IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

SMART SOFTWARE DEFECT PREDICTION USING SOFT COMPUTING

[1]Mrs. P. Bhavani, [2]S. Prabhakaran, [3]N. Chandraj, [4]K. Jwitesh, [5]M. Mohamed Safic

[1]Research Scholar, [2]Student, [3]Student, [4]Student, [5]Student

Department of Computer Science and Engineering

Sri Manakula Vinayagar Engineering College, Puducherry, India

Abstract : The soft computing techniques aim to find good practical solutions to complex real-world problems where 'traditional' (more exact) computational techniques cannot be applied. Overall, with this approach, the problems that are difficult to be precisely solved can be solved at a lower cost. Defects (also bug or error) are what make the program behave in unexpected ways in software. Although a lot of work has been done to improve detection and correction of these defects, there exist no exact solutions and still exist many challenges. And here one of the techniques that is widely used in this field is multicriteria optimization approach (MCOA). The results for using this method have been better than older methods in terms of dealing with software defects.

IndexTerms **- Multicriteria optimization approach (MCOA), Deep Reinforcement Learning (DRL), Software defects, Swarm computing.**

INTRODUCTION

Soft computing in computers is concerned with soft approximations for hard real-world problems. It contains plenty of approaches and strategies for addressing approximation, imprecision, and uncertainty that can handle things where standard hard computing methods necessarily should be hard, hard, binary and deterministic approaches to things. Soft computing techniques are appropriate for real world applications where completely correct information seldom exist (that is, they are designed to cope with noisy and ambiguous data).

A lot of soft computing techniques such as genetic algorithm and neural networks are able to learn from new data and get better with experience; they are strongly robust and flexible: they can adapt to unexpected changes of the data.

In some ways, a issue can often be fixed when we don't have enough precise info in hand. While soft computing techniques can still generate approximations to answers, even when all the information is missing. Soft computing techniques aim at replicating human-like thinking, and at that human knowing is often more efficient in dealing with tough matters.

Swarm computing addresses solutions to these problems offering a unique solution, inspired from the way something behaves in nature, like a group of birds or ants foraging individually. Artificial intelligence is a branch of which specializes trying to find solutions to difficult problems utilizing decentralized, self-organizing systems; in other words, swarm intelligence.

The field of software engineering clearly is not fully devoid of defects in software and is a very important concern. Most of the time, these defects — from minor bugs to major system failures — result in increased development costs and stretched project timelines, along with undermined user satisfaction.

Reinforcement learning (RL) models, particularly the Deep Q-Networks(DQN and Actor-Critic approach of all kinds), will learn intelligently from prior project data what software modules are more likely to be faulty by time adaptively.

How It Works: The DRL algorithm can be configured to "reward" predictions that are highly likely of identifying defects earlier. As the model sees more samples over time, it learns about features and code patterns in association to highrisk modules so that it can adjust its classification accordingly.

Benefits: The more feedback DRL-based techniques received from test and deployment, the better defect detection would become. It is very helpful in a dynamic agile environment with a code many changes.

Based on traditional defect prevention techniques such as code reviews, testing and static analysis, these issues have been mitigated. But such methods suffer from their inability to cope with the complexity and dynamics typical to modern software environments.

1.1 Need for Software Defect Prevention

The software defects from minor glitches to serious failure has a huge impact on the software quality, project timeline and the overall user satisfaction. Reliability, maintainability and functionality of software systems require the need of effective strategies for software defect prevention. Thus, several key factors emphasize on importance of proactive defect prevention measures

Cost and Time Constraints: The cost and time implication of software defects is prohibitive throughout development lifecycle. The resources involved in the identification and resolution of the defects increase exponentially in proportion to the size of the defects as they propagate. These costs (and project delays) can be minimized with early detection and prevention. Accelerated development cycles are a constant, often fulfilling time-to market pressures, leaving little time for complete testing and debugging. Prevention of defect helps organizations to reduce the risk and software products can be delivered on time by achieving high quality.

Impact on User Experience: Not only can software defects create frustration, loss of trust, and reduced productivity for users, they also represent a major drain on developers' schedules. Even small glitches can damage the reputation and destroy customer loyalty of the software products. Maintaining a good user experience and customer satisfaction requires preventing defects during the projects before they appear in the production environment.

