IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁂ Peer-reviewed & Refereed journal ⁂ Vol. 14, Issue 4, April 2025

DOI: 10.17148/IJARCCE.2025.14447

# Estimating Software Anomalies Using Machine Learning

**Dr. Mahesh Kotha[1], G Akshith Reddy[2], Kasi Sailaja[3], Dr. Krishna Kumar N[4], Velpula Sunil Kumar[5]**

Associate professor, Department of CSE (AI&ML), CMR Technical Campus, Hyderabad[1]

Assistant Professor, Department of CSE, Guru Nanak Institutions Technical Campus, Hyderabad[2]

Department of CSE, Guru Nanak Institutions Technical Campus, Hyderabad[3]

Associate Professor, Department of Computer Applications, University Institute of Computer science and Applications, Guru Nanak University, Hyderabad[4]

Assistant Professor, Department of CSE, Jyothishmathi Institute of Technology and Science, Telangana[5]

**Abstract:** Estimating Software Anomalies is a critical aspect of ensuring the reliability and quality of software projects. It involves identifying and predicting bugs or faults in the source code. By detecting faults early on, developers can address them promptly, leading to improved software development processes and reduced debugging time and costs. This project focuses on developing an intelligent system that leverages advanced machine learning techniques to predict software faults in the source code. The proposed software fault prediction (SFP) model utilizes a combination of LSTM networks, bidirectional LSTM networks, support vector machines (SVM), and neural networks. These techniques enable the system to learn complex patterns in the data and make accurate predictions. The effectiveness of the proposed system is evaluated using real-world software projects, where it outperforms existing software fault prediction models. By implementing this intelligent system, developers can quickly and accurately detect faults, leading to enhanced reliability and quality of software projects. The system's ability to identify bugs during the development stage of a new project version and detect faults in a completed version offers significant benefits. It empowers developers to improve the software development process, reduce debugging time, and ultimately deliver higher-quality software projects.

**Keywords:** LSTM, SFP, SVM, software anomalies, machine learning algorithm.

## I.INTRODUCTION

The motivation for a software fault prediction project is to proactively identify and address potential defects or issues in software systems before they cause significant problems or impact users. By accurately predicting which parts of the software are more likely to fail or have defects, developers can allocate resources more effectively, prioritize testing efforts, and implement targeted fixes and improvements. This proactive approach helps to enhance software quality, increase user satisfaction, minimize downtime, and reduce maintenance costs in the long run.
A software fault prediction project aims to leverage data and analytics to identify customers at risk of defection, enabling companies to take proactive measures, improve customer retention, and drive business growth.

There are several motivations for undertaking a project on software fault prediction. Here are a few key reasons:

1. Cost Reduction: Predicting faults helps identify and resolve them early on, reducing the overall cost associated with bug fixing, patching, and maintenance.
2. Enhanced Customer Satisfaction: Predicting and addressing potential faults in advance delivers a more stable and reliable product to customers, leading to higher satisfaction and better user experience.
3. Time Efficiency: Fault prediction helps prioritize testing efforts, saving time and resources by focusing on parts of the software more likely to contain faults.
4. Process Improvement: Fault prediction provides insights into the software development process, allowing for improvements in coding practices and the prevention of similar issues in the future.
5. Proactive Maintenance: Predicting faults enables proactive maintenance activities, such as early bug fixing and system updates, minimizing downtime and ensuring optimal performance.
6. Research and Advancements: Undertaking a project in software fault prediction contributes to the broader knowledge base, advancing software engineering practices and making a meaningful impact on the industry.

Overall, a project on software fault prediction offers the opportunity to improve software quality, reduce costs, and optimize development processes

## II.RELATED WORKS

The problem addressed in this project is the accurate prediction of software faults in source code. Existing methods for fault prediction have limitations in terms of accuracy and adaptability to different software projects. There is a need for an intelligent system that utilizes advanced machine learning techniques to analyze code, identify fault patterns, and make precise predictions. The objective is to develop a model that can effectively detect and prioritize potential fault-prone areas, enabling proactive measures to improve software quality and reliability.

This project aims to address the limitations of existing software fault prediction methods by leveraging advanced machine learning techniques. The goal is to develop a model that can accurately identify software faults in source code, assisting developers in detecting and addressing issues early in the development process. By improving fault prediction accuracy, developers can allocate resources effectively, prioritize testing efforts, and implement targeted fixes and improvements. Ultimately, this project strives to enhance software quality and reliability, reducing the impact of faults and improving the overall user experience.

The objectives of a software fault prediction project are to develop an intelligent system capable of accurately predicting software faults in source code. The system aims to enhance software reliability by detecting faults early in the development process, enabling developers to allocate resources effectively and address potential issues promptly. By minimizing software faults, the project seeks to reduce maintenance costs associated with bug fixing and patching.

