IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 3, December 2023

Enhancing Software Quality: Bug Identification with SMLT

Jagan G and Akila K
Department of Computer Science and Engineering
SRM Institute of Science and Technology, Vadapalani, Chennai, Tamil Nadu, India

Abstract: *A software bug refers to a mistake, imperfection, or malfunction in a computer program or system that results in an inaccurate or unexpected outcome, or leads it to act in an unanticipated manner. The majority of defects are caused by faults and flaws that are created in the operating systems and components that these programs employ, or in the design or source code of the programs themselves. Incorrect code produced by compilers is the cause of a few. A program is considered to be buggy if it has a large number of bugs or if those bugs significantly impair its functionality. Typically, a logical error made by the coder results in bugs. The full data set will be analyzed using the supervised statistical machine learning process (SMLT) to gather various data, including variable identification, uni-, bi-, and multivariate analysis, treatments for missing values, validation of the data, data cleaning and preparation, and data visualization. to provide a machine learning-based technique that compares supervised categorization machine learning algorithms to determine which software bugs are present and which ones are not.*

Keywords: Machine Learning, Data Visualization, SMLT

I. INTRODUCTION

A software issue is an error, flaw, or fault in a system or program on a computer that results in incorrect or unexpected behaviour or causes the program or system to act in ways that are not intended. The majority of defects are caused by faults and flaws that are either in the source code or design of a program, or in the parts and operating systems that these programs employ.

Machine learning uses historical data to forecast future events. Artificial intelligence (AI) in the form of machine learning (ML) gives computers the capacity to learn without explicit programming. The creation of programs for computers that can adapt to new data and the fundamentals of machine learning, such as the execution of a straightforward Python machine learning algorithm, are the main goals of machine learning. Software engineers can specialize in developing code that is prone to defects by using the datasets, techniques, and frameworks that are often produced by recent studies on software defect prediction. This improves software quality and optimizes resource usage. A thorough picture of the current status of defect prediction studies is lacking because there are many published, sophisticated, and divergent software defect predictions datasets, methodologies, and frameworks.

II. LITERATURE REVIEW

In Arun Balaji, S., Gowtham, J., Amirtharaj, B., & Bala Abirami, B.(2021). A software problem is an error, flaw, or fault in an application or system for computers that results in incorrect or unexpected behavior or causes the program or system to act in ways that are not intended. The majority of defects are caused by faults and flaws that are either in the source code or design of a program, or in the parts and operating systems that these programs employ. Some are brought on by compilers generating inaccurate code. A program is considered to be buggy if it has a large number of bugs or bugs that significantly impair its functionality. Typically, a logic error made by the coder results in bugs. to provide a machine learning-based technique that compares supervised categorization machine learning algorithms to determine which software bugs are present and which ones are not.

In David Paterson, Jose Campos, Rui Abreu (2019). There is a lot of study on test case prioritizing as a way to speed up the process of finding software regressions. Despite the fact that a wide range of strategies have been created and

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 3, December 2023

assessed, earlier tests have revealed that they are ineffective at ranking test suites in order to identify actual errors. This paper proposes a test case prioritizing strategy centered around defect prediction, a method that estimates the probability that a given Java class will include a problem by analyzing code factors such as the number of authors and modifications. Defect prediction should prioritize tests to quickly find faults in a class if it can correctly identify which class is more likely to have bugs. This makes intuitive sense. We performed an actual evaluation comparing this paper's technique, named G-clef, versus eight current test case prioritizing strategies using six real-world Java programs with 395 real defects.

In Aurélien Francillon, Sam L. Thomas, and Andrei Costin (2021). This chapter aims to familiarize the reader with the field of defect finding in embedded systems, the backbone of the Worldwide Web of Things. There are some unique characteristics of embedded software that set it apart from general-purpose software. Embedded systems in particular are frequently left unattended, have lesser defensive levels, and are more vulnerable to software assaults. Yet, since they are so "opaque" and because embedded and bespoke software frequently interacts with peripherals and hardware, it is more challenging to analyze their security. These variations affect our capacity to identify software defects in these kinds of systems. This chapter addresses how vulnerabilities in software may be found at various phases of the application life-cycle, such as during development, during component integration, during testing, during device deployment, or in the field by outside parties.