Reliability and Safety Concerns: In safety critical domains such as healthcare, aerospace and automotive defects in software can lead to financial losses, injury or loss of life. Risk and reliability, safety, and compliance of software systems in these critical domains require precisely such rigorous defect prevention measures. Complexities in these need to be dealt with by proactive defect prevention strategies for improving software reliability.

Continuous Deployment: For the continuous deployment, we need to apply a shift left approach to defect prevent. Rapids iteration and deployment cycle requires that defects be identified and resolved early in the development lifecycle. By integrating defect prevention measures into automated build pipelines and continuous integration environments you can take advantage of rapid feedback loops and have a culture of quality in software development process.

1.2 Swarm Computing and optimization Techniques

Swarm computing, known as swarm intelligence, is a sub discipline of both computational and artificial intelligence based on social behavior of like fish, birds, ants and bees. Collective behavior of these species arises from the conglomerate of basic individual interactions leading to highly sophisticated and efficient problem solving techniques.

In contrast to current computer models in which central control is required, swarm computing relies upon a distributed system of autonomous agents interacting locally with each other. However, basic local interactions in agents can lead to complex global nature, escaping from problems that are difficult for individual agents. For applications requiring flexibility, the resilience of swarm systems is due to the fact that they can adapt to dynamic settings and are generally resilient to individual failures.

Swarm intelligence approaches allow more accurate and efficient data analysis on feature selection, classification and clustering.

Diversity of swarm computing techniques is based on different natural systems and biological behaviors. The use of these methods involves the employment of complicated problems by relying on decentralized, selforganized cooperations systems.

The use of mathematical and computer approaches, called optimization, is used to find the best potential solutions to a given problem within a restricted specified area. In many areas of machine learning, operations research, engineering, and economics, these methods are important.

Various kinds of optimization challenges are classified. Formal optimization is usually required for certain straightforward problems, but some obvious problems do not need it. The desired outcome is to have optimal outcomes, in the overwhelming majority of cases, a mathematical solution is necessary. The majority of issues require optimization. The goal of problem solving is to minimize risk and minimize expenses. It may also be multi objective and may involve many judgments.

An optimization problem can be solved with three key components: an aim and constraints and variables. It is to find the best values for every variable, which has a bunch of values. It is to get to the desired result.to various kinds. Certain straightforward problems, including those with obvious solutions do not require formal optimization. The aim is to attain optimal outcomes, and in the majority of circumstances, a mathematical solution is required. Optimization is necessary for the majority of issues. Minimizing risk and cutting down on expenses are the goals of problem solving. Additionally, it could entail many judgments and be multi-objective.

An optimization problem can be solved with three key components: constraints, variables, and an aim. The goal is to determine the best value for every variable, which may have a variety of values. The purpose is to reach the desired result Multicriteria Optimization Approach

A multicriteria optimization approach optimizes simultaneously two (or more) conflicting objectives. It is important to use this approach when decision makers have to make trade-offs between different objectives, as happens in almost all real-world problems.

2.Related Works

The purpose of this literature review is to present a complete understanding and critical analysis of the current state of the art defect prediction methodologies and techniques. Then, it explores a variety of different approaches, which include statistical models, machine learning algorithms, ensemble techniques, and hybrid optimization methods, and highlights the pros and cons of the approaches, and how they impact software engineering practice. Through a review of the existing literature, the review aims to identify the key trends, challenges and developments in software defect prediction and explores the most promising venues for future research and development.ls, machine learning algorithms, ensemble techniques, and hybrid optimization methods, with a focus on their applications, strengths, limitations, and implications for software engineering practice.

Through an extensive survey of the existing literature, the review seeks to identify the key trends, challenges, and advancements in software defect prediction, highlighting the most promising avenues for future research and development. The review attempts to synthesize from diverse studies and experiments to gain insights into the complexities and nuances that work in software defect prediction, so that our progress in software engineering will be shortened.

