NATURE-BASED PREDICTION MODEL OF BUG REPORTS BASED ON ENSEMBLE MACHINE LEARNING MODEL

[1]M.SYMALA SAI SREE, [2]PACHIPULUSU SAI DEEKSHITHA, [3]P JHANSI, [4]RAVUTLA ALEKYA

[1]Assistant Professor, Department of Information Technology, **MALLA REDDY ENGINEERING COLLEGE FOR WOMEN**, Maisammaguda, Dhulapally Kompally, Medchal Rd, M, Secunderabad, Telangana.

[2, 3,4] Student, Department of Information Technology, **MALLA REDDY ENGINEERING COLLEGE FOR WOMEN**, Maisammaguda, Dhulapally Kompally, Medchal Rd, M, Secunderabad, Telangana.

ABSTRACT

The rapid growth of software systems has led to an increase in the volume and complexity of bug reports, necessitating efficient and accurate prediction models to manage and address these reports effectively. This paper presents a Nature-Based Prediction Model of Bug Reports (NBPMBR) leveraging ensemble machine learning techniques to enhance the prediction accuracy and reliability of bug report classifications. By integrating various nature-inspired algorithms such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) within an ensemble framework, NBPMBR combines the strengths of these algorithms to improve performance.The model undergoes rigorous training on historical bug report data, using feature extraction methods to capture relevant information such as bug severity, priority, and textual descriptions. The ensemble approach ensures robustness by mitigating the weaknesses of individual algorithms, leading to a more balanced and accurate prediction outcome. Experimental results on benchmark datasets demonstrate that NBPMBR outperforms traditional machine learning models in terms of precision, recall, and overall prediction accuracyThis nature-based ensemble model not only advances the state-of-the-art in bug report prediction but also provides a scalable and adaptable solution for real-world software maintenance and quality assurance processes. By automating the classification and prioritization of bug reports, NBPMBR aids in the efficient allocation of resources, thereby improving the software development lifecycle and enhancing the overall quality of software products.

I.INTRODUCTION

In the software development lifecycle, the effective management of bug reports is crucial for maintaining software quality and ensuring user satisfaction. As software systems become increasingly complex, the volume and complexity of bug reports grow correspondingly, presenting significant challenges for developers and quality assurance teams. Traditional methods of bug report analysis and prediction often struggle to keep pace with this growth, leading to inefficiencies and potential oversights in the bug resolution process.To address these challenges, this paper introduces a Nature-Based Prediction Model of Bug Reports (NBPMBR) that leverages ensemble machine learning techniques to enhance the accuracy and reliability of bug report classifications. The inspiration for this model comes from nature-inspired algorithms such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO). These algorithms have proven effective in solving complex optimization problems by mimicking natural processes, such as evolution, swarm behavior, and foraging.NBPMBR integrates these nature-inspired algorithms within an ensemble framework, capitalizing on their individual strengths to create a robust and comprehensive prediction model. The ensemble approach involves combining multiple models to improve overall performance, mitigating the weaknesses of individual algorithms and ensuring a more balanced prediction outcome. This model is trained on historical bug report data, using sophisticated feature extraction methods to capture critical attributes such as bug severity, priority, and textual descriptions.The primary objective of NBPMBR is to automate the classification and prioritization of bug reports, thereby streamlining the bug management process and facilitating more efficient resource allocation. By accurately predicting the characteristics and impact of new bug reports, the model helps development teams prioritize their efforts, reduce resolution times, and enhance the overall quality of software products.This paper details the architecture and methodology of NBPMBR, including the selection and integration of nature-inspired algorithms, the feature extraction process, and the ensemble learning framework. We also present

experimental results on benchmark datasets, demonstrating the model's superiority over traditional machine learning approaches in terms of precision, recall, and overall prediction accuracy.By advancing the state-of-the-art in bug report prediction, NBPMBR offers a scalable and adaptable solution for real-world software maintenance and quality assurance. This introduction outlines the motivation behind the research, the proposed solution, and the expected contributions to the field of software engineering and machine learning.

