International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 12 Issue: 03 | Mar 2025 www.irjet.net p-ISSN: 2395-0072

# Software Bug Prediction: Ensuring Trustworthiness in Software Development: A Review

**Rishit Kantaria**

*Assistant Professor, Department of Information Technology, Dr Subhash University, Gujarat, India*

---

**Abstract –** *Software Bug Prediction is an important research domain which is much useful for the Software Developers and IT Industries. Software Bugs increases the cost of whole Software Development Life Cycle (SDLC).*

*This review paper consist of comprehensive analysis of recent advancements in Software Bug Prediction in field of Machine Learning , Deep Learning and Applying Models on Live Workflow. As data pre-processing plays a vital role in any of the Machine Learning Models , this paper also discuss the role of various pre-processing techniques like Dimensionality Reduction , Feature Extraction and Feature Prediction to improve the accuracy of a Model.*

***Key Words***: Bug Prediction, Machine Learning, Deep Learning, Dimensionality Reduction, Feature Extraction, Evaluation Metrics, Classification.

## 1. INTRODUCTION

Software quality is a critical concern in modern software development, as defects and bugs can lead to significant financial losses, security vulnerabilities, and user dissatisfaction. To mitigate these risks, software bug prediction has emerged as a key research area, enabling early identification of potential defects before deployment. By leveraging historical code metrics, machine learning, deep learning, and statistical techniques, researchers have developed various predictive models to enhance software quality and reliability.

Traditional bug prediction approaches rely on supervised learning algorithms such as decision trees, support vector machines, and artificial neural networks. More recent advancements include ensemble learning, hyper parameter optimization, and deep learning techniques that improve accuracy and scalability. Additionally, just-in-time bug prediction models and explainable AI methods are gaining traction, offering real-time defect identification and interpretability.

Despite significant progress, software bug prediction faces challenges such as imbalanced datasets, feature selection complexity, and model generalization across different projects. This review paper provides a comprehensive analysis of existing bug prediction methodologies, evaluating their strengths, limitations, and emerging trends. Furthermore, it highlights the impact of programming languages, feature engineering, and dimensionality reduction techniques on bug prediction accuracy. Finally, we discuss future research directions, emphasizing the need for hybrid models, explainable AI, and real-world applications of predictive analytics in software engineering.

The rest of this review paper is organized as follows: It provides an overview of traditional and modern software bug prediction techniques, including machine learning and deep learning approaches. Focuses and discusses key challenges in bug prediction, such as data imbalance, feature selection, and model interpretability. Additionally, it presents a comparative analysis of existing methodologies, evaluating their effectiveness and limitations. It also explores emerging trends, including explainable AI and just-in-time defect prediction. Finally, it also summarizes the findings and suggests future research directions for improving software bug prediction models."

## 2. OVERVIEW TRADITIONAL BUG PREDICTION TECHNIQUES

Traditional Bug Prediction Techniques primarily focuses on Software Metrics such as Lines of Code, Coupling, Cohesion, and Cyclomatic Complexity. Various Machine Learning Algorithms are also used for Bug Prediction like Logistic Regression, SVM, and Decision Trees. The underlying limitations in above Traditional Bug Prediction are the High False Positive and High False Negative Rates. The other limitation is the implementation of inappropriate Data Pre Processing Techniques.

**Table -1:** Comparison of Traditional and Modern Software Bug Prediction Techniques.

| Feature | Traditional Bug Prediction Models | Modern Software Bug Prediction Models |
|---|---|---|
| Approach | Software Metrics , Code Metrics. | Deep Learning and Ensemble Learning Techniques. |
| Data Used | Historical Data | Real time Data |
| Algorithms | Logistic Regression, Decision Trees, SVM, Naïve | Neural Networks, Random Forest, XGBoost, |

| | | |
|---|---|---|
| | Bayes | Transformer-based models |
| Handling Imbalanced Data | Struggles with class imbalance | Uses resampling techniques, synthetic data generation |
| Flexibility | Requires manual tuning for different projects | Adaptive models that self-improve with more data |

## 3. MODERN BUG PREDICTION TECHNIQUES

Cross-Validation Parameter Selection (CVParameter Selection), employing a predefined set of hyper parameter values for multiple learning algorithms. CVParameter Selection involves systematically trying out different hyper parameter values for multiple learning algorithms through cross-validation to explore the model's performance under different configurations, ultimately selecting the hyper parameter values that result in the best model performance. Each algorithm has distinct hyper parameters chosen to maximize accuracy and performance on the software bug prediction dataset. [1].