In Romi Satria Wahono (2015). Software engineers can concentrate on development efforts related to defect-prone code by using the datasets, techniques, and frameworks that are often produced by recent research on software defect prediction. This improves software quality and optimizes resource utilization. A thorough picture of the state of defect prediction work at the moment is lacking since there are many published, complicated, and diverse software fault prediction datasets, methodologies, and frameworks. The purpose of this evaluation of the literature is to determine and evaluate the research patterns, datasets, techniques, and frameworks that have been employed in software predicting defects between 2000 and 2013.Classification methods account for 77.46% of research papers, estimate methods for 14.08% of studies, and clustering and association approaches for 1.41% of studies. Furthermore, 64.79% of the study papers made use of public datasets, whereas 35.21% made use of private datasets. Software flaws have been predicted using nineteen different approaches. Seven of the 19 approaches are the most often used for predicting software defects. Researchers suggested boosting algorithms, feature selection, ensemble some artificial intelligence approaches, and parameter modification for specific classifiers as ways to increase the accuracy using machine learning classifiers for software fault prediction.

In Praveen, A. K., Harsita, R., Murali, R. D., & Niveditha, S. (2023). a fraudulent checking program that uses machine learning (ML) techniques (Random Forest Classifiers, logarithm regression, supported vector machines, and the XGBoost Classifiers) and natural language processing (NLP) to identify phony job listings. These methods will be contrasted before being included into a team model that our task detector uses. The goal is to anticipate as accurately as possible using neural networks for actual or fictitious job predictions. Supervised learning techniques (SMLT) are utilized in dataset analysis to gather diverse information, including variable identification, management of missing values, and data validity analysis. The complete dataset is processed, including data preparation and cleaning.

III. RESEARCH METHODOLOGY

Combining the benefits of software testing with machine learning, the SMLT model offers a versatile method for identifying software defects. This process consists of several important phases. First of all, it includes gathering and preprocessing information from a variety of sources, such as system logs, bug reports from the past, and code repositories. The SMLT model, which was originally trained on an extensive collection of software artifacts including the associated bug-related data, is then given this data. Once deployed, the SMLT model uses its machine learning skills to identify probable bug-prone locations by doing a thorough study of the code base and related artifacts. Finding trends, abnormalities, and disparities that could point to the existence of software flaws is part of this investigation. Additionally, the model may use this data to cross-reference past bug data and forecast with confidence which code segments are likely to cause problems.

The suggested approach next includes an automated testing stage in which the model creates test cases using input data to verify the predictions it makes about bugs. These tests assist in determining the severity existing each problem and

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 3, December 2023

verifying the existence of defects. Upon detection of a defect, developers receive immediate notification, and comprehensive reports with recommended changes are produced to enable effective debugging and resolution.

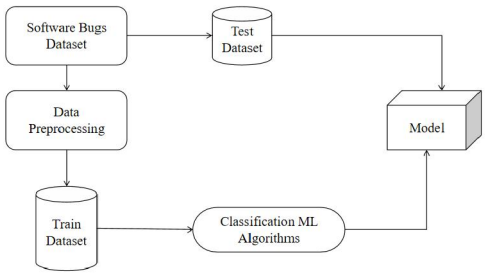

Fig 1. Architecture of Proposed model

1) Data Wrangling:

The data will be loaded into this area of the report, cleaned, and then trimmed and prepared for analysis. Verify the document's procedures and provide justification for any cleaning decisions. loading the specified dataset while importing the library packages. Analyzing the parameter identification by data type and form, assessing duplicate and missing values, etc. while fine-tuning models and techniques to make the most of validation and test datasets while evaluating your models, a validation dataset is a sample of data withheld form training your model which is utilized to offer an estimate of model competence. Renaming the provided dataset, removing a column, and other actions are examples of data cleaning and preparation for uni-, bi-, and multi-variate analysis. Data cleaning procedures and methods differ depending on the kind of dataset.

2) Data collection:

The training collection and test set comprise the data set that was gathered to forecast the provided data. To divide both the Training set with Assessment set, 7:3 ratios are typically used. The training set is subjected to the Data Model, which was developed using Random Forest, logistical, Decision Tree, and the support vector classification (SVC) algorithms. Test set prediction is then carried out depending on the accuracy of the test results.

3) Preprocessing:

There might be missing values in the obtained data, which could cause inconsistencies. Preprocessing data is necessary to increase algorithm efficiency and yield better outcomes. In addition to removing the outliers, variable conversion must be completed.