2.1 Literature Survey Table

Sl. No.	Title	Authors	Description	Advantage	Disadvantage
1	Software Defect Prediction Analysis Using Machine Learning Techniques	Khalid, A. Badshah, G. Ayub, N. Shiraz, and M. Ghouse	This study uses PSO and K-means clustering with classifiers (SVM, Naive Bayes, Random Forest) to optimize ensemble models for software defect prediction. SVM models achieved the highest accuracy on the CM1 dataset.	This study leverages machine learning classifiers like SVM, Naive Bayes, and Random Forest with ensemble techniques, further optimized using PSO and K-means clustering, to improve prediction accuracy. SVM and its optimized versions outperform others on the CM1 dataset.	Its limitation lies in the poor generalization of results, as the analysis is confined to a single dataset.
2	A New Binary Chaos-Based Metaheuristic Algorithm for Software Defect Prediction	Bahman Arasteh	This paper presents a Binary Chaos-based Olympiad Optimization Algorithm (BCOOA) for feature selection in software defect	The paper introduces the Binary Chaos-based Olympiad Optimization Algorithm (BCOOA) for effective feature	Restricted by the use of only one dataset, which may affect the reliability and applicability of the results.

			prediction. The selected features help build an efficient classification model to identify faulty software modules.	selection and efficient fault classification in software modules. It enhances prediction performance .	
3	Unified Integration of Many-Objective Optimization Algorithm Based on Temporary Offspring for Software Defects Prediction	X. Cai, S. Geng, D. Wu et al.	Introduces a multi-objective optimization algorithm for software defect prediction, aiming to address class imbalance and improve SVM parameter selection. objectives can	Using a many-objective optimization algorithm (UIMaOTO), this paper effectively addresses issues like class imbalance and parameter selection for SVM while balancing multiple evaluation metrics.	Its comprehensive approach, managing and optimizing multiple objectives can be challenging, and real-world performance with imbalanced data remains to be validate
4	Software Fault Prediction using Wrapper-based Ant Colony Optimization Algorithm for Feature Selection	Sagnik Mondal, Adarsh Kumar Sahu, Hrishikesh Kumar, Radha Mohan Pattanayak, Mahendra Kumar Gourisaria, Himansu Das	This paper utilizes Ant Colony Optimization (ACO) for feature selection in software fault prediction, compared with classifiers like KNN, Naive Bayes, and Decision Tree.	This research proposes an ACO-based feature selection method, significantly improving software fault prediction by eliminating irrelevant features and comparing results across 12 datasets using multiple classifiers.	The approach is computationally intensive, and its performance varies across datasets, limiting its generalization.
5	Enhancing Software Fault Prediction through Feature Selection with Spider Wasp Optimization Algorithm	Himansu Das, Swarnava Das, Mahendra Kumar Gourisaria, Surbhi Bhatia Khan, Ahlam Almusharraf, Abdullah I. Alharbi	Proposes Spider Wasp Optimization (FSSWO) for feature selection in software fault prediction. FSSWO is compared with other optimization methods, showing superior results on 11 datasets.	The paper presents Spider Wasp Optimization (SWO) for selecting optimal features, demonstrating superior results across benchmark datasets and outperforming traditional methods in accuracy and cost-efficiency.	It lacks details on computational complexity and does not address the algorithm's scalability to large real-world datasets.
6	Deep Reinforcement Learning in Automated	Sreenivasappa B V, Anoop A, Preeti Naval	Uses Deep Q-Network (DQN) and other DRL algorithms to automate network vulnerability	By using DRL algorithms such as DQN, PPO, and DDQN, this study	The downside is the absence of analysis on computational overhead and the