Another objective is to improve customer satisfaction by delivering a more stable and reliable software product. By predicting and addressing potential faults in advance, the system aims to enhance the user experience and increase customer satisfaction. The project also aims to optimize development processes by analysing historical data and identifying common causes of faults. This
knowledge can help improve coding practices and prevent similar issues in the future, contributing to continuous process enhancement.

Additionally, the project seeks to contribute to research and advancements in software fault prediction. By exploring novel techniques and methodologies, it aims to advance the field and make meaningful contributions to the broader knowledge base in software engineering. In summary, the objectives of a software fault prediction project are to develop an intelligent system for accurate fault prediction, enhance software reliability, reduce maintenance costs, improve customer satisfaction, optimize development processes, and contribute to research in the field.

In software fault prediction, various existing systems and models have been developed to detect and predict faults in source code. These systems contribute to improving software quality and reliability. Existing approaches include:

1. Static Code Analysis Tools: These tools perform static analysis of source code to identify potential issues and suggest improvements.
2. Statistical and Machine Learning-based Approaches: Techniques like Support Vector Machines (SVM), Random Forests, and Naive Bayes classifiers are used to predict faults by analyzing historical data and code features.
3. Rule-based Systems: Rule-based systems use predefined rules or patterns to identify potential faults based on best practices and common programming errors.
4. Process Metrics-based Models: Certain models use process metrics like lines of code and complexity metrics to identify patterns and predict potential faults, focusing on the software development process itself.

These existing systems have significantly contributed to fault prediction by helping to identify and address issues, ultimately enhancing software quality.

## III.LITERATURE SURVEY

This section discusses several existing methodologies for resume screening and various types of methods . These are few papers that Automated Resume Screening.

“Software Defect Prediction Using Machine Learning Techniques” [1], was published by C.Lakshmi Prabha, Dr.N.shivakumar , Proceedings of the Fourth International Conference on Trends in Electronics and Informatics (ICOEI 2020): Software defect prediction yields tangible results for development teams, contributing to industrial outcomes and aiding in the anticipation of development faults within areas of potentially defective code. This assists developers in pinpointing bugs and structuring their testing endeavors. The accuracy percentage of classification for effective prediction holds significance in early detection. Additionally, datasets containing software defects are supported and, to some extent, acknowledged despite their vast dimensions.

Addressing this challenge is achieved through a hybridized approach involving PCA, Random Forest, Naïve Bayes, and the SVM Software Framework. This approach is employed on five datasets—PC3, MW1, KC1, PC4, and CM1— for software analysis using the Weka simulation tool. A systematic research analysis is conducted, measuring and comparing parameters such as confusion, precision, recall, recognition accuracy, and more against existing schemes. The analytical assessment suggests that the proposed approach offers more valuable solutions for predicting device defects.

“Comparative Analysis of Supervised Learning Techniques of Machine Learning for Software Defect Prediction”[2], was published by Anurag Gupta1 Ratnesh Kumar Shukla, Dr. Abhishek Bhola, Alok Singh Sengar, Proceedings of the SMART–2021, IEEE Conference: Detecting software bugs or predicting defects is of paramount importance for organizations, enabling the early identification of flaws in the software development process. This empowers software developers to pinpoint potential areas where defects could emerge. Within this research paper, various statistical techniques, including Linear Regression, Naïve Bayes, Random Forest, Decision Tree, and Artificial Neural Networks, have been meticulously compared. The objective was to identify the most effective technique for bug prediction. This comparison was based on Performance Measures such as Accuracy, Precision, Recall, and F-measure.

| SL No. | Year of Publish | Authors and Paper Title | Techniques /Methods | Merits | Demerits |
|---|---|---|---|---|---|
| 1 | 2020 | Software Defect Prediction using Machine Learning Techniques.<br><br>C.Lakshmi Prabha, Dr.N.Shivakumar | Naive Bayes,Random Forest,SVC,Neural Network. | SVM can handle High Dimensional Data with a large number of features. | SVM algorithm is not suitable for large data sets. |
| 2 | 2021 | Comparative Analysis of Supervised Learning Techniques of Machine Learning for Software Defect Prediction.<br>Anurag Gupta, Ratnesh Kumar Shukla, Dr.Abhishek Bhola, Alok Singh Sengar | Linear Regression,Naive Bayes,Random Forest,ANN,Decision Tree. | Random Forest can handle both categorical and continuous data,making it suitable for a wide range of software fault prediction tasks. | A Decision Tree often requires more time to train the model. |
| 3 | 2022 | A Study on Predicting Software Defects with Machine Learning Algorithms.<br><br>Anjali.C,Julia Punitha Malar Dhas, J.Amar Pratap Singh | SVM,Decision Tree,RandomForest. | SVM can handle High Dimensional Data with a large number of features. | SVM algorithm is not suitable for large data sets. |