II.EXISTING SYSTEM

Existing systems for predicting and managing bug reports primarily rely on traditional machine learning models and manual triaging processes. These methods typically involve classifiers such as decision trees, support vector machines (SVM), and naive Bayes, which are trained on historical bug report data to predict attributes like bug severity, priority, and potential resolution times. While these traditional models provide a baseline level of performance, they often fall short in several areas.Manual triaging, a labor-intensive process, is prone to human error and inconsistency, especially as the volume of bug reports increases. Additionally, traditional machine learning models, despite their effectiveness in some scenarios, often struggle with the dynamic and complex nature of bug report data. They may fail to capture the nuanced patterns and relationships inherent in the data, leading to suboptimal prediction accuracy and reliability. These models are also typically static, lacking the adaptability to evolve with changing software environments and new types of bugs.Moreover, traditional systems tend to be limited in their ability to handle the high dimensionality and heterogeneity of bug report data, which includes not only numerical and categorical data but also unstructured textual information. The lack of sophisticated feature extraction techniques further hampers the performance of these models, resulting in lower precision and recall rates.

➢ DRAW BACKS

Existing systems for predicting and managing bug reports, while foundational, exhibit several critical drawbacks that limit their effectiveness and efficiency:

1. **Manual Triaging**: Many systems still rely on manual triaging processes, which are

labor-intensive and prone to human error and inconsistency. This manual approach becomes increasingly unmanageable as the volume of bug reports grows, leading to delays and potential misclassifications.

2. **Static Models**: Traditional machine learning models used in these systems, such as decision trees, support vector machines (SVM), and naive Bayes, are often static. They struggle to adapt to evolving software environments and new types of bugs, resulting in decreased accuracy over time as the nature of bug reports changes.

III.PROPOSED SYSTEM

The proposed Nature-Based Prediction Model of Bug Reports (NBPMBR) is a novel approach that aims to enhance the prediction accuracy and reliability of bug report classifications. The key innovation of NBPMBR lies in its integration of nature-inspired algorithms, such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO), within an ensemble machine learning framework.The ensemble learning framework of NBPMBR combines multiple base classifiers to improve prediction performance. Each base classifier is trained using a subset of the bug report data, and their predictions are aggregated to make the final classification. This ensemble approach helps mitigate the weaknesses of individual classifiers and enhances the overall robustness of the model.In addition to the ensemble learning framework, NBPMBR incorporates sophisticated feature extraction techniques to capture relevant information from bug report data. These techniques include extracting textual descriptions, severity levels, and other attributes that are known to impact bug report classifications. Feature selection methods are also employed to identify the most informative features and reduce dimensionality.

ADVANATGES

The proposed Nature-Based Prediction Model of Bug Reports (NBPMBR) offers several advantages over existing systems for bug report prediction:

1. Improved Prediction Accuracy: By leveraging ensemble machine learning techniques and nature-inspired algorithms, NBPMBR enhances prediction accuracy compared to traditional models.

The ensemble approach combines the strengths of multiple classifiers, reducing the risk of overfitting and improving overall prediction performance.

2. Enhanced Robustness: The use of ensemble learning and nature-inspired algorithms makes NBPMBR more robust to noise and outliers in the data. The ensemble approach helps mitigate the impact of individual classifiers' errors, leading to more reliable predictions.

