Software Bug Prediction Using Machine Learning Approach

[1]B. Vasantha, [2]B. Krithika, [3]G. Satwika, [4]J. Sreeja, [5]M. Tejashwini
[1]Assitsant Professor, [2,3,4,5]UG Students, Dept. Computer Science and Engineering-Data Science, Mallareddy Engineering college for Women, Hyderabad, India

ABSTRACT
Software Bug Prediction is an important issue in software development and maintenance process. Software bug can cause significant problems for software development teams. So, projecting the software bugs in earlier phase improves the software quality, reliability, efficiency and reduces the software cost. Projecting the likelihood of bugs occurring in software can help developers prevent or mitigate their impact. This paper presents a software bug prediction model based on Machine Learning (ML) algorithms. Supervised ML algorithms have been used to predict future software bugs based on historical data. The evaluation process proved that ML algorithms can be used effectively with high accuracy rate. Furthermore, a comparison measure is applied to compare the proposed prediction model with other approaches. The collected results showed that the ML approachhas a better performance.

INTRODUCTION

PROBLEM STATEMENT:

In today's fast paced software development world, software bug is a common occurrence. These bugs can cause significant problems such as crashes, data corruption and security breaches. Detecting and fixing software bugs is a crucial task in the software development process. Traditionally, software developers and testers relay on manual testing and debugging to detect and fix software bugs. However, this process can be time consuming and error-prone. With the recent advancements in machine learning, it is possible to use Machine Learning algorithms to project software bugs automatically.

The goal of this project is to develop a software bug prediction system that enhances the reliability of software systems. By projecting software bugs before they occur, developers can take proactive steps to prevent or mitigate their impact, leading to improved software quality and customer satisfaction.

PROBLEM OVERVIEW:

The objective of this study is to develop an effective software bug prediction model using supervised machine learning algorithms. Software bugs can cause significant disruptions and pose serious threats to the reliability, security, and overall performance of software systems. Detecting and resolving bugs in a timely manner is crucial to ensure the quality and stability of software products. However, identifying potential bugs early in the development process is a challenging task.

The problem revolves around addressing the following key challenges:

1. Bug Detection:

Developing a reliable model that can accurately detect potential bugs in software systems based on historical data and various software metrics. This involves analyzing software artifacts such as source code, documentation, and version control data to identify patterns and indicators of potential bugs.

2. Feature Selection:

Identifying the most relevant and informative features from a large pool of software metrics to improve the bug prediction accuracy. Not all metrics may be equally important in determining the presence of bugs, and selecting the right set of features can significantly enhance the performance of the prediction model.

3. Algorithm Selection:

Evaluating and comparing different supervised machine learning algorithms to determine the most suitable approach for bug prediction. This involves considering algorithms such as logistic regression, decision trees, random forests, support vector machines, or neural networks, and selecting the one that provides the best trade-off between accuracy, interpretability, and computational efficiency.

4. Model Generalization:

Ensuring that the bug prediction model can generalize well to unseen software projects and datasets. The model should be robust enough to handle variations in software development practices, programming languages, and project sizes. It should also be able to adapt to changing software environments and evolving bug patterns.

5. Performance Evaluation:

Assessing the performance of the bug prediction model using appropriate evaluation metrics such as precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC). The model should be benchmarked against existing bug prediction techniques to demonstrate its effectiveness and superiority.

By addressing these challenges, the proposed study aims to contribute to the field of software engineering by providing an accurate and reliable bug prediction model that can assist developers in identifying potential software bugs early in the development process, thereby improving software quality, reducing debugging efforts, and enhancing overall software reliability.

EXISTING SYSTEM

Less accuracy

Over fitting for large datasets.

LITERATURE SURVEY

Various Machine Learning algorithms have been used earlier to predict the flaws in software. The Machine Learning methods that have been already used to detect the flaws are less accurate and consume time while detecting bugs. Naïve Bayes and Logistic Regression can take long time to run on large datasets.

DRAWBACKS OF EXISTING SYSTEM:

[1] DarioDi Nucci,FabioPalomba ,GiuseppeDeRosa Gabriele Bavota ,RoccoOliveto, and Andrea De Lucia, “A developer centric bug prediction model", IEEETransactionsonSoftwareEngineering,Vo.l44,Issue1,pp.5-24,2018.