“A Study on Predicting Software Defects with Machine Learning Algorithms”[3] , was published by Anjali C, Julia Punitha Malar Dhas, J. Amar Pratap Singh, 2022 International Conference: Software Defect Prediction (SDP), even in its preliminary stages, emerges as a critical and significant undertaking. It has garnered substantial attention as a method of assuring quality. The operational component services can generate extensive volumes of reports and defect data. While considerable effort has been directed towards formulating defect prediction models through machine learning (ML), there has also been exploration into determining the efficacy of the source code. ML, functioning as a supervised algorithm, is harnessed to achieve enhanced outcomes. To enhance our comprehension of defect prediction within the realm of SOS, this study proposes a fault diagnosis framework grounded in web- accessible care. ML tools, notably Random Forest (RF), Decision Tree (DT), and Support Vector Machine (SVM), are harnessed to assess the model's utility, with specific metrics derived from feature extraction techniques. Ultimately, the performance of these ML algorithms is juxtaposed and the superior performer is subjected to in-depth analysis.

## IV.PROPOSED WORK

The proposed system for software fault prediction involves the following steps. Firstly, relevant data is collected from software artifacts such as source code, bug reports, version control systems, and development logs. This data is pre-processed to handle missing values, normalize features, and address data quality issues. Meaningful features are extracted, including static code metrics, process metrics, and textual features from bug reports or comments. Feature selection techniques are applied to identify the most relevant subset of features for fault prediction, reducing dimensionality and preventing overfitting. Machine learning algorithms such as logistic regression, decision trees, random forests, support vector machines, or neural networks are selected and trained using the data. The model's hyperparameters are optimized to enhance its performance. The trained model is deployed and integrated into the software development process, providing real time predictions, visualizations, or alerts to assist developers in identifying potential fault-prone areas. Performance evaluation is conducted using metrics like accuracy, precision, recall, and F1 score.

Cross-validation or holdout validation techniques are used to estimate the model's performance on unseen data and detect overfitting. Continuous monitoring and updating of the deployed model are performed using new data from the software development process. This ensures the model maintains accuracy and adapts to changing software conditions.
System Architecture is the conceptual model that defines the structure, behaviour, and views of a system. It defines how the system is implemented and the working of the project.

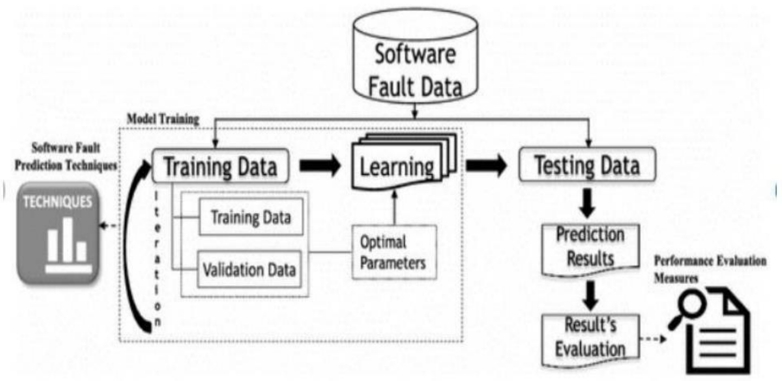


Fig. 1. Proposed Framework for Automated Resume Screening.

## LONG SHORT-TERM MEMORY (LSTM)

Long Short-Term Memory (LSTM) is a powerful recurrent neural network (RNN) architecture used in machine learning for handling sequential data. LSTM networks are designed to capture long-term dependencies by introducing a memory cell that selectively remembers or forgets information over extended time intervals. They have been successfully applied in various tasks such as speech recognition, sentiment analysis, and time series forecasting. In software fault prediction, machine learning algorithms are utilized to analyse software-related data, including code metrics and bug reports, to make predictions about fault occurrences. LSTM models, specifically designed for handling sequential or time series data, are used to capture temporal patterns and improve fault prediction accuracy. To implement LSTM-based software fault prediction, the data is prepared by organizing it in a sequential format and encoding it appropriately for input to the LSTM network. The model architecture consists of memory cells and gating mechanisms for information storage and retrieval. The LSTM model is then trained using historical data, adjusting its weights through backpropagation to capture patterns and dependencies. Evaluation is performed using metrics to assess the model's performance, and the trained LSTM model can make predictions for new sequential data. Continuous model refinement, including hyperparameter tuning and incorporating domain-specific knowledge, contributes to improving the performance of LSTM-based software fault prediction. By effectively handling time-dependent features and capturing complex temporal relationships, LSTM networks enhance the accuracy and reliability of software fault prediction systems.