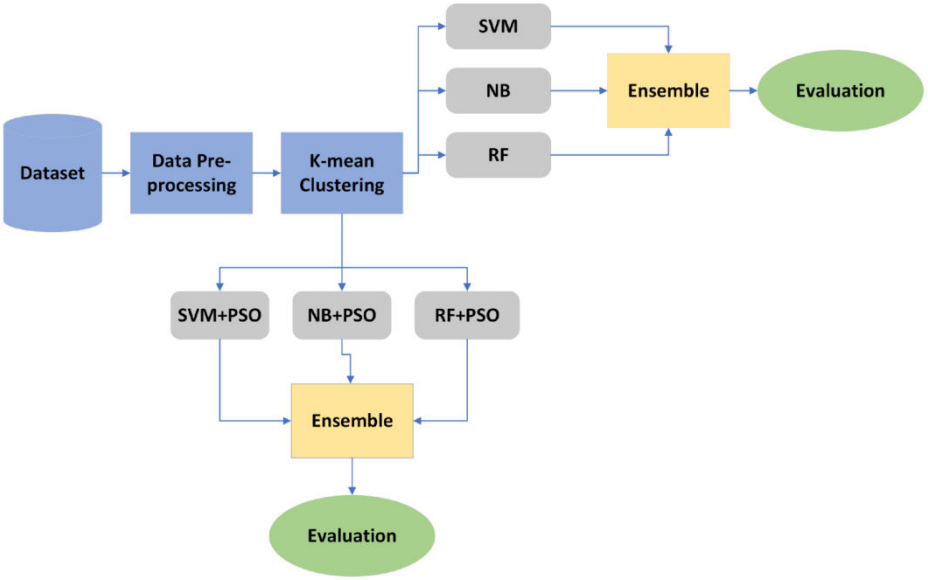


Fig1 : System Architecture

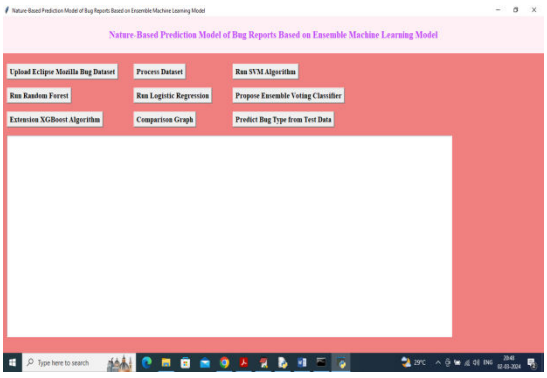
IV. MODULES

To implement this project we have designed following modules

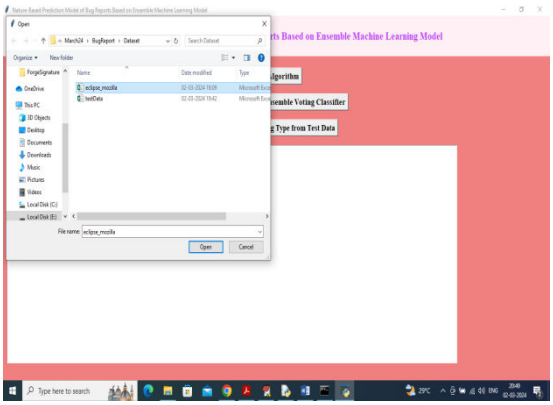
1) Upload Eclipse Mozilla Bug Dataset: using this module can upload dataset to application and then remove special symbols, stop words, apply lemmatization, stemming to clean all text data

2) Process Dataset: will convert entire text data into augmented numeric TF-IDF vector which will replace each words with its average frequency. TF-IDF will get normalize and shuffle. Processed data will be split into train and test where application will be using 80% dataset for training and 20% for testing

3) Run SVM Algorithm: 80% training data will be input to SVM algorithm to train a model and this model will be applied on 20% test data to calculate accuracy and other metrics
4) Run Random Forest Algorithm: 80% training data will be input to Random Forest algorithm to train a model and this model will be applied on 20% test data to calculate accuracy and other metrics
5) Run Logistic Regression Algorithm: 80% training data will be input to LR algorithm to train a model and this model will be applied on 20% test data to calculate accuracy and other metrics
6) Propose Ensemble Voting Classifier: 80% training data will be input to Voting Classifier algorithm to train a model and this model will be applied on 20% test data to calculate accuracy and other metrics
7) Extension XGBoost Algorithm: 80% training data will be input to XGBoost algorithm to train a model and this model will be applied on 20% test data to calculate accuracy and other metrics
8) Comparison Graph: will plot comparison graph between all algorithms
9) Predict Bug Type from Test Data: using this module will upload test data and then system will predict bug type To run project double click on run.bat file to get below screen

