Dataset1)	95	93	93	93
Random Forest	88	86	65	64
Decision Tree	85	73	68	65
SVC	92	87	83	85
KNN	81	73	55	60
Gaussian NB	95	93	89	91
Linear SVC	88	82	75	74
Complement NB	72	60	74	63

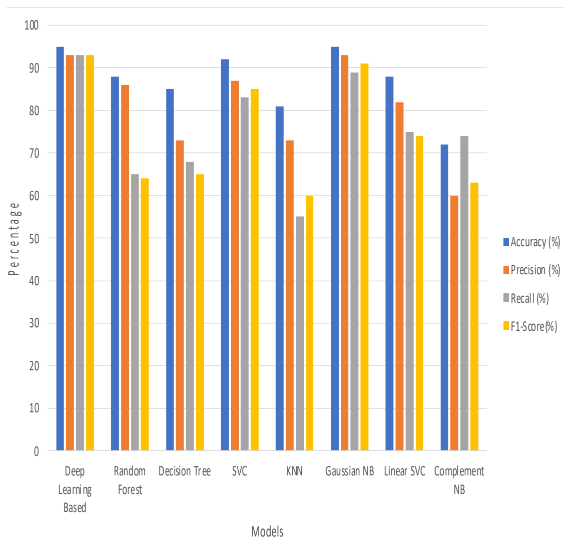


Figure 6: Comparison Graph With Other Models For Dataset1

Table 3: Dataset2 model results with other models.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Deep Learning Based Model (Dataset2)	93	92	93	92
Random Forest	90	97	90	93
SVC	78	95	78	84
KNN	81	73	55	60
Gaussian NB	88	93	88	90
Linear SVC	79	94	79	85
Complement NB	57	93	57	69

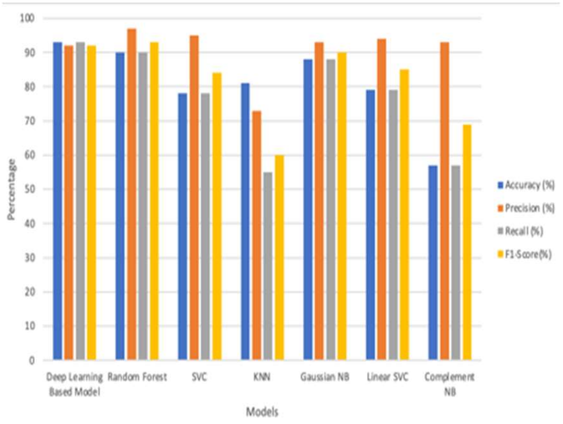


Figure 7: Comparison Graph With Other Models For Dataset2

It is observed that the proposed deep learning-based model obtains better accuracy among all the models for both dataset1 and dataset2.

5. CONCLUSION

Predicting the component's reliability in CBS is an important activity as the quality of the software highly depends on the reliability of individual components. The reliability prediction models for CBS are highly in demand due to their less availability. The model designed in this study predicts the software reliability using a deep learning approach. The performance of the designed model on the datasets is on the higher side. Results of the component reliability prediction help to improve its quality by taking corrective actions. The comparison with other well-known models such as KNN, random forest, decision tree, SVC, linear SVC, Gaussian NB, and complement NB shows that the designed model outperforms in terms of accuracy, precision, recall, and f1-score. The accuracy of the proposed techniques can still be challenged considering components and integration changes. The study can be extended with more datasets for improving the predictability of software reliability and to test the other parameters that can help in reliability prediction.

REFERENCES:

[1] Diwaker et al., "A New Model for Predicting Component-Based Software Reliability Using Soft Computing", *IEEE Access*, Vol. 7, 2019, pp. 147191-147203,. doi: 10.1109/ACCESS.2019.2946862

[2] Shivani Yadav and Bal kishan, "Software Reliability Prediction by using Deep Learning Technique"**,** *International Journal of Advanced Computer Science and Applications(IJACSA),* Vol. 13, No. 3, 2022, pp. 1-13.

[3] Chander Diwaker, Pradeep Tomar, Ramesh C. Poonia, and Vijander Singh, "Prediction of Software Reliability using Bio Inspired Soft Computing Techniques" *Journal of Medical Systems*, Vol. 42, No. 5, 2018, pp. 1-16. doi:10.1007/s10916-018-0952-3

[4] Lotfi A. Zadeh, "Fuzzy Logic and Soft Computing: Issues, Contentions and Perspectives", *Proceedings of IIZUKA'94: Third Int. Conf. on Fuzzy Logic, Neural Nets and Soft Computing,* Iizuka, 1994, pp. 1–2.

[5] Madsen, Henrik, Poul Thyregod, B. Burtschy, Florin Popentiu, and Grigore Albeanu, "On using soft computing techniques in software reliability engineering", *International Journal of Reliability, Quality and Safety Engineering,* Vol. 13, No. 1, 2006, pp. 61-72. doi:10.1142/s0218539306002094

[6] Yasutaka Kamei, Akito Monden, and Ken-lchi Matsumoto, "Empirical Evaluation of SVM-Based Software Reliability Model," *Proceedings Fifth ACM-IEEE International Symposium on Empirical Software Engineering (ISESE),* Vol. 2, 2006, pp. 39-41.