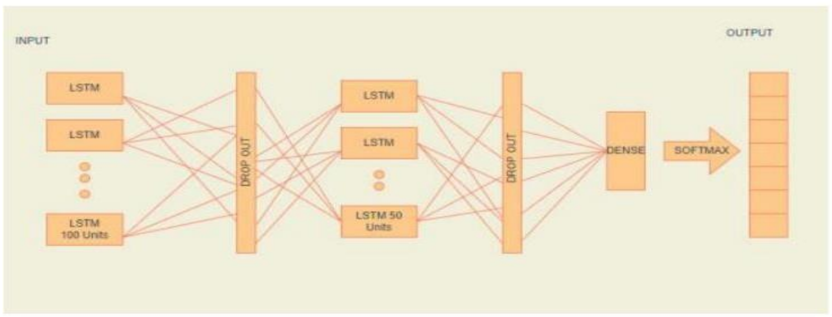


Fig. 2. LSTM Architecture.

## STACKED BI-DIRECTIONAL LSTM

Stacked Bi-directional LSTM is a powerful extension of LSTM that combines bidirectional processing and multiple layers to capture complex dependencies in sequential data. It is widely used in various machine learning tasks. The architecture consists of stacked layers of bidirectional LSTM units that process the input sequence in both forward and backward directions, enabling a comprehensive understanding of the data. In software fault prediction, relevant data is collected and pre-processed in a sequential format. The model is then trained to capture temporal patterns and dependencies, and evaluated using appropriate metrics.

The trained model can accurately predict software faults for new sequential data, and its performance can be further improved through techniques like hyperparameter tuning and model refinement. Stacked bidirectional LSTM networks are particularly advantageous in capturing both past and future context, making them effective in handling complex relationships and long-term dependencies, thereby enhancing the accuracy of software fault prediction.

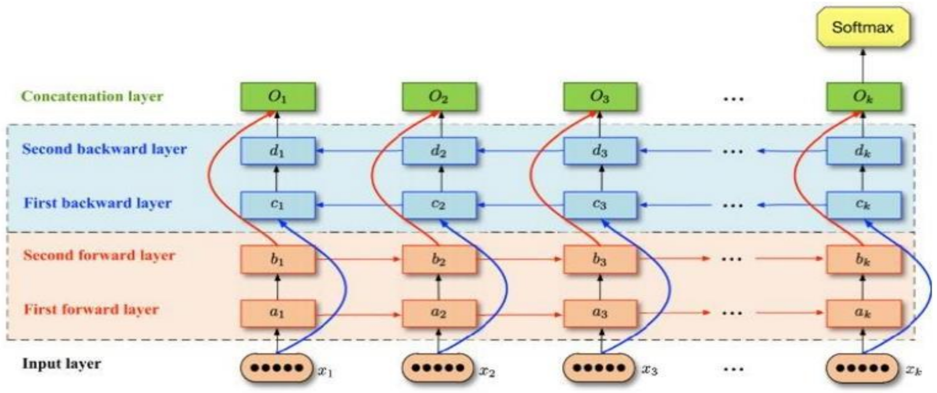


Fig. 3. Stacked Bi Directional LSTM Architecture.

However, it is essential to consider factors such as having sufficient training data, conducting careful hyperparameter tuning, and optimizing the model to prevent overfitting. These considerations play a vital role in the effective utilization of stacked bidirectional LSTM networks in software fault prediction. Through iterative refinement and optimization based on evaluation results and insights gained from the software fault prediction task, the model can be continuously improved. By harnessing the capabilities of stacked bidirectional LSTM networks, developers can enhance the accuracy and performance of software fault prediction, enabling proactive identification and mitigation of potential issues in software systems.

## NEURAL NETWORKS PREDICTIONS

Neural network predictions are generated by powerful machine learning models that can learn complex patterns and make predictions based on the input data. In software fault prediction, neural networks are utilized to analyse software-related data and make predictions about fault occurrences. The process involves data preparation, where relevant data is collected and pre-processed, and the neural network architecture is designed. The model is then trained using the training set, and its performance is evaluated using appropriate metrics. Once trained and evaluated, the neural network can make predictions on new or unseen software-related data. To improve the neural network's performance, various techniques can be employed, such as finetuning the model architecture, adjusting hyperparameters, and applying regularization techniques. The success of neural networks in software fault prediction depends on factors like data availability and quality, appropriate network architecture design, and effective training strategies. The choice of neural network architecture should consider the specific characteristics of the software-related data and the problem being addressed. Experimentation and exploration of different approaches can help identify the most effective neural network approach for software fault prediction. By leveraging the capabilities of neural networks, software fault prediction becomes a data-driven process that can uncover intricate fault indicators in software-related datasets. The ability of neural networks to capture complex relationships and patterns in the data contributes to improving the accuracy and reliability of software fault prediction systems.