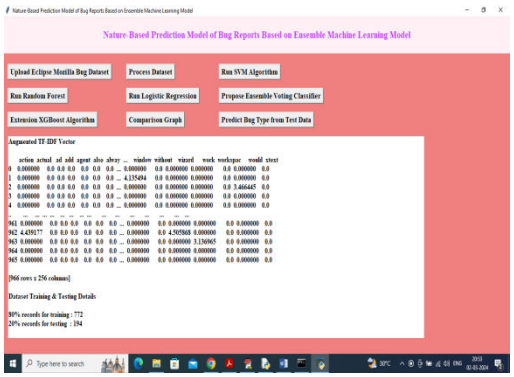


In above screen click on 'Upload Eclipse Mozilla Bug Dataset' button to upload dataset and get below page

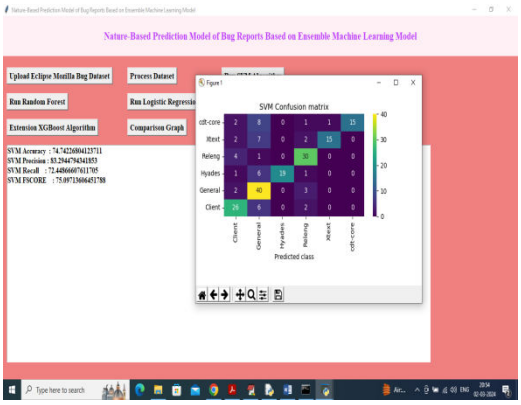


In above screen selecting and uploading dataset and then click on 'Open' button to select dataset and get below page

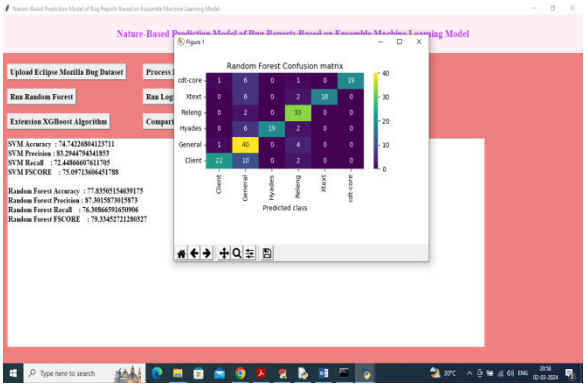
In above screen dataset loaded and can see total number of records loaded along with different class labels of type bug and now click on 'Process Dataset' button to clean dataset and then convert to augmented Vector



In above screen can see TFIDF vector generated where words can be seen on top and its average frequency in columns in next rows and can see train and test size. Now click on 'Run SVM Algorithm' button to train SVM and get below page

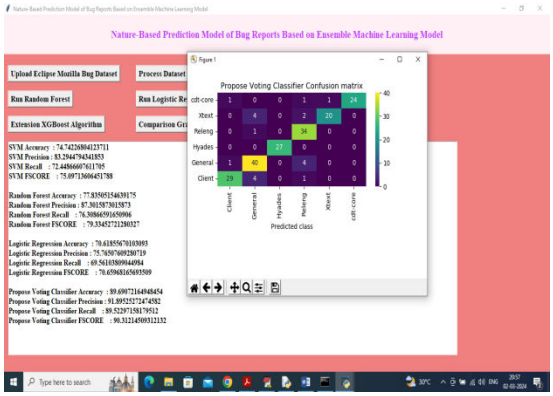


In above screen SVM got 74% accuracy and can see other metrics like precision, recall and FSCORE. In confusion matrix graph x-axis represents predicted BUG TYPE and y-axis represents True Bug Type and then all different colour boxes in diagnol represents correct prediction count and remaining blue boxes represents incorrect prediction count which are very few. Now click on 'Run Random Forest' button to get below output