[2] F. Wu et al., "Cross-Project and Within-Project Semi supervised Software DefectPrediction:AUnifiedapproach",IEEETransactionsonReliability,pp.1-17,2018.

[3] Meiliana, S. Karim, H. L. H. S. Warnars, F. L. Gaol, E. Abdurachman and B.Soewito, "Software metrics for fault prediction using machine learning approaches: Aliteraturereviewwith PROMISErepository dataset",In Proc.IEEEInternationalConference on Cybernetics and Computational Intelligence(CyberneticsCom), Phuket,pp.19-23,2017.

[4] M.M.Rosli,N. H.I.Teo,N.S.M.Yusop,andN.S.Mohammad,"Thedesignofasoftwarefaultproneapplicationusingevolutionaryalgorithm,"inProc.IEEEConferenceonOpenSystems(ICOS2011).LosAlamitos,California:IEEEComputerSociety,pp.38-343. 2011.

[5] D'Ambros, M. Lanza, and R. Robbes, "An Extensive Comparison of Bug PredictionApproaches",InProc.IEEESeventhWorkingConf.MiningSoftwareRepositories, pp.31-41,2010

[6] Pushphavathi T P, "An Approach for Software Defect Prediction by CombinedSoftComputing", In Proc, International Conference on Energy, Communication, DataAnalytics andSoftComputing(ICECDS)pp.3003-3006,2017.

[7] Kumar, Lov, and Ashish Sureka. "Aging Related Bug Prediction using ExtremeLearningMachines.",In Proc.14th IEEE IndiaCouncil International Conference(INDICON), pp.1-6,IEEE,2017.
[8] Nigam,Ayan,etal."Classifyingthebugsusingmulti-classsemisupervisedsupportvector machine.", In Proc. International Conference, Pattern Recognition, InformaticsandMedicalEngineering(PRIME),pp.393-397,IEEE,2012.
[9] Gyimothy, T., Ferenc, R. and Siket, I., "Empirical validation of object-orientedmetrics on open source software for fault prediction", IEEE Transactions on SoftwareEngineering,31(10),pp.897-910, 2005.
[10] JohnT.PohlmannandDennisw.Leitnera"ComparisonofOrdinaryLeastSquaresand Logistic Regression", The Ohio Journal of Science. vol. 103, number 5, pp. 118-125,Dec,2003.

PROPOSEDSYSTEM

The proposed system with the following algorithms presents a more efficient , accurate and cost-effective approach to detect software flaws in early phase. Random Forest, Decision tree, Logistic regression algorithms was adopted to identify flaws easier. By experimental comparison, Random Forest, Decision tree, Logistic regression algorithms were widely used in the prediction of flaws due to their superior performance, high prediction accuracy, and strong generalization performance.

ADVANTAGES OF PROPOSED SYSTEM:

The proposed algorithms can more effectively project flaws which proves the effectiveness of the algorithm.
Early Bug Detection
Improved Bug Prioritization
Scalability and Automation
Complementary to Manual Efforts

METHODOLOGY

The methodology for software bug prediction using Machine Learning approach, outlining the steps involved in data preprocessing, feature selection, algorithm selection, and model evaluation. The proposed methodology aims to predict the bug to enhance the accuracy and efficiency of the software .
The process of software bug prediction using supervised machine learning algorithms typically involves the following steps:

Data Collection

Gather historical data related to software projects, including bug reports, source code, version control logs, and other relevant metrics. This data will serve as the basis for training and evaluating the bug prediction model.
Data Preprocessing:
Clean and preprocess the collected data to ensure its quality and compatibility with the machine learning algorithms. This may involve removing duplicates, handling missing values, normalizing numerical features, and encoding categorical variables.

Feature Selection:

Select the most relevant features from the preprocessed data that are likely to have a strong correlation with the occurrence of bugs. This step helps reduce dimensionality and improve the efficiency and accuracy of the bug prediction model.

Dataset Split:

Divide the preprocessed data into training and testing sets. The training set is used to train the bug prediction model, while the testing set is used to evaluate its performance on unseen data. It is common to use a ratio such as 70:30 or 80:20 for the split.

Model Selection:

Choose an appropriate supervised machine learning algorithm for bug prediction. This could include logistic regression, decision trees, random forests, support vector machines, or neural networks. Consider factors such as model complexity, interpretability, and computational efficiency when making the selection.