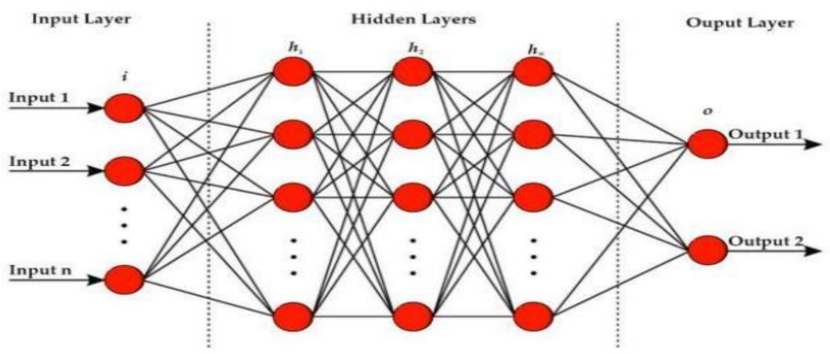


Fig. 3.1.4: Artificial Neural Networks

**BDA PREDICTIONS**

Big Data Analytics (BDA) predictions involve applying advanced analytics techniques to large datasets to extract valuable insights and make accurate forecasts. In software fault prediction, BDA leverages tools and technologies to analyse vast amounts of software-related data, such as code repositories, bug reports, and performance logs. Data collection and storage utilize scalable solutions like Hadoop Distributed File System (HDFS) to handle the volume of data. Pre-processing techniques handle missing values and outliers, while data integration combines multiple sources for a comprehensive view of software development. Machine learning algorithms suited for big data analytics, such as decision trees, random forests, and neural networks, are selected for model training. The models are validated using testing sets or crossvalidation methods to ensure their generalization capabilities. Predictive analytics enable real-time analysis of large-scale datasets, providing early indications of faults and facilitating proactive measures. Anomaly detection techniques help identify abnormal patterns, contributing to fault 17 prevention. Continuous learning and improvement are achieved by updating models with new data and incorporating feedback from development teams and system monitoring. Scalability and performance optimization techniques, like distributed computing and parallel processing, ensure efficient processing of large-scale data. By leveraging big data analytics, software fault prediction benefits from analysing diverse datasets, enabling proactive fault detection, and enhancing overall software quality and reliability through data-driven decision-making.

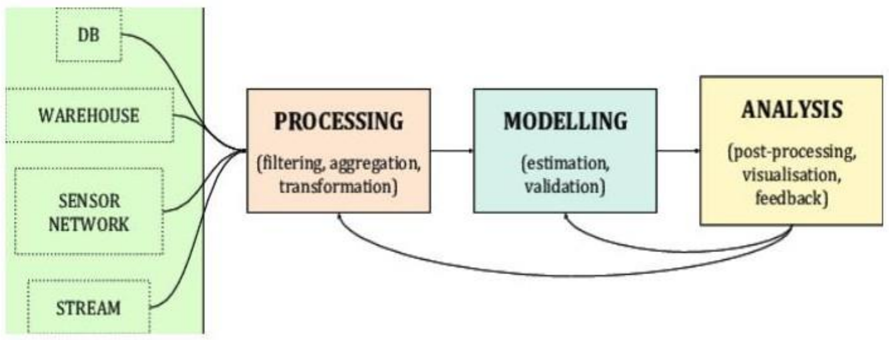

DB
WAREHOUSE
SENSOR NETWORK
STREAM
PROCESSING
(filtering, aggregation, transformation)
MODELLING
(estimation, validation)
ANALYSIS
(post-processing, visualisation, feedback)

Fig. 5. Workflow of a BDA Prediction.

## V.RESULTS AND DISCUSSION

Implementation and testing are crucial phases in software fault prediction. Once a fault prediction model is developed, it needs to be implemented in the software environment, integrating it into the development lifecycle and configuring data collection mechanisms. Rigorous testing is then conducted to evaluate the model's performance using diverse datasets and metrics. This ensures the model's reliability, validates its accuracy, and allows for iterative refinements to improve its effectiveness. It ensures that the software fault prediction model is not only theoretically sound but also practical and reliable in real-world scenarios. It enables developers and stakeholders to have confidence in the model's predictions and make informed decisions regarding fault mitigation strategies and software quality improvements. To evaluate the performance and assess the results of our models, we are utilizing a confusion matrix in our software fault prediction analysis. Specifically, we are constructing confusion matrices for the LSTM (Long Short-Term Memory), ANN (Artificial Neural Network), and CNN (Convolutional Neural Network) models. The confusion matrix provides valuable insights into the accuracy of our predictions by categorizing the actual and predicted fault instances into true positives, true negatives, false positives, and false negatives. This comprehensive evaluation approach allows us to measure the effectiveness of each model and make informed decisions based on the generated confusion matrices to enhance the overall performance of our software fault prediction system.

Here, we utilize binary classification due to the nature of the dataset. Binary classification involves two statements: class labels and probabilities. To evaluate the accuracy of our predictions, we employ a confusion matrix, which provides insights into the number of correctly and incorrectly predicted classes. The confusion matrix is structured with the X-axis representing the predictions and the Y-axis representing the class labels. When working with a balanced dataset, we can calculate accuracy based on the values in the confusion matrix. However, in an imbalanced dataset, where categories have significant variations in values, one category may dominate while others remain minimal.

ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 4, April 2025

DOI: 10.17148/IJARCCE.2025.14447

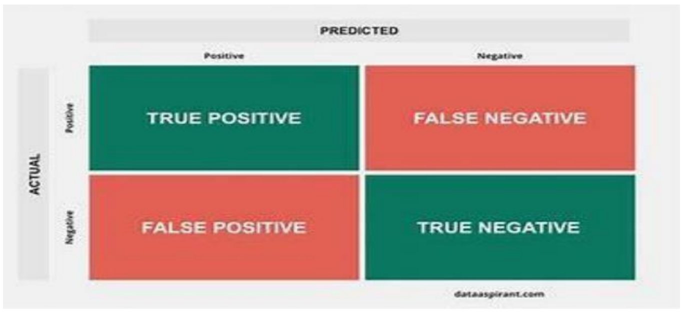
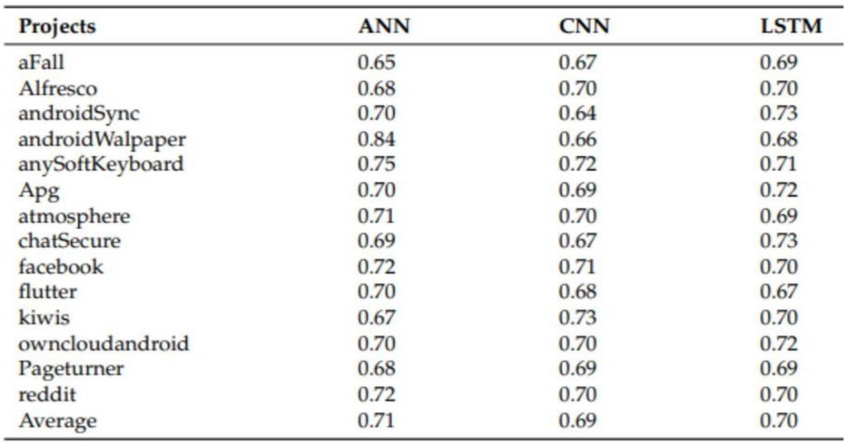


Fig. 6. Confusion Matrix.



| Projects | ANN | CNN | LSTM |
|---|---|---|---|
| aFall | 0.65 | 0.67 | 0.69 |
| Alfresco | 0.68 | 0.70 | 0.70 |
| androidSync | 0.70 | 0.64 | 0.73 |
| androidWalpaper | 0.84 | 0.66 | 0.68 |
| anySoftKeyboard | 0.75 | 0.72 | 0.71 |
| Apg | 0.70 | 0.69 | 0.72 |
| atmosphere | 0.71 | 0.70 | 0.69 |
| chatSecure | 0.69 | 0.67 | 0.73 |
| facebook | 0.72 | 0.71 | 0.70 |
| flutter | 0.70 | 0.68 | 0.67 |
| kiwis | 0.67 | 0.73 | 0.70 |
| owncloudandroid | 0.70 | 0.70 | 0.72 |
| Pageturner | 0.68 | 0.69 | 0.69 |
| reddit | 0.72 | 0.70 | 0.70 |
| Average | 0.71 | 0.69 | 0.70 |

Table 1. Comparing Performance Metrics of all Algorithms.

## VI.CONCLUSION

In conclusion, software fault prediction (SFP) poses challenges due to small and unbalanced datasets. However, weighted stacking methods proved beneficial, outperforming individual machine learning models. Despite limited feature selection impact, deep learning models excelled in both within version and cross version SFP. Stacked bi-directional LSTM surpassed Dense neural network, Convolution 1D, and bi-directional LSTM in within version SFP. In cross version SFP, sparse autoencoder stood out among all models, leveraging sparseness to enhance its learning process. These findings suggest SFP's feasibility and viability as a tool for the future. NLP/NLU techniques hold potential for reading and understanding source code to identify bugs. Exploring a new, deep-learningcompatible dataset and implementing intelligent ensembling techniques are avenues for future research, aiming to further improve software fault prediction accuracy.

Future work in SFP includes the creation of a tailored dataset, optimizing it for deep learning models, to obtain more representative data for evaluation. Incorporating NLP/NLU techniques to directly analyse source code would enhance bug detection accuracy. Furthermore, exploring advanced ensembling techniques would leverage the strengths of multiple models, enhancing prediction accuracy. These endeavours will advance the field of SFP, making it a powerful tool for proactive fault mitigation and software quality improvement.

## REFERENCES

[1]. A. O. Balogun, A. O. Bajeh, V. A. Orie, and A. W. Yusuf-asaju, “Software defect prediction using ensemble learning: an ANP based evaluation method,” Journal of Engineering Technology, vol. 3, no. 2, pp. 50–55, 2018.