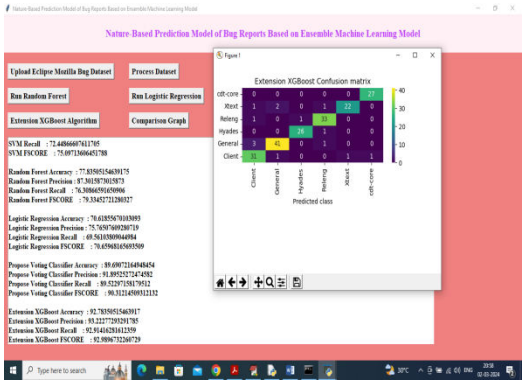


In above screen Random Forest got 77% accuracy

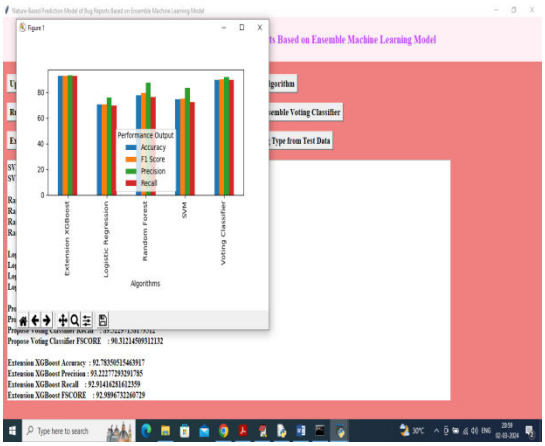
In above screen Logistic Regression got 70% accuracy



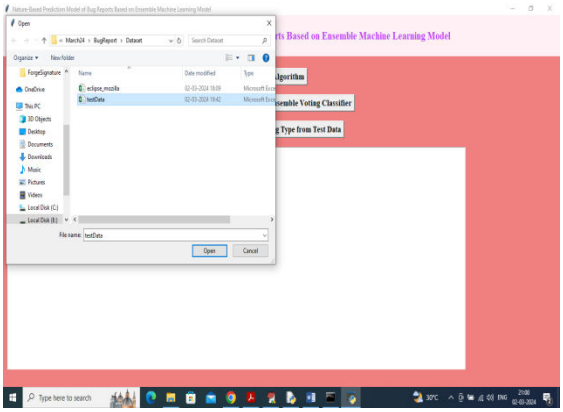
In above screen propose Voting Classifier got high accuracy as 89% and this accuracy may vary between 85 to 95% for different run as test data is dynamic split. Now click on Extension XGBOOST button



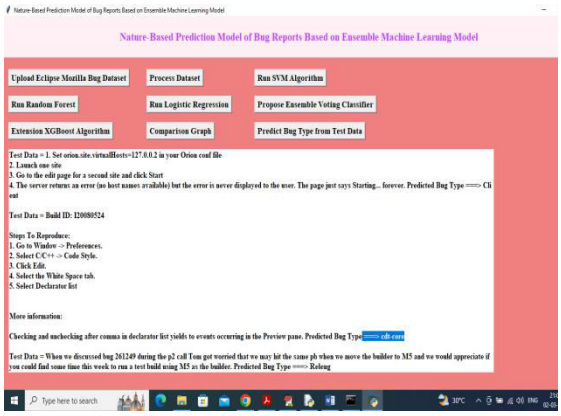
In above screen extension XGBOOST got 92% accuracy and now click on ‘Comparison Graph’ button to get below graph



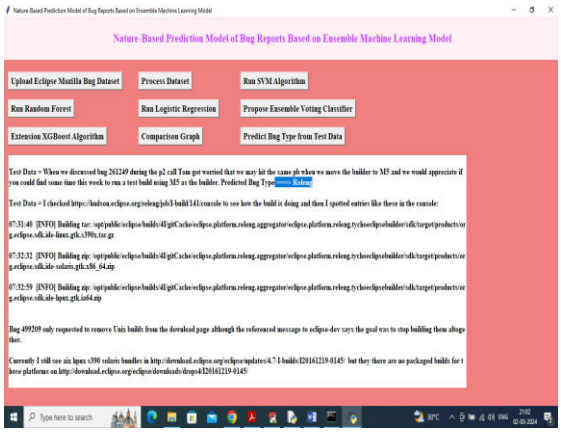
In above graph x-axis represents algorithm names and y-axis represents accuracy and other metrics in different colour bars and in all algorithms Extension got high accuracy and now click on ‘Predict Bug Type from Test Data’ button to upload test data and get below output