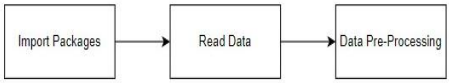

Fig 2. Data Preprocessing

4) Data Visualization:

In the fields of applied statistics and artificial intelligence, data visualization is a critical competency. In fact, the main focus of statistics is on quantitative data descriptions and estimations. An essential set of tools for developing a qualitative understanding is provided by data visualization. This can be useful for discovering trends, faulty data, outliers, and a great deal more while examining and becoming acquainted with a dataset. Plots and graphs that are more emotional and stakeholders than measurements of association or significance can convey and illustrate important relationships with the use of data visualizations, provided the user has some topic expertise. It will be suggested to delve deeper into some of the books indicated at the conclusion, as information visualization and exploration of data are entire topics in and of themselves. When data is presented visually, such with charts and graphs, it may sometimes make sense. In applied statistics as well as applied machine learning, the ability to display data samples and other types

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 3, December 2023

of information rapidly is crucial. It will reveal the many plot types that you should be aware of when using Python to visualize data and show you how you can employ them to enhance your understanding of your own data.

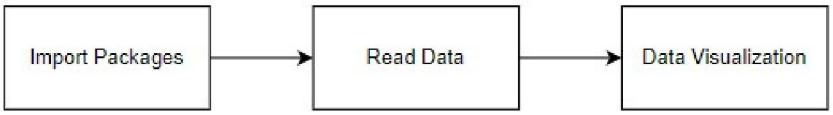

Fig 3.Data Visualization

5) Building the classification model:

The software bug forecast, The following factors contribute to the effectiveness of an extremely precise prediction model: In the categorization problem, it yields superior results.

It performs well when preprocessing a mixture of continuous, categorical, and discrete data as well as outliers and irrelevant variables.

It generates out-of-bag estimate error, which is rather straightforward to tweak and has shown to be impartial in several testing.

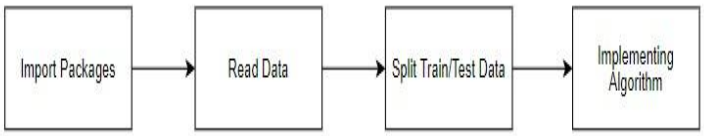

Fig 4. Model Implementation

IV. MODEL IMPLEMENTATION

4.1. Logistic Regression for Bug Detection:

Examine the practical aspects of classifying software defects using Logistic Regression. Recognize how to train models, prepare data, and assess their output.

4.2. Random Forest for Bug Detection:

Examine the potential of the ensemble learning algorithm Random Forest for locating software problems. Discover how to improve model parameters and deal with imbalanced datasets.

4.3. Naive Bayes for Bug Detection:

Examine the possibilities for using the probabilistic classifier Naive Bayes for software bug detection jobs. Recognize the underlying presumptions of the model known as Naive Bayes and how it is used to bug report text categorization.

4.4. k-Nearest Neighbors (KNN) for Bug Detection:

Learn about the method known as KNN and how to use it to find software bugs. Study up on distance measures and how to get the best value for 'k.'

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 3, December 2023

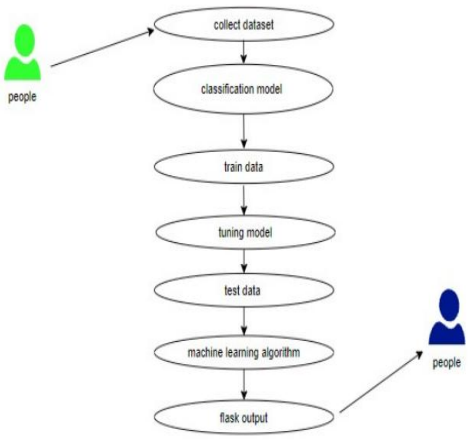

Fig 5. User Access Architecture

V. RESULTS AND DISCUSSIONS

Various machine learning methods yielded encouraging results in assessing the efficacy of our software bug identification models. With an accuracy score of 78%, Logistic Regression was the most accurate method, shortly followed by Naive Bayes. Having a success rate of 79%, Random Forest performed somewhat better, proving its usefulness in software bug classification. Its k-Nearest Neighbors (KNN) algorithm shown its potential for bug discovery even with a respectable 77% accuracy rate. These findings show that our artificial intelligence learning-based methods for detecting bugs are reliable and effective in helping to find software flaws. The little variations in accuracy highlight how crucial it is to choose the best model for a given bug detection task, taking into account elements like data properties and processing power.

