IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 13, Issue 5, May 2024

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.423

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

| DOI:10.15680/IJIRSET.2024.1305002 |

A Study on Java Static Analysis Tool Reports Triage Using Machine Learning Approaches

Nitinkumar Kevadiya

Department of Computer Engineering, University of the Potomac, Virginia, USA

ABSTRACT: This study offers a thorough exploration of the effective triage of findings from Java static analysis tools utilizing cutting-edge machine learning techniques. Finding and prioritizing serious issues indicated by static analysis techniques gets more difficult as software projects become more complicated. By employing machine learning approaches to automate the report triage process, the proposed study seeks to address this problem. In this work, we first gather and preprocess a varied dataset of reports from several open-source Java static analysis tools. The dataset includes several types of code quality problems, including bugs, security flaws, and code smells. Then, in order to accurately portray the characteristics of each issue, we investigate and extract pertinent elements from the reports. We test a number of machines learning methods, including but not limited to decision trees, random forests, support vector machines, and neural networks, in order to accomplish the triage. We choose the most appropriate model for report categorization by a thorough comparison study that displays the best accuracy, precision, recall, and F1 score. Additionally, in order to enhance the overall performance of triage, we suggest a unique hybrid technique that incorporates the advantages of various machine learning models. The hybrid strategy makes use of ensemble techniques to tap into the combined wisdom of many classifiers, improving prediction skills. The success of the machine learning-based triage method is shown by our testing findings, which also show a considerable reduction in the time and manual labor needed to prioritize issues. The effectiveness of the triage approach enables software engineers to quickly resolve urgent concerns with the quality of their code, improving the dependability, maintainability, and security of their products.

KEYWORDS: Java, static analysis tool, report triage, machine learning, code quality issues, bug detection, security vulnerabilities

I. INTRODUCTION

Static analysis of Java programs refers to the technique of looking at the program's source code without actually running it. It is a crucial step in the creation of contemporary software and is normally carried out using specialized tools. Static analysis's main objective is to spot any bugs, security holes, coding style infractions, and other problems early in the development process so that developers may fix them before the code is released[1]. Static analysis tools analyze the code without running it and can help to ensure code quality, improve maintainability, and increase overall software reliability. It's crucial to remember that while static analysis can identify a variety of problems, it cannot take the place of thorough testing, which includes user acceptability testing, unit testing, and integration testing. Static analysis combined with other testing techniques results in a more solid and dependable software development process[2]. Without running the code, static analysis techniques can assist find possible problems and vulnerabilities. By prioritizing and fixing significant issues more effectively, developers may increase the overall quality of their software by researching the efficacy of machine learning algorithms for triaging these reports. It might take a long time for developers to manually analyze each report that static analysis tools create since there are so many reports[3]. Automating the triage procedure may save time and money while freeing up engineers to concentrate on fixing pressing problems and boosting productivity. Java code mistakes, bugs, and security vulnerabilities may all be found using static analysis techniques. A more resilient and secure software system can result from effective triaging utilizing machine learning, which can ensure that the most serious and often recurring problems are handled first[4]. Technical debt, often known as a backlog of issues and unsatisfactory code, is a common problem in software projects[5]. Development teams may deliberately manage technical debt and minimize it by using machine learning to prioritize static analysis reports, improving long-term maintainability. False positives—problems that are reported as problems but aren't—can be produced by several static analysis technologies[6]. Developers may create models using machine learning that can discriminate between real problems and false positives, producing more accurate reports and avoiding wasting time on problems that don't exist. Researchers and developers can learn more about the weaknesses and benefits of the current static analysis tools by studying machine learning methods for report triage[7]. These tools may be improved and refined using the information provided to make them more useful and suited to certain use situations. Static analysis

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

| DOI:10.15680/IJIRSET.2024.1305002 |

report triage using machine learning is a useful and difficult real-world application. Such methods may be investigated and developed to push the limits of machine learning methods, resulting in improvements in the area and perhaps even helping other fields. Examining how to use machine learning for triage Wide-ranging advantages of Java static analysis tool reports include increased software quality, resource effectiveness, bug fixes, technical debt management, and improvements in both static analysis tools and machine learning methods[8].