| Algorithm | Accuracy | ROC | Kappa | Sensitivity | Specificity | F-measure |
|---|---|---|---|---|---|---|
| Vote | 0.7189 | 0.763 | 0.1898 | 0.828 | 0.373 | 0.702 |
| Bagging | 0.7178 | 0.749 | 0.2325 | 0.836 | 0.411 | 0.782 |
| Random Forest | 0.7172 | 0.716 | 0.2366 | 0.839 | 0.425 | 0.715 |
| AdaBoostM1 | 0.7169 | 0.781 | 0.2189 | 0.809 | 0.416 | 0.766 |
| Logistics Regression | 0.5135 | 0.613 | 0.1395 | 0.722 | 0.406 | 0.759 |
| SVM | 0.6107 | 0.522 | 0.0664 | 0.713 | 0.359 | 0.639 |
| Neural Networks | 0.5094 | 0.519 | 0.0683 | 0.713 | 0.318 | 0.640 |
| Naive Bayes | 0.6042 | 0.509 | 0.1932 | 0.632 | 0.314 | 0.670 |
| Decision Tree | 0.5954 | 0.656 | 0.1901 | 0.733 | 0.357 | 0.667 |

**Evaluation Metrics** [1].

The study conducted in "An Extensive Comparison of Bug Prediction Approaches"[2] provides a critical evaluation of various models and their effectiveness. Key takeaways include:

- **Performance Metrics:** The study compares traditional and modern techniques using accuracy, precision, recall, and AUC-ROC scores.
- **Key Features:** It highlights the most influential software metrics in bug prediction, such as cyclomatic complexity, code churn, and coupling metrics.
- **Challenges Identified:** The research discusses common issues, including data imbalance, scalability concerns, and feature engineering complexity.
- **Dataset Usage:** Benchmark datasets like PROMISE and NASA have been used to test the effectiveness of different models.
- **Future Research Areas:** The study suggests the adoption of hybrid AI models and real-time bug prediction integration in DevOps workflows.[2]

The study "Explainable Just-In-Time Bug Prediction: Are We There Yet?"[3] emphasizes the need for explain ability in JIT bug prediction models. Key insights include:

- **Importance of Explainability:** The paper highlights the role of explainable AI (XAI) in making JIT defect predictions interpretable for developers.
- **Techniques for Explainable JIT Bug Prediction:** Methods such as SHAP (SHapley Additive Explanations), LIME (Local Interpretable Model-Agnostic Explanations), and attention-based deep learning models are explored to enhance interpretability.
- **Challenges in Explain ability:** The study discusses difficulties in balancing predictive accuracy with interpretability, as well as limitations of current XAI techniques when applied to commit-level bug prediction.
- **Experimental Findings:** The paper provides a comparative analysis of explainable models versus traditional black-box JIT models, highlighting accuracy versus interpretability trade-offs.
- **Future Research Directions:** Suggestions include the integration of hybrid models combining deep learning and rule-based techniques, incorporation of XAI techniques in real-time DevOps pipelines, and improvements in user trust through enhanced transparency.

Comparative Analysis from the Study Conducted in "A Novel Dimensionality Reduction-based Software Bug Prediction"[4]

**Table -2:** Comparison of Traditional Feature Selection and Dimensionality Reduction Technique

| Method | Traditional Feature Selection | Dimensionality Reduction Technique |
|---|---|---|
| Computational Efficiency | Higher overfitting risk | Limited generalization to specific datasets |
| Overfitting Risk | Manual or rule-based | Slower due to high feature |

| | selection | count |
|---|---|---|
| Model Generalization | Higher overfitting risk | Limited generalization to specific datasets |

The study conducted in "Bug Predictive Models based on Data Analytics and Soft Computing"[5] Key takeaways include.

- **Role of Data Analytics:** Data analytics techniques help identify patterns in defect datasets and improve prediction accuracy.
- **Soft Computing Models:** Methods like fuzzy logic, neural networks, and genetic algorithms enhance adaptability and handle uncertainty in bug prediction.
- **Performance Evaluation:** The study compares models using accuracy, precision, and recall, showing that soft computing methods can reduce false positives.
- **Challenges:** Scalability of soft computing for large projects and the need for explain ability in these models.

The study conducted in "Software Bug Prediction Employing Feature Selection and Deep Learning" [6] Key takeaways include.

- **Impact of Feature Selection:** The study investigates how feature selection affects the performance of bug prediction models. By reducing the number of features, the models become more efficient and potentially more accurate.
- **Deep Learning Application:** The research applies deep learning algorithms to software bug prediction, demonstrating that these advanced models can effectively learn from complex data patterns, leading to improved prediction accuracy.
- **Experimental Validation:** Through empirical experiments, the study shows that combining feature selection with deep learning results in better performance compared to traditional methods, detecting up to 32.22% more bugs.

The study conducted in "The Effect of Programming Language in Software Bug Prediction" [7] Key takeaways include.