[2]. Z. Li, X.-Y Jing, and X. Zhu, “Progress on approaches to software defect prediction,” IET Software, vol. 12, no. 3, pp. 161–175, 2018.

[3]. R. Malhotra and S. Kamal, “An empirical study to investigate oversampling methods for improving software defect prediction using imbalanced data,” Neurocomputing, vol. 343, pp. 120– 140, 2019.

[4]. Ravindra Changala, "Sustainable Manufacturing through Predictive Maintenance: A Hybrid Jaya Algorithm and Sea Lion Optimization and RNN Model for Industry 4.0", 2024 8th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), ISSN: 2768-0673, DOI: 10.1109/I-SMAC61858.2024.10714701, October 2024, IEEE Xplore.

[5]. Ravindra Changala, "Enhancing Robotic Surgery Precision and Safety Using a Hybrid Autoencoder and Deep Belief Network Approach: Real-Time Feedback and Adaptive Control from Image Data",2024 8th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), ISSN: 2768-0673, DOI: 10.1109/I-SMAC61858.2024.10714701, October 2024, IEEE Xplore.
[6]. Ravindra Changala, "Swarm Intelligence for Multi-Robot Coordination in Agricultural Automation", 2024 10th International Conference on Advanced Computing and Communication Systems (ICACCS), ISSN: 2575-7288, DOI: 10.1109/ICACCS60874.2024.10717088, October 2024, IEEE Xplore.
[7]. D.-L Miholca, G. Czibula, and I. G. Czibula, "A novel approach for software defect prediction through hybridizing gradual relational association rules with artificial neural networks," " Information Sciences, vol. 441, pp. 152–170, 2018.
[8]. A. Alsaeedi and M. Z. Khan, "Software defect prediction using supervised machine learning and ensemble techniques: a comparative study," Journal of Software Engineering and Applications, vol. 38 12, no. 05, pp. 85–100, 2019.
[9]. Ravindra Changala, "Hybrid AI Approach Combining Decision Trees and SVM for Intelligent Tutoring Systems in STEM Education", 2024 10th International Conference on Advanced Computing and Communication Systems (ICACCS), ISSN: 2575-7288, DOI: 10.1109/ICACCS60874.2024.10717088, October 2024, IEEE Xplore.
[10]. Ravindra Changala, "Next-Gen Human-Computer Interaction: A Hybrid LSTM-CNN Model for Superior Adaptive User Experience", 2024 Third International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), ISBN:979-8-3503-6908-3, DOI: 10.1109/ICEEICT61591.2024.10718496, October 2024, IEEE Xplore.
[11]. Ravindra Changala, "Enhancing Early Heart Disease Prediction through Optimized CNN-GRU Algorithms: Advanced Techniques and Applications", 2024 Third International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), ISBN:979-8-3503-6908-3, DOI: 10.1109/ICEEICT61591.2024.10718395, October 2024, IEEE Xplore.
[12]. J. Li, P. He, J. Zhu, and M. R. Lyu, "Software defect prediction via convolutional neural network," in Proceedings of the 2017 IEEE International Conference on Software Quality, Reliability & Security. QRS, 2017, pp. 318–328.
[13]. Ravindra Changala, "Integration of IoT and DNN Model to Support the Precision Crop", International Journal of Intelligent Systems and Applications in Engineering, Vol.12 No.16S (2024) .
[14]. Ravindra Changala, Development of Predictive Model for Medical Domains to Predict Chronic Diseases (Diabetes) Using Machine Learning Algorithms and Classification Techniques, ARPN Journal of Engineering and Applied Sciences, Volume 14, Issue 6, 2019.
[15]. S. Jacob and G. Raju, "Software defect prediction in large space systems through hybrid feature selection and classification," International Arab Journal of Information Technology, vol. 14, no. 2,pp. 208–214, 2017.
[16]. Ravindra Changala, "Sentiment Analysis in Mobile Language Learning Apps Utilizing LSTM-GRU for Enhanced User Engagement and Personalized Feedback", 2024 Third International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), ISBN:979-8-3503-6908-3, DOI: 10.1109/ICEEICT61591.2024.10718406, October 2024, IEEE Xplore.
[17]. Ravindra Changala, "Image Classification Using Optimized Convolution Neural Network", 2024 Parul International Conference on Engineering and Technology (PICET), ISBN:979-8-3503-6974-8, DOI: 10.1109/PICET60765.2024.10716049, October 2024, IEEE Xplore.
[18]. Ravindra Changala, "Sentiment Analysis Optimization Using Hybrid Machine Learning Techniques", 2024 Parul International Conference on Engineering and Technology (PICET), ISBN:979-8-3503-6974-8, DOI: 10.1109/PICET60765.2024.10716049, October 2024, IEEE Xplore.
[19]. D.-L Miholca, G. Czibula, and I. G. Czibula, "A novel approach for software defect prediction through hybridizing gradual relational association rules with artificial neural networks," Information Sciences, vol. 441, pp. 152–170, 2018.
[20]. Ravindra Changala, Brain Tumor Detection and Classification Using Deep Learning Models on MRI Scans", EAI Endorsed Transactions on Pervasive Health and Technology, Volume 10, 2024.
[21]. Ravindra Changala, "Optimization of Irrigation and Herbicides Using Artificial Intelligence in Agriculture", International Journal of Intelligent Systems and Applications in Engineering, 2023, 11(3), pp. 503–518..
[22]. G. P. Bhandari and R. Gupta, "Machine learning based software fault prediction utilizing source code metrics," 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), pp. 40–45, 2018.
[23]. Ravindra Changala, "Advancing Surveillance Systems: Leveraging Sparse Auto Encoder for Enhanced Anomaly Detection in Image Data Security", 2024 International Conference on Data Science and Network Security (ICDSNS), ISBN:979-8-3503-7311-0, DOI: 10.1109/ICDSNS62112.2024.10690857, October 2024, IEEE Xplore.