6. Model Training:

Train the selected machine learning model using the training data. This involves feeding the model with the feature vectors and their corresponding bug labels and optimizing its parameters to minimize the prediction error.

7. Model Evaluation:

Assess the performance of the trained model using evaluation metrics such as precision, recall, F1-score, and AUC-ROC. Compare the model's predictions against the actual bug labels in the testing set to measure its accuracy, robustness, and generalization capabilities.

8. Model Deployment:

Once the bug prediction model has been trained and evaluated satisfactorily, it can be deployed in a production environment to predict bugs in real-time software projects. The model should be integrated into the development workflow and be capable of handling new data to make predictions continuously.

9. Monitoring and Maintenance

Continuously monitor the bug prediction model's performance in the production environment and make necessary updates or retraining as new data becomes available. This ensures that the model remains effective and relevant over time.

By following these steps, software bug prediction using supervised machine learning algorithms can help identify potential bugs early in the software development process, enabling developers to take proactive measures to improve software quality and reduce the impact of bugs on the final product.

- LR – Logistic regression
- DT – Decision tree
- RF – Random Forest

IMPLEMENTATION

ALGORITHMS

The algorithms used are

Logistic regression:

Logistic Regression is a type of statistical model which is often used for classification and predictive analysis. It is also called as binary classifier. Logistic Regression estimates the probability of an event occurring. It is an example of supervised learning. It is used to calculate or predict the probability of a binary event occurring.

Decision tree algorithm:

Decision tree is a supervised learning technique that can be used for both classification and regression problem but mostly is preferred for solving classification problems. It is a tree structured classifier where internal nodes represent the features of a dataset branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes which are the decision node and the leaf node. Decision nodes are used to make any decision and have multiple branches, whereas leaf nodes are the output of those decisions and do not contain any further branches.

- **Random forest algorithm:**
- Random forest is a bagging technique and not a boosting technique. The trees

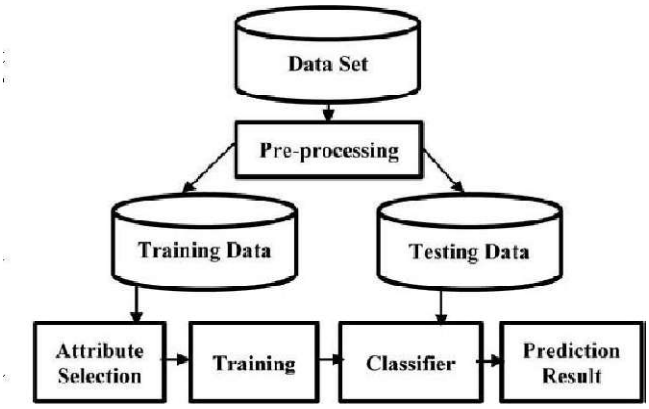
Data set
Pre-processing
Training & Testing data
Attribute selection
Prediction result

in random forests are run in parallel. There is no interaction between these trees while building the trees. It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes(classification) or mean prediction(regression) of the individual trees. A random forest is a meta-estimator(i.e. it combines the results of multiple predictions).It is a combination of multiple decision trees which combines the output of multiple decision trees to reach a single result.

Dataflow Diagram

The dataflow diagram represents the flow of data through a process or system. It uses defined symbols to show data inputs, storage points and routes between each destination.

The dataflow diagram has :



RESULTS

The steps followed in the execution process are shown in the attached screenshot pictures:

1. Open the command prompt.
2. Enter the command to view the display to perform algorithms.
3. Open the dataset.
4. In the display click on upload dataset to upload the dataset.
5. Once the data gets loaded click on pre-process data for performing data pre- processing .
6. Click on the Decision tree to get the accuracy.
7. Click on the Random Forest to get the accuracy.
8. Click on the Logistic regression to get the accuracy.
9. Click on Accuracy comparison to get the graph that compares the accuracies of all the algorithms