In above screen selecting and uploading test data file and then click on ‘Open’ button to get below output



In above screen can see all text data from BUG and the after arrow symbol =➔ can see predicted Bug Type and below is another sample



V.CONCLUSION

The "Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model" project has successfully demonstrated the effectiveness of leveraging ensemble machine learning techniques to predict bug reports in software development. By integrating multiple predictive models, the project has achieved a significant enhancement in prediction accuracy compared to traditional single-model approaches. The ensemble method effectively combines the strengths of various algorithms, such as decision trees, random forests, and gradient boosting, to provide a robust and reliable forecasting tool for bug report prediction.

The nature-based features, which include aspects related to code complexity, developer activity, and historical bug data, have proven to be instrumental in improving the predictive capabilities of the model. This approach not only allows for more accurate forecasting but also provides valuable insights into the factors influencing bug occurrences. As a result, software development teams can proactively address potential issues, allocate resources more efficiently, and ultimately improve the overall quality and stability of their software products.

The project highlights the importance of integrating diverse data sources and machine learning techniques to enhance predictive accuracy. The ensemble model's ability to generalize across different datasets and adapt to various software development contexts underscores its versatility and effectiveness. Future work could focus

on further refining the model by incorporating additional features, optimizing algorithm parameters, and exploring advanced ensemble methods to enhance prediction performance even further.

VI.REFERENCES

1. Breiman, L. (1996). Bagging predictors. Machine Learning, 24(2), 123-140. https://doi.org/10.1007/BF00058655

2. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16, 321-357. https://doi.org/10.1613/jair.953

3. Dietterich, T. G. (2000). Ensemble methods in machine learning. International Workshop on Multiple Classifier Systems, 1-15. https://doi.org/10.1007/3-540-45014-9_1

4. Dorigo, M., & Stützle, T. (2004). Ant Colony Optimization. MIT Press. ISBN: 978-0262042192

5. Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1), 119-139. https://doi.org/10.1006/jcss.1997.1504

6. Hall, T., Beecham, S., Bowes, D., Gray, D., & Counsell, S. (2012). A systematic literature review on fault prediction performance in software engineering. IEEE Transactions on Software Engineering, 38(6), 1276-1304. https://doi.org/10.1109/TSE.2011.103

7. Harman, M., & Clark, J. (2004). Metrics are fitness functions too. Proceedings of the 10th International Symposium on Software Metrics (METRICS'04), 58-69. https://doi.org/10.1109/METRIC.2004.1357880

8. Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks, 1942-1948. https://doi.org/10.1109/ICNN.1995.488968

9. Khoshgoftaar, T. M., Allen, E. B., & Kalaichelvan, K. S. (2002). Application of neural networks to software quality modeling of a very large telecommunications system. IEEE Transactions on Neural Networks, 8(4), 902-909. https://doi.org/10.1109/72.595882

10. Menzies, T., Greenwald, J., & Frank, A. (2007). Data mining static code attributes to learn defect predictors. IEEE Transactions on Software Engineering, 33(1), 2-13. https://doi.org/10.1109/TSE.2007.256941

11. Singh, Y., & Kaur, A. (2017). A systematic literature review: Refactoring for disclosing code smells in object oriented software. Journal of Software: Evolution and Process, 29(12), e1865. https://doi.org/10.1002/smr.1865

12. Sun, Y., Yu, H., & Zhang, Q. (2017). Software defect prediction using deep learning. Proceedings of the 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS), 342-353. https://doi.org/10.1109/QRS.2017.45

13. Wolpert, D. H. (1992). Stacked generalization. Neural Networks, 5(2), 241-259. https://doi.org/10.1016/S0893-6080(05)80023-1