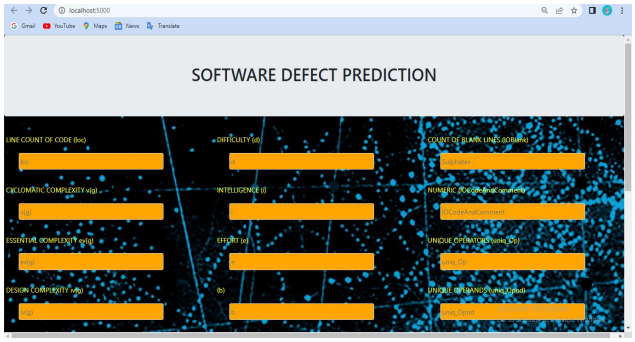

Fig 5.1. Final Deployment on Flask Modal

Fig 5.2. Prediction Results Diagram

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 3, December 2023

Overall, these findings support more research into machine learning for software quality assurance and provide developers with useful tools to improve software dependability and user happiness but might be especially useful for proximity-based bug discovery. These findings highlight how crucial it is to choose the best model depending on the particulars of the given situation. Overall, our research shows how important machine learning is for software quality assurance, providing flexible methods to reduce errors and improve user experience.

VI. CONCLUSION

To sum up, our investigation into machine learning algorithms for software problem identification has shown that they have a great deal of promise for improving software quality. Engineers and QA teams have useful alternatives with these models, which have accuracy rates between 78% and 79%. The particulars and demands of the current bug detection assignment should guide the selection of the model to use. The adaptability of machine learning offers a strong ally in effectively detecting and resolving errors as software systems get more sophisticated. Software development may become more stable and trustworthy by utilizing these models, which will eventually improve stakeholder satisfaction and user experiences. Data cleaning plus processing, blank value analysis, exploratory analysis, and model construction and assessment were the first steps in the analytical process. Higher accuracy scores will reveal which test set has the greatest accuracy. This program can assist in predicting software defects.

REFERENCES

[1]. Arun Balaaji, S., Gowtham, J., Amirtharaj, B., & Bala Abirami, B. IDENTIFING SOFTWARE BUGS OR NOT USING SMLT MODEL.

[2]. Praveen, A. K., Harsita, R., Murali, R. D., & Niveditha, S. (2023). Detecting Fake Job Posting Using ML Classifications and Ensemble Model. Advances in Science and Technology, 124, 362-369.

[3]. Prabha, C. L., & Shivakumar, N. (2020, June). Software defect prediction using machine learning techniques. In 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184) (pp. 728-733). IEEE.

[4]. Esteves, G., Figueiredo, E., Veloso, A., Viggiato, M., & Ziviani, N. (2020). Understanding machine learning software defect predictions. Automated Software Engineering, 27(3-4), 369-392.

[5]. Tadapaneni, P., Nadella, N. C., Divyanjali, M., & Sangeetha, Y. (2022, July). Software Defect Prediction based on Machine Learning and Deep Learning. In 2022 International Conference on Inventive Computation Technologies (ICICT) (pp. 116-122). IEEE.

[6]. Chappelly, T., Cifuentes, C., Krishnan, P., &Gevay, S. (2017, February). Machine learning for finding bugs: An initial report. In 2017 IEEE workshop on machine learning techniques for software quality evaluation (MaLTeSQuE) (pp. 21-26). IEEE.

[7]. Aquil, M. A. I., & Ishak, W. H. W. (2020). Predicting software defects using machine learning techniques. International Journal, 9(4), 6609-6616.

[8]. Matloob, F., Aftab, S., Ahmad, M., Khan, M. A., Fatima, A., Iqbal, M., ... &Elmitwally, N. S. (2021). Software defect prediction using supervised machine learning techniques: A systematic literature review. Intelligent Automation & Soft Computing, 29(2), 403-421.

[9]. Mehmood, I., Shahid, S., Hussain, H., Khan, I., Ahmad, S., Rahman, S., ... & Huda, S. (2023). A Novel Approach to Improve Software Defect Prediction Accuracy Using Machine Learning. IEEE Access.

[10]. Santos, G., Figueiredo, E., Veloso, A., Viggiato, M., & Ziviani, N. (2020, December). Predicting software defects with explainable machine learning. In Proceedings of the XIX Brazilian Symposium on Software Quality (pp. 1-10).

[11]. Hassan, F., Farhan, S., Fahiem, M. A., & Tauseef, H. (2018). A review on machine learning techniques for software defect prediction. Technical Journal, 23(02), 63-71.