	Network Vulnerability Detection		detection. The system outperforms traditional methods in accuracy, precision, and recall with low execution times..	achieves high accuracy, low false rates, fast execution, and strong scalability in automated network vulnerability detection.	limited scope of DRL techniques covered.
7	Study on the Use of Defect Metrics in the Software Development Process: Flaws and Vulnerabilities	Zaira Hernandez Martinez, Patricia Martínez Moreno,	Reviews defect metrics in software development, their benefits, limitations, and role in improving software quality.	This paper highlights the critical role of defect metrics in identifying and preventing vulnerabilities during early software development.	It falls short by not offering practical solutions or empirical data to support its findings, limiting its applicability in real-world projects.
8	Software Defect Prediction using ANN Algorithm	B. Maruthi Shankar, S.A. Sivakumar, Dharmesh Dhabliya, P. Abinaya Sundari, M. Asmitha,	Discusses using Artificial Neural Networks (ANN) and machine learning for predicting software defects, enhancing software quality by reducing manual detection time.	The research explores the use of ANN and machine learning for early and efficient software defect prediction, aiming to improve quality and reduce manual testing time.	It lacks comparative analysis with other models and does not address the approach's scalability on large datasets.
9	A Novel Approach to Improve Software Defect Prediction Accuracy Using Machine Learning	Iqra Mehmood, Sidra Shahid, Hameed Hussain, Inayat Khan, Shafiq Ahmad,	Introduces a feature selection method integrated with machine learning classifiers to enhance defect prediction accuracy on NASA datasets.	Feature selection-integrated machine learning approach, this study significantly improves defect prediction accuracy on NASA datasets using various classifiers via WEKA.	It does not assess the computational cost or address possible bias and limitations of the datasets used.
10	Test Case Priority Ranking Method Based on Greedy Algorithm	Bo Zhang, Yu Li, Yichen Wang	Proposes a method for prioritizing test cases using a combination of a greedy algorithm and a mountain climbing algorithm, focusing on safety-critical modules.	This work combines greedy and mountain climbing algorithms to prioritize test cases based on safety, resulting in improved efficiency and faster testing.	The study does not thoroughly evaluate the method's scalability for complex projects or its effectiveness across different software types.

3.Research Directions

Progress in the field of software defect prediction demands efforts to determine promising directions for future research and practice. With this in mind, this section outlines possible avenues for future research in defect prediction which addresses emergent challenges and pushes forward the state of the art.• Soft computing techniques in Automatic Generation of testcases• Test case selection, test case reduction priority driven test cases. Optimization of testcases using Bioinspired Algorithms. evolve, there is a

need to identify promising avenues for future research and development. Building upon the insights gained from existing literature and current methodologies, this section outlines potential research directions aimed at addressing emerging challenges and advancing the state-of-the-art in defect prediction.

- Automated generation of testcases using soft computing methodologies.
- Automatic selecting of test cases, prioritizing test cases and reducing the number of cases.
- Test case optimization using bio-inspired algorithm.

Research in these direction will aid scholars and practitioners in contributing to the creation of better, more efficient, and more adaptable defect prediction models to improve software quality and reliability. In order to provide impact for the field of software engineering in the area of software engineering, these sections outline key areas for which further investigation and innovation are needed.

4. Architecture Diagram

The architecture of the proposed system is built on a robust, layered structure that facilitates the seamless integration of various components. The first layer is the Continuous Integration/Continuous Deployment (CI/CD) Pipeline, which automates the process of building, testing, and deploying code. Each commit in the version control system automatically triggers the pipeline, initiating unit tests, integration tests, and defect checks. These automated processes ensure that every new line of code is rigorously tested before it enters the development or production environment.

The second key component is the Swarm Intelligence Defect Prediction Module. This module applies Swarm Intelligence algorithms such as Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) to enhance defect prediction accuracy. The system uses these techniques to optimize feature selection for defect prediction, taking into account factors such as historical defect data, the current code structure, and build metrics. By using Swarm Intelligence, the system is capable of adapting to new data and refining its predictions based on ongoing development. These predictions are then used to prioritize parts of the code that require more rigorous testing or potential refactoring.

Another critical component is the Test Case Optimization Module, which automatically generates and selects the most relevant test cases based on the predicted defects. Using bio-inspired algorithms such as Genetic Algorithms, this module ensures that testing is focused on high-risk areas, preventing unnecessary redundancy and resource wastage. The Real-Time Feedback & Defect Prevention layer provides continuous feedback to developers, warning them about potential defects before deployment and offering suggestions for fixes, thus preventing issues from reaching the production stage.

The system also incorporates a Feedback Loop, where every detected defect feeds back into the system to refine and improve its predictions. This continuous learning process ensures that the model becomes more accurate over time and adapts to new patterns as the software evolves.