[7] Ping-Feng Pai, and Wei-Chiang Hong, "Software reliability forecasting by support vector machines with simulated annealing algorithms", *Journal of Systems and Software,* Vol. 79, No. 6, 2006, pp. 747–755.

[8] Jun Zheng, "Predicting software reliability with neural network ensembles", *Expert Systems with Applications,* Vol. 36, No. 2, 2009, pp. 2116–2122.

[9] Yogesh Singh, and Pradeep Kumar, "Application of feed-forward neural networks for software reliability prediction.", *ACM SIGSOFT Software Engineering Notes,* Vol. 35, No. 5, 2010, pp. 1–6.

[10] Kirti Tyagi, and Arun Sharma, "An adaptive neuro-fuzzy model for estimating the reliability of component-based software systems", *Applied Computing Informatics,* Vol. 10, No. 1, 2014, pp. 38–51.

[11] Chander Diwaker, and Pradeep Tomar, "Assessment of Ant Colony using Component-Based Software Engineering Metrics", *Indian Journal of Science and Technology,* Vol. 9, No. 44, 2016, pp. 1–5.

[12] Ravi Kumar Sharma, and Parul Gandhi, "Estimate reliability of component-based software system using modified neuro-fuzzy model", *International Journal of Engineering & Technology*, Vol. 6, No. 2, 2017, pp. 45–49.

[13] Vinod Kumar Kulamala, A. Sarath Chandra Teja, Abha Maru, Yogesh Singla, and Durga Prasad Mohapatra, "Predicting Software Reliability using Computational Intelligence Techniques: A Review", *International Conference on Information Technology (ICIT)*, December 19-21 (India), 2018, pp. 114-119. doi:10.1109/icit.2018.00033

[14] R. Bharathi, and R. Selvarani, "Hidden Markov Model Approach for Software Reliability Estimation with Logic Error", *International Journal of Automation and Computing,* Vol. 17, No. 2, 2020, pp. 305-320.

[15] Sampa ChauPattnaik, Mitrabinda Ray, and Mitali Madhusmita Nayak, "Component Based Reliability Prediction", *International Journal of System Assurance Engineering and Management,* Vol. 12, No. 3, 2021, pp. 391-406. doi:10.1007/s13198-021-01079-x.

[16] Shivani Yadav and Bal Kishan, "Reliability of Component-Based Systems- A Review", *International Journal of Advanced Trends in Computer Science and Engineering*, Vol.8, No. 2, 2019, pp. 293–299.

[17] Shivani Yadav and Bal Kishan, "Assessment of software quality models to measure the effectiveness of software quality parameters for

Component Based Software (CBS)," *Journal of Applied Science and Computations*, Vol. 6, Issue 4, 2019, pp. 2751–2756.

[18] Shivani Yadav and Bal Kishan, "Analysis and Assessment of Existing Software Quality Models to Predict the Reliability of Component-Based Software," *International Journal of Emerging Trends in Engineering Research,* Vol. 8, No. 6, 2020, pp. 2824–2840.

[19] Shivani Yadav and Bal Kishan, "Component-Based Software System using Computational Intelligence Technique for Reliability Prediction," *International Journal of Advanced Trends in Computer Science and Engineering*, Vol. 9, No. 3, 2020, pp. 3708–3721.

[20] Shivani Yadav and Bal Kishan, "Assessments of Computational Intelligence Techniques for Predicting Reliability of Component Based Software Parameter and Design Issues," *International Journal of Advanced Research in Engineering and Technology*, Vol. 11, Issue 6, 2020, pp. 565–584.

[21] Omdev Dahiya and Kamna Solanki,"An Efficient APHT Technique for Requirement-Based Test Case Prioritization," *International Journal of Engineering Trends and Technology*, Vol. 69, 2021, pp. 215–227.

[22] Omdev Dahiya and Kamna Solanki,"Prevailing Standards in Requirement-Based Test Case Prioritization: An Overview," *ICT Analysis and Applications,* 2021, pp. 467–474.

[23] Omdev Dahiya and Kamna Solanki,"A Study on Identification of Issues and Challenges Encountered in Software Testing," *In Proceedings of International Conference on Communication and Artificial Intelligence ,* 2021, pp. 549–556.

[24] Omdev Dahiya, Kamna Solanki, and Amita Dhankhar, "Risk-based testing: identifying, assessing, mitigating & managing risks efficiently in software testing," *International Journal of Advanced Research in Engineering and Technology,* Vol. 11, No. 3, 2020, pp. 192–203.

[25] Omdev Dahiya and Kamna Solanki,"A systematic literature study of regression test case prioritization approaches," *International Journal of Engineering & Technology*, Vol. 7, No. 4, 2018, pp. 2184–2191.

[26] https://www.inf.u-szeged.hu/~ferenc/papers/GitHubBugDataSet/

[27] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, 2011, pp. 2825-2830.

[28] Elena N. Akimova, Alexander Yu. Bersenev, Artem A. Deikov, Konstantin S. Kobylkin, Anton V. Konygin, Ilya P. Mezentsev, and Vladimir E. Misilov, "A Survey on Software Defect Prediction Using Deep Learning", *Mathematics*, Vol. 9, No.11, 2021, pp. 1-14. https://doi.org/10.3390/math9111180