- **Programming Language Influence:** The study examines how different programming languages, such as Java, C, C++, and Python, impact software bug prediction. It highlights that language-specific features can affect the accuracy of bug prediction models.
- **Empirical Analysis:** Through empirical analysis, the research demonstrates that certain programming languages exhibit distinct bug patterns, which can be leveraged to improve the performance of bug prediction models.

## 4. LITERATURE REVIEW

| Reference | Focus Area | Key Contributions |
|---|---|---|
| [1] D. Al-Fraihat et al. (2024) | Hyperparameter Optimization | Optimizing bug prediction models using ensemble learning techniques. |
| [2] M. D'Ambros et al. (2010) | Comparative Analysis | Extensive comparison of bug prediction approaches based on historical defect data. |
| [3] R. Aleithan (2021) | Explainable AI (XAI) | Discusses the importance of XAI in just-in-time (JIT) bug prediction models. |
| 4] A. Gupta (2023) | Dimensionality Reduction | Highlights the role of dimensionality reduction in improving bug prediction accuracy. |
| [5] V. Jadhav et al. (2023) | Soft Computing & Data Analytics | Examines the impact of soft computing techniques on bug prediction accuracy. |
| [6] S. M. Abozeed(2020) | Feature Selection & Deep Learning | Investigates how deep learning and feature selection improve bug prediction models. |
| [7] S. Sztwiertnia(2021) | Programming Language Impact | Analyzes how different programming languages affect software bug prediction accuracy. |

## 5. CONCLUSIONS

The review of software bug prediction techniques demonstrates the significant evolution from traditional rule-based approaches to AI-driven methodologies. Research studies have shown that deep learning and ensemble learning models outperform conventional statistical methods by leveraging complex data patterns and optimizing hyperparameters [1]. The effectiveness of dimensionality reduction techniques in improving prediction accuracy and computational efficiency has also been highlighted [4]. Furthermore, the integration of explainable AI (XAI) methods ensures transparency in bug prediction models, which is essential for their practical adoption in software development [3].

Soft computing-based models, including fuzzy logic and genetic algorithms, have emerged as promising alternatives to conventional techniques by enhancing adaptability and handling uncertainty [5]. Additionally, feature selection techniques, combined with deep learning, have been proven to improve model efficiency and reduce false positives [6]. The influence of programming languages on bug prediction further emphasizes the need to tailor predictive models according to language-specific characteristics [7].

Despite these advancements, key challenges such as data imbalance, feature selection complexity, and generalization across different software projects remain. Future research should focus on improving scalability, integrating predictive analytics with DevOps workflows, and further developing hybrid AI models that enhance both accuracy and interpretability. The continued evolution of software bug prediction techniques will play a crucial role in ensuring software quality and reducing maintenance costs in real-world applications.

## REFERENCES

1. D. Al-Fraihat, Y. Sharrab, A.-R. Al-Ghuwairi, H. Alshishani, and A. Algarni, "Hyperparameter Optimization for Software Bug Prediction Using Ensemble Learning," *IEEE Access*, vol. 12, pp. 51869-51878, 2024, doi:10.1109/ACCESS.2024.3380024.

2. M. D'Ambros, M. Lanza, and R. Robbes, "An Extensive Comparison of Bug Prediction Approaches," in *Proceedings of the 7th IEEE Working Conference on Mining Software Repositories*, 2010, pp. 31–41. doi: 10.1109/MSR.2010.5463279.

3. R. Aleithan, "Explainable Just-In-Time Bug Prediction: Are We There Yet?" in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, 2021, pp. 129–131. doi: 10.1109/ICSE-Companion52605.2021.00056.

4. Gupta, "A Novel Dimensionality Reduction-based Software Bug Prediction," in *2023 13th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India, 2023, pp. 123-128. doi: 10.1109/Confluence.2023.10048829.

5. V. Jadhav, P. Devale, and R. V. Bidwe, "Bug Predictive Models based on Data Analytics and Soft Computing Techniques: A Survey," in *Proceedings of the International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2023, pp. 123-130.

6. S. M. Abozeed, M. Y. ElNainay, S. A. Fouad, and M. S. Abougabal, "Software Bug Prediction Employing Feature Selection and Deep Learning," in *Proceedings of the 2019 International Conference on Advances in the Emerging Computing Technologies (AECT)*, 2020, pp. 1-6. doi: 10.1109/AECT47998.2020.9194215

7. S. Sztwiertnia, M. Grübel, A. Chouchane, D. Sokolowski, K. Narasimhan, and M. Mezini, "Impact of Programming Languages on Machine Learning Bugs," in *Proceedings of the 1st ACM International Workshop on AI and Software Testing/Analysis (AISTA)*, Virtual, Denmark, 2021, pp. 9-12. doi: 10.1145/3464968.3468408.