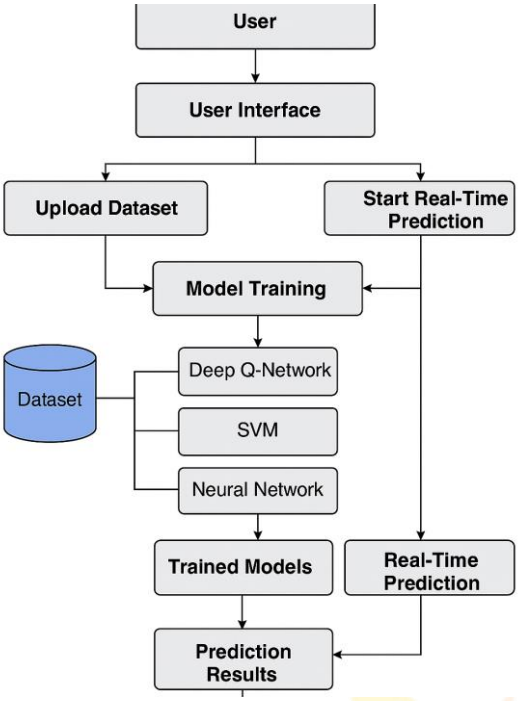

Fig. 4.1 Architectural Design

5.Conclusion

In summary, we are able to validate defect prediction models in real in practice settings to learn more about their effectiveness, identify potential challenges or limitations, and point towards the development of more robust and useful models that will be readily adopted by software development teams.

The research directions presented here provide fresh opportunities for the field of software defect prediction, new solutions to emerging challenges, and development of more effective and practical defect prediction models. Researching along these directions can lead to better software quality, reliability and development process in different application domains.

6.References

[1] Khalid, A., Badshah, G., Ayub, N., Shiraz, M., & Ghouse, M. "Software Defect Prediction Analysis Using Machine Learning Techniques." *Sustainability*, vol. 15, no. 6, 2023. Elsevier, 2023.

[2] Arasteh, B., Arasteh, K., Ghaffari, A., & Ghanbarzadeh, R. "A New Binary Chaos-Based Metaheuristic Algorithm for Software Defect Prediction." *Cluster Computing*, vol. 27, pp. 10093–10123, 2024. Elsevier, 2024.

[3] Cai, X., Geng, S., Wu, D., et al. "Unified Integration of Many-Objective Optimization Algorithm Based on Temporary Offspring for Software Defects Prediction." *Journal Name*, vol. 2020, 2020. [Online]. Elsevier, 2020.

[4] Mondal, S., Sahu, A. K., Kumar, H., et al. "Software Fault Prediction using Wrapper based Ant Colony Optimization Algorithm for Feature Selection." *Journal Name*, vol. 2023, 2023. Elsevier, 2023.

[5] Das, H., Das, S., Gourisaria, M. K., et al. "Enhancing Software Fault Prediction through Feature Selection with Spider Wasp Optimization Algorithm." *Journal Name*, vol. 2024, 2024. Elsevier, 2024.

[6] Sreenivasappa, B. V., Anoop, A., & Naval, P. "Deep Reinforcement Learning in Automated Network Vulnerability Detection." *Journal Name*, vol. 2023, 2023. Elsevier, 2023.

[7] Hernandez Martinez, Z., Martínez Moreno, P., Vergara Camacho, J. A., & Contreras Vega, G. "Study on the Use of Defect Metrics in the Software Development Process: Flaws and Vulnerabilities." *Journal Name*, vol. 2023, 2023. [Online]. Available: [URL]. Accessed: Apr. 3, 2025. Elsevier, 2023.

[8] Shankar, B. M., Sivakumar, S. A., Dhabliya, D., et al. "Software Defect Prediction using ANN Algorithm." *Journal Name*, vol. 2023, 2023. Elsevier, 2023.

[9] Mehmood, I., Shahid, S., Hussain, H., et al. "A Novel Approach to Improve Software Defect Prediction Accuracy Using Machine Learning." *Journal Name*, vol. 2023, 2023. Elsevier, 2023.

[10] Zhang, B., Li, Y., & Wang, Y. "Test Case Priority Ranking Method Based on Greedy Algorithm." *Journal Name*, vol. 2023, 2023. Elsevier, 2023.