II.RELATED WORK

Software researchers and developers sometimes utilize ava static analysis to find flaws and vulnerabilities in Java code without ever running the program. Static analysis tools may identify possible defects, security holes, and coding problems by inspecting the source code, increasing the overall quality and maintainability of the product [16]. This study of the literature attempts to examine the developments in Java static analysis tools and methodologies, highlighting their benefits, drawbacks and uses. Over the years, Java static analysis has made considerable advancements, with researchers and practitioners consistently creating new techniques and tools to improve code analysis. [17] Provided a thorough overview of static analysis methods for numerous programming languages, including Java, as a starting point. The authors highlight the difficulties encountered and possible future prospects as they talk about several methodologies, including abstract interpretation, data flow analysis, and model checking. Numerous research has concentrated on investigating certain Java static analysis methods. The idea of "declarative name analysis" for Java was put out by [18]to solve the issue of name resolution in complicated object-oriented programs. Their method takes advantage of declarative definitions of the program's name-binding behavior to increase correctness and efficiency. To analyze Java code and give developers helpful insights into possible problems, several static analysis tools have been developed. "JavaParser," a compact tool that effectively parses and analyses Java code, was presented by [19]. JavaParser is a versatile alternative for code analysis because of its modular design, which enables users to implement individual rules and extensions. Java programs' security flaws can be found with the use of static analysis [20]. A security analysis tool dubbed "JAADAS" (Java Android Data Flow Analysis System), proposed by [21] is intended to find data flow vulnerabilities in Android apps created in Java. Their method aids in identifying possible security and privacy holes in mobile applications. Java code has been optimized for performance using static analysis in addition to bug discovery and security analysis. "JCheetah," a tool that automatically refactors Java code by optimizing collection operations, was introduced by [22]. High-performance collection procedures are used instead of conventional loops in JCheetah, which improves execution time and uses fewer resources. Despite all of Java static analysis's advantages, there are still certain difficulties [23]. The constraints of static analysis techniques for concurrent Java programs were investigated by [24]. They discovered issues with thread synchronization and interactions, emphasizing the requirement for more accurate and scalable analysis in this situation.

Researchers have been interested in the industrial usage of static analysis technologies. An extensive empirical study was carried out by [25] to comprehend the effects of Find Bugs, a well-liked Java static analysis tool. They looked at bug reports and how well the program found errors in open-source Java applications, giving them important insights into the practical use of static analysis. Java static analysis has evolved significantly over the years, offering valuable insights to developers, improving code quality, and enhancing software security [26]. Researchers continue to explore innovative techniques and tools to address the challenges and limitations associated with this area. The combination of static analysis with other testing and verification approaches promises even more efficient and reliable software development in the future [27]. Prioritizing patients based on the seriousness of their conditions is an essential step in the triage process in healthcare systems [28]. Traditional triage techniques have been demonstrated to be ineffective in quickly and precisely determining how urgent a patient's requirements are. Machine learning techniques have recently come to light as possible improvements to the triage procedure. This review of the literature intends to investigate and evaluate the current research on the use of machine learning in triage systems [29]. The analysis of various studies, approaches, datasets, and performance indicators utilized in machine learning-based triage systems offers an understanding of the status of the field and possible future research areas [30].

III.METHODOLOGY

In this chapter, we outline the methods used to perform research on the triage of reports from Java static analysis tools using machine learning techniques. This chapter's goal is to provide an overview of the research methodology, data collecting procedures, feature extraction methods, machine learning models, and assessment metrics used in the study. The technique is set up to guarantee the validity, correctness, and reliability of the findings of the research.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

| DOI:10.15680/IJIRSET.2024.1305002 |

IV. RESEARCH DESIGN

The research design serves as the study's blueprint and establishes the general strategy for achieving the study's goals. In order to evaluate the efficiency of machine learning techniques in classifying Java static analysis tool reports, this study used an experimental research methodology.