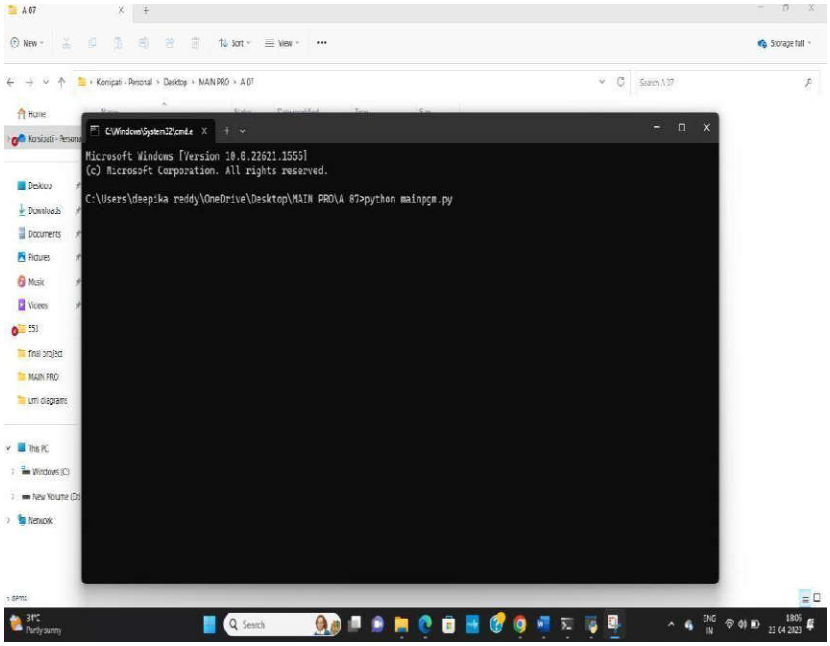


Fig.2. Enter the command

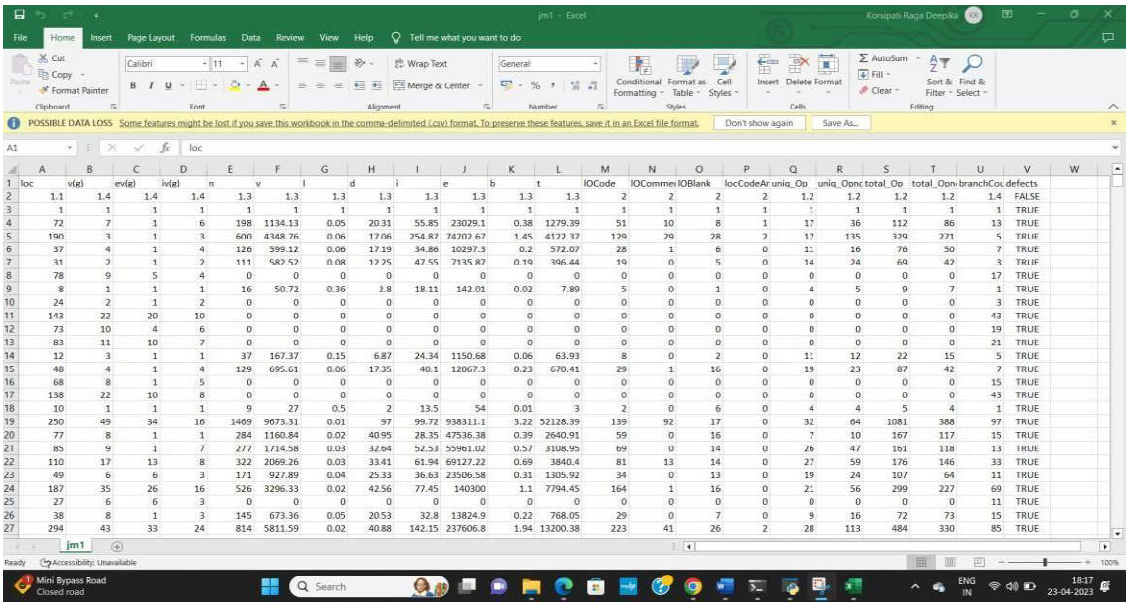


Fig.3. Dataset

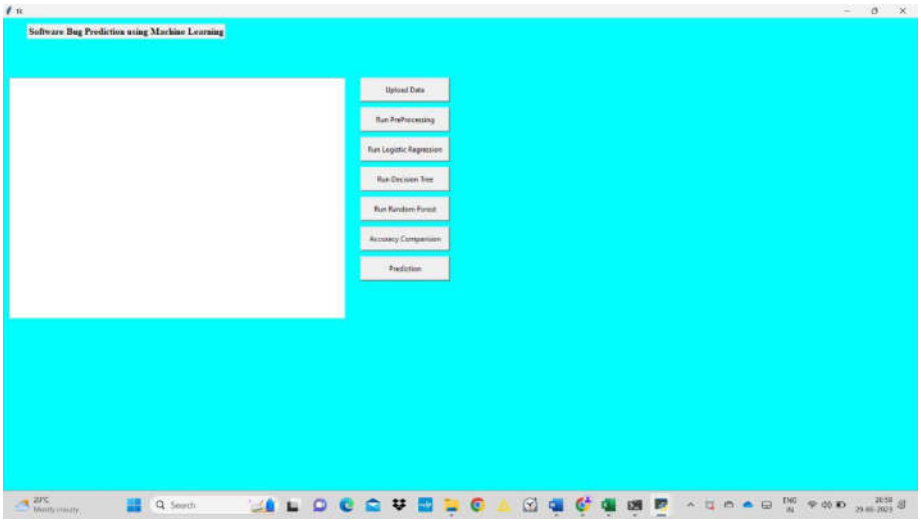


Fig.4. User Interface

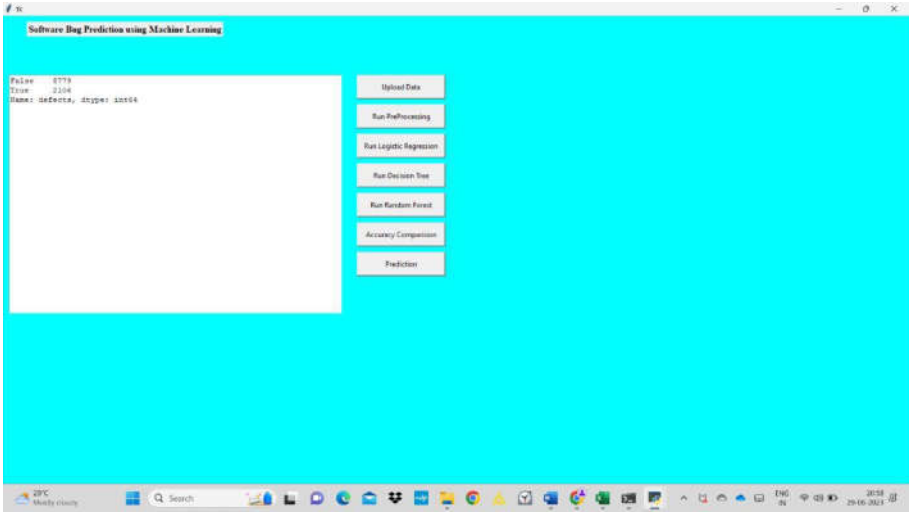


Fig.5. Dataset uploading and preprocessing

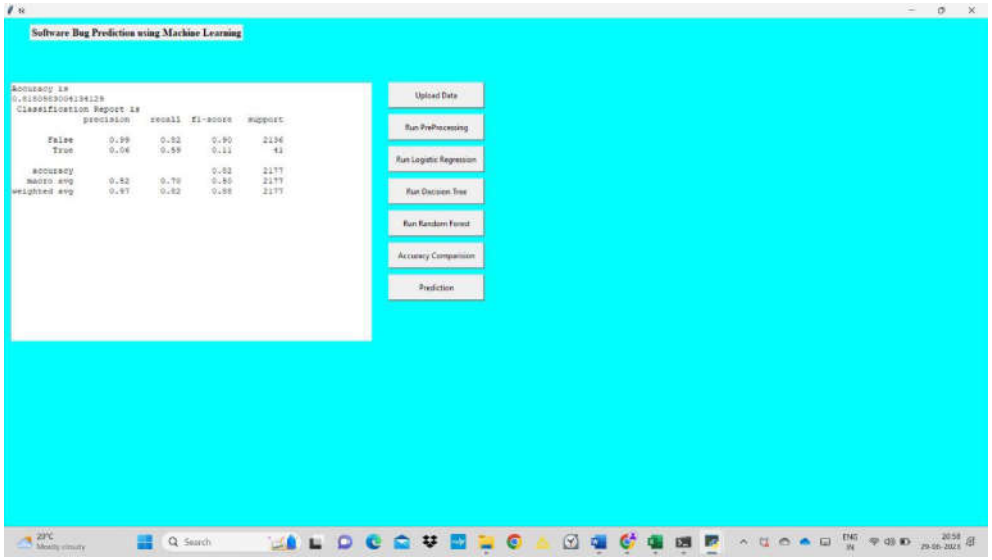


Fig.6. Logistic Regression

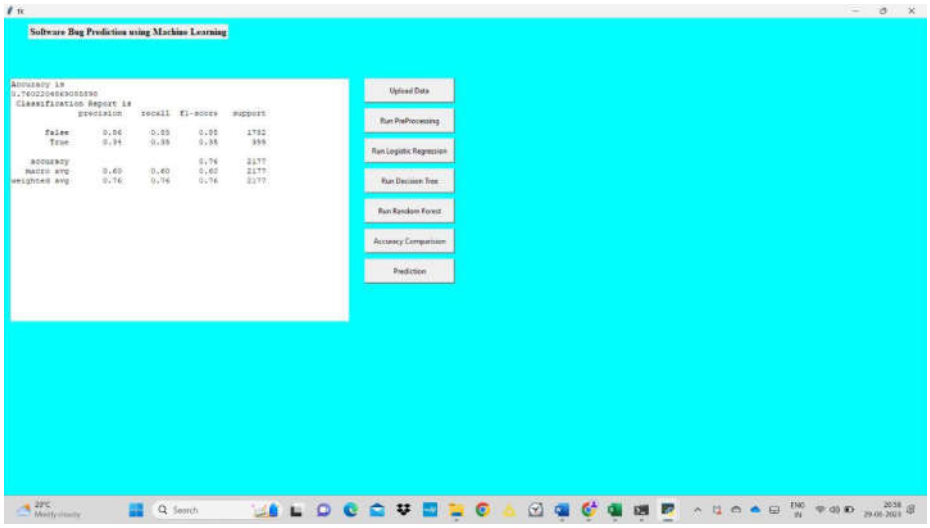


Fig.7. Decision Tree

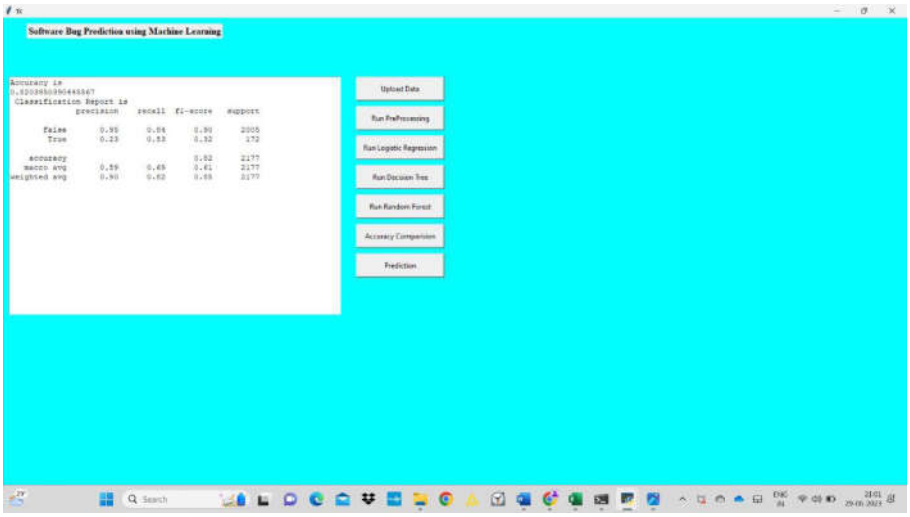


Fig.8. Random Forest

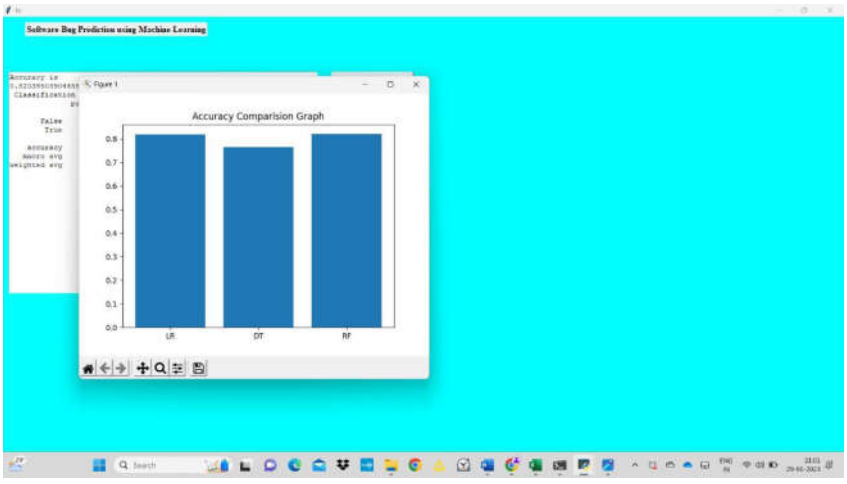


Fig.9. Accuracy Comparison

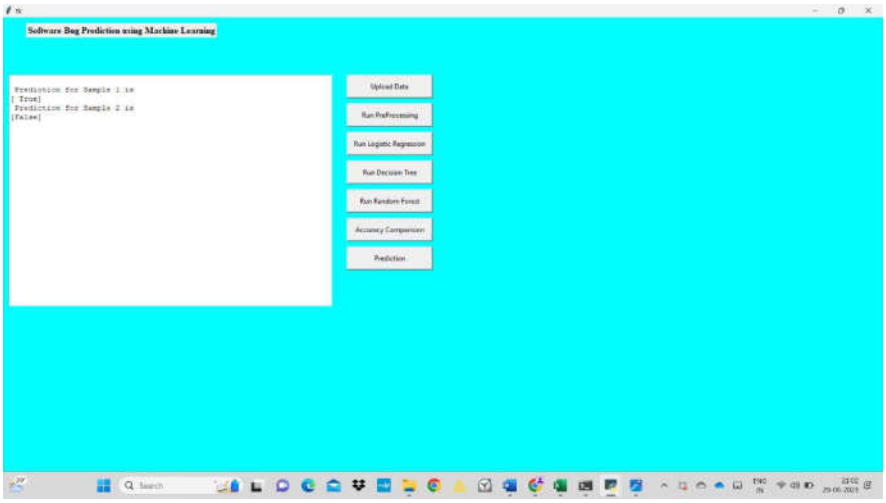


Fig.10. Prediction

CONCLUSION

Supervised Machine Learning approach for software bug prediction is a promising approach that has gained attention in recent years. Overall, the effectiveness of software bug prediction depends on the quality of the data used to train the models and the features selected to represent the