[24]. Ravindra Changala, "Healthcare Data Management Optimization Using LSTM and GAN-Based Predictive Modeling: Towards Effective Health Service Delivery", 2024 International Conference on Data Science and Network Security (ICDSNS), ISBN:979-8-3503-7311-0, DOI: 10.1109/ICDSNS62112.2024.10690857, October 2024, IEEE Xplore.

[25]. Ravindra Changala, "Implementing Genetic Algorithms for Optimization in Neuro-Cognitive Rehabilitation Robotics", 2024 International Conference on Cognitive Robotics and Intelligent Systems (ICC - ROBINS), ISBN:979-8-3503-7274-8, DOI: 10.1109/ICC-ROBINS60238.2024.10533965, May 2024, IEEE Xplore.

[26]. A. Kumar, D. Prusti, I. S. Purusottam, and S. K. Rath, “Real time SOA based credit card fraud detection system using machine learning techniques,” 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), pp. 1–6, 2021.

[27]. N. Li, M. Shepperd, and Y. Guo, “A systematic review of unsupervised prediction techniques for software defect prediction,” Information and Software Technology, vol. 122.

[28]. Ravindra Changala, "Monte Carlo Tree Search Algorithms for Strategic Planning in Humanoid Robotics", 2024 International Conference on Cognitive Robotics and Intelligent Systems (ICC - ROBINS), ISBN:979-8-3503-7274-8, DOI: 10.1109/ICC-ROBINS60238.2024.10533937, May 2024, IEEE Xplore.

[29]. Ravindra Changala, "Biometric-Based Access Control Systems with Robust Facial Recognition in IoT Environments", 2024 Third International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS), ISBN:979-8-3503-6118-6, DOI: 10.1109/INCOS59338.2024.10527499, May 2024, IEEE Xplore.

[30]. Ravindra Changala, "Real-Time Anomaly Detection in 5G Networks Through Edge Computing", 2024 Third International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS), ISBN:979-8-3503-6118-6, DOI: 10.1109/INCOS59338.2024.10527501, May 2024, IEEE Xplore.

[31]. S. Wang, T. Liu, J. Nam, and L. Tan, “Deep Semantic Feature Learning for Software Defect Prediction,” IEEE Transactions on Software Engineering, vol. 46, pp. 1267–1293, 2020.

[32]. Ravindra Changala, "Enhancing Quantum Machine Learning Algorithms for Optimized Financial Portfolio Management", 2024 Third International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS), ISBN:979-8-3503-6118-6, DOI: 10.1109/INCOS59338.2024.10527612, May 2024, IEEE Xplore.

[33]. Ravindra Changala, "Integration of Machine Learning and Computer Vision to Detect and Prevent the Crime", 2023 International Conference on New Frontiers in Communication, Automation, Management and Security (ICCAMS), ISBN:979-8-3503-1706-0, DOI: 10.1109/ICCAMS60113.2023.10526105, May 2024, IEEE Xplore.

[34]. Ravindra Changala, "Controlling the Antenna Signal Fluctuations by Combining the RF-Peak Detector and Real Impedance Mismatch", 2023 International Conference on New Frontiers in Communication, Automation, Management and Security (ICCAMS), ISBN:979-8-3503-1706-0, DOI: 10.1109/ICCAMS60113.2023.10526052, May 2024, IEEE Xplore.

[35]. Ravindra Changala, “Optimizing 6G Network Slicing with the EvoNetSlice Model for Dynamic Resource Allocation and Real-Time QoS Management”, International Research Journal of Multidisciplinary Technovation, Vol 6 Issue 4 Year 2024, 6(4) (2024) 325-340.

[36]. H. K. Dam et al., “Lessons Learned from Using a Deep Tree-Based Model for Software Defect Prediction in Practice,” 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR), pp. 46–57, 2019.