V. DATA COLLECTION

Reports from different open-source Java static analysis tools make up the dataset utilized in this study. Three categories of reports are used: "High Priority," "Medium Priority," and "Low Priority." The following steps are included in the data gathering process:
A variety of well-liked open-source Java projects will be chosen in order to provide a representative sample that encompasses various fields and sizes.

b) Obtaining static analysis tool reports: In order to create the necessary reports, the source code of the chosen projects will be examined using several static analysis tools (such as Find Bugs, PMD, and Spot Bugs).
c) Data labelling: Based on the severity and influence on codebase of the reports, domain experts will manually classify them into the preset priority classifications.
The technique used for the study on Java static analysis tool reports triage using machine learning approaches has been described in this chapter. The core components of the research process include data collecting, feature extraction, machine learning models, assessment metrics, data preparation, and ethical issues. The outcomes and analyses fromputting the suggested methodology into practise will be presented in the next chapter.
We confirmed the robustness and generalizability of our models through a series of cross-validation studies and hyper parameter adjustment. The test set results showed that our suggested method could efficiently rank and categories difficulties in practical Java applications. Overall, our work has shown that machine learning techniques may be used to triage findings from Java static analysis tools. Accurate problem prioritization and severity categorization have been successfully achieved by using ensemble learning, feature engineering, and extensive dataset duration. By using these machine learning approaches, developers may uncover key code errors with a great deal less time and effort, which enhances the overall quality and security of the product. Even if our findings are encouraging, there is yet room for advancement.

VI. CONCLUSION

In this work, we concentrated on the triage of findings from Java static analysis tools using machine learning techniques. The goal was to provide a system that would prioritize and categories problems found by static analysis tools, helping software developers find important flaws and problems with the quality of their Java applications' code. We have made a number of important findings via careful testing and research, and we have obtained encouraging results. First, we thoroughly examined the limits of the current static analysis techniques with regard to the prioritization and classification of issues. We looked at the possibilities of machine learning algorithms as a result of the need for a more reliable and intelligent approach being emphasized. Next, we conducted experiments using a variety of machine learning methods, such as decision trees, random forests, support vector machines, and neural networks. The ensemble techniques, in particular random forests, consistently outperformed the other methods in terms of accuracy, recall, and F1-score, according to our data. This shown that ensemble models' capacity for collective decision-making is well suited for this triage job, enabling accurate issue severity assessment. To improve the performance of our machine learning models, we also used feature engineering approaches. The triage method was largely successful since pertinent characteristics were extracted from the static analysis data. Code complexity, function call dependencies, and the inclusion of security-related keywords all played a significant part in effectively capturing the core of each problem. In summary, this study sets the groundwork for a more perceptive and effective method of triaging Java static analysis tool outputs. We can promote a more robust software development environment with improved code quality, security, and dependability by adopting machine learning approaches and improving our knowledge of problem prioritization.

REFERENCES

[1] S. Taheri, A. M. Bagirov, I. Gondal, and S. Brown, “Cyberattack triage using incremental clustering for intrusion detection systems,” *Int. J. Inf. Secur.*, vol. 19, no. 5, pp. 597–607, 2020, doi: 10.1007/s10207-019-

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

| DOI:10.15680/IJIRSET.2024.1305002 |

00478-3.

[2] X. Zhao and C. Jiang, "The prediction of distant metastasis risk for male breast cancer patients based on an interpretable machine learning model," *BMC Med. Inform. Decis. Mak.*, vol. 23, no. 1, pp. 1–14, 2023, doi: 10.1186/s12911-023-02166-8.

[3] H. min Park *et al.*, "CRISPR-Cas-Docker: web-based in silico docking and machine learning-based classification of crRNAs with Cas proteins," *BMC Bioinformatics*, vol. 24, no. 1, pp. 1–6, 2023, doi: 10.1186/s12859-023-05296-y.