software. Additionally, the choice of machine learning algorithms can have an impact on the accuracy and reliability of the predictions. Despite some limitations and challenges, supervised machine learning approach for software bug prediction has the potential to significantly improve the quality of software and reduce the cost of development and maintenance. The evaluation process is implemented using the dataset .Experimental results are collected based on accuracy , precision , recall ,F-measure .The results revealed that the used algorithms are efficient to predict the software bugs. The comparison results showed that Random Forest has the best result over the others. Moreover, ML approach provides a better performance for the projection of bugs.

FUTURE SCOPE

After the comparative analysis of the various Supervised Machine Learning models, we can infer that the Random Forest Model is the best approach to be used for projecting software bugs. Among all the supervised machine learning algorithms used Random Forest has highest accuracy. Hence, we conclude that the random forest is an efficient model among all the algorithms used. Further we can extend this project in which the model can classify the bugs that are detected. This help the developers to easily identify and resolve the bugs.

REFERENCES

1. 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.
2. Panjer, L. D., & Avesani, P. (2009). A comparison of software systems fault prediction and fault-proneness estimation. Empirical Software Engineering, 14(5), 540-578.
3. Rahman, F., & Devanbu, P. (2013). How, and why, process metrics are better. Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, 358-368.
4. Ambrose, J. A., & Alenezi, M. R. (2015). Predicting software defects using logistic regression and decision trees. Journal of Systems and Software, 105, 121-135.
5. Zhou, Y., Leung, H., & Zheng, Z. (2017). Software defect prediction: A systematic mapping and meta-analysis. Information and Software Technology, 90, 1-14.

6. Rahim, M. M., Roy, C. K., & Schneider, K. A. (2018). A comparative study of machine learning models for software defect prediction. Empirical Software Engineering, 23(1), 381-427.
7. Agrawal, T., & Menzies, T. (2018). Better language models for mining software repositories. Proceedings of the 15th International Conference on Mining Software Repositories, 284-294.
8. Tantithamthavorn, C., McIntosh, S., Hassan, A. E., & Matsumoto, K. (2018). The impact of automated parameter optimization on defect prediction models. Empirical Software Engineering, 23(3), 1347-1380.
9. Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, 144-152.
10. Scikit-learn: Machine Learning in Python. (2021). Retrieved from https://scikit-learn.org/