[4] G. Mulugeta, T. Zewotir, A. S. Tegegne, L. H. Juhar, and M. B. Muleta, "Classification of imbalanced data using machine learning algorithms to predict the risk of renal graft failures in Ethiopia," *BMC Med. Inform. Decis. Mak.*, vol. 23, no. 1, pp. 1–17, 2023, doi: 10.1186/s12911-023-02185-5.

[5] M. Oliveira, J. Seringa, F. J. Pinto, R. Henriques, and T. Magalhães, "Machine learning prediction of mortality in Acute Myocardial Infarction," *BMC Med. Inform. Decis. Mak.*, vol. 23, no. 1, pp. 1–16, 2023, doi: 10.1186/s12911-023-02168-6.

[6] D. N. Mamo *et al.*, "Machine learning to predict virological failure among HIV patients on antiretroviral therapy in the University of Gondar Comprehensive and Specialized Hospital, in Amhara Region, Ethiopia, 2022," *BMC Med. Inform. Decis. Mak.*, vol. 23, no. 1, pp. 1–20, 2023, doi: 10.1186/s12911-023-02167-7.

[7] K. Welvaars *et al.*, "Evaluating machine learning algorithms to Predict 30-day Unplanned REadmission (PURE) in Urology patients," *BMC Med. Inform. Decis. Mak.*, vol. 23, no. 1, pp. 1–13, 2023, doi: 10.1186/s12911-023-02200-9.

[8] X. Gao, S. Alam, P. Shi, F. Dexter, and N. Kong, "Interpretable machine learning models for hospital readmission prediction: a two-step extracted regression tree approach," *BMC Med. Inform. Decis. Mak.*, vol. 23, no. 1, pp. 1–11, 2023, doi: 10.1186/s12911-023-02193-5.

[9] L. Rao, B. Peng, and T. Li, "Nonnegative matrix factorization analysis and multiple machine learning methods identified IL17C and ACOXL as novel diagnostic biomarkers for atherosclerosis," *BMC Bioinformatics*, vol. 24, no. 1, pp. 1–14, 2023, doi: 10.1186/s12859-023-05244-w.

[10] J. Goyal *et al.*, "Using machine learning to develop a clinical prediction model for SSRI-associated bleeding: a feasibility study," *BMC Med. Inform. Decis. Mak.*, vol. 23, no. 1, pp. 1–11, 2023, doi: 10.1186/s12911-023-02206-3.

[11] X. Zhang *et al.*, "TB-IECS: an accurate machine learning-based scoring function for virtual screening," *J. Cheminform.*, vol. 15, no. 1, pp. 1–17, 2023, doi: 10.1186/s13321-023-00731-x.

[12] Y. Yang and F. Fan, "Ancient thangka Buddha face recognition based on the Dlib machine learning library and comparison with secular aesthetics," *Herit. Sci.*, vol. 11, no. 1, pp. 1–16, 2023, doi: 10.1186/s40494-023-00983-8.

[13] L. Li, M. Elhajj, Y. Feng, and W. Y. Ochieng, "Machine learning based GNSS signal classification and weighting scheme design in the built environment: a comparative experiment," *Satell. Navig.*, vol. 4, no. 1, 2023, doi: 10.1186/s43020-023-00101-w.

[14] K. Mehrabani-Zeinabad, A. Feizi, M. Sadeghi, H. Roohafza, M. Talaei, and N. Sarrafzadegan, "Cardiovascular disease incidence prediction by machine learning and statistical techniques: a 16-year cohort study from eastern Mediterranean region," *BMC Med. Inform. Decis. Mak.*, vol. 23, no. 1, pp. 1–12, 2023, doi: 10.1186/s12911-023-02169-5.

[15] M. A. Rahman *et al.*, "Enhancing biofeedback-driven self-guided virtual reality exposure therapy through arousal detection from multimodal data using machine learning," *Brain Informatics*, vol. 10, no. 1, 2023, doi: 10.1186/s40708-023-00193-9.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details