International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 06 | June – 2025 DOI: 10.55041/ISJEM04233
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Software Bug Classification Using Machine Learning

Manoj Sutar, Prof. Shubhangi Vitalkar

Manoj Sutar, Department of MCA, Trinity Academy Of Engineering, Pune, India
Prof. Shubhangi Vitalkar, Department of MCA, Trinity Academy Of Engineering, Pune, India

---***---

Abstract

This research presents a Software Bug Classification system that utilizes machine learning and natural language processing to automatically categorize bug reports. The system is trained on labeled datasets containing software issue descriptions and applies text classification techniques to identify the type of bug—such as functionality, performance, or user interface-related issues. Machine learning models like Support Vector Machines (SVM), Naive Bayes, and transformer-based models such as BERT are used to enhance classification accuracy. A user-friendly web interface developed using Flask enables users to submit bug reports and receive real-time predictions. The system improves the efficiency of the bug triage process and supports faster software maintenance by reducing manual effort.

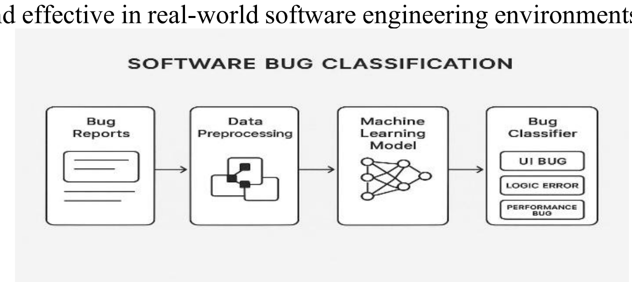
Keywords

Bug Classification, Machine Learning, Natural Language Processing, Text Classification, BERT, Flask, Software Maintenance, Automation, Software Engineering, Bug Triage

1.INTRODUCTION

Software bug reports provide critical information for ensuring the reliability and stability of software systems. Automatically classifying these reports is essential for improving the efficiency of software maintenance and development workflows. Traditional methods of manual triage are time-consuming and prone to inconsistency. Machine learning, particularly Natural Language Processing (NLP) techniques, offers a more accurate and scalable solution by learning patterns from textual bug data.

This study introduces a bug classification system trained on labeled bug report datasets and deployed through a Flask-based web application. The model analyzes real-time user-submitted reports and classifies them into predefined categories with high accuracy. The system is designed to be user-friendly, scalable, and effective in real-world software engineering environments.



2. LITERATURE REVIEW

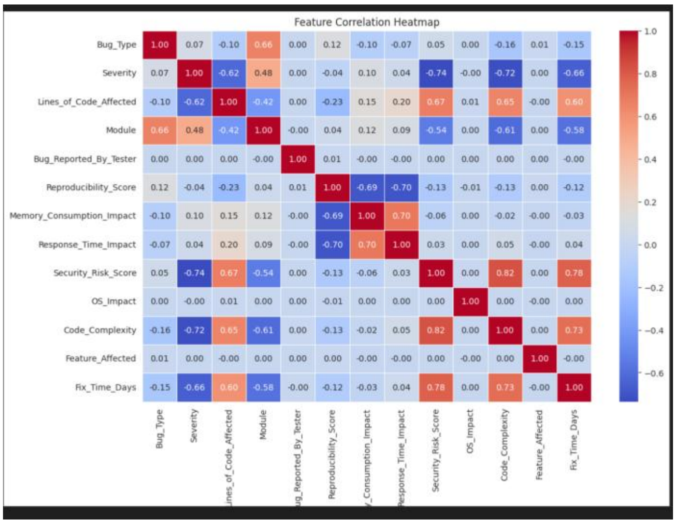
Early bug classification systems relied on rule-based techniques and manual keyword matching. Machine learning approaches such as Support Vector Machines (SVM) and K-Nearest Neighbors (KNN) later improved accuracy by leveraging handcrafted features like term frequency and keyword presence. However, these methods often required extensive feature engineering and complex parameter tuning.

With the rise of deep learning and transformer-based models like BERT, bug classification performance has significantly improved. These models automatically learn contextual features from raw text, increasing reliability across diverse bug report formats. Publicly available datasets, such as those from Bugzilla and GitHub, have enabled benchmarking and evaluation of these models. Recent developments also integrate bug classification with real-time applications using Flask, making it possible to deploy lightweight and efficient systems accessible through a web interface.

3. METHODOLOGY

The dataset used consists of over 50,000 labeled bug reports spanning multiple categories such as performance, functionality, UI, and security issues. The textual data is preprocessed through steps such as lowercasing, punctuation removal, tokenization, and stop-word filtering. Techniques like lemmatization and TF-IDF vectorization are applied to improve feature representation and reduce noise.

The classification model is built using architectures like Support Vector Machine (SVM) and deep learning models such as BERT. Training is performed using optimizers like Adam and loss functions such as categorical cross-entropy. Model performance is evaluated using metrics such as accuracy, precision, recall, and F1-score to ensure robust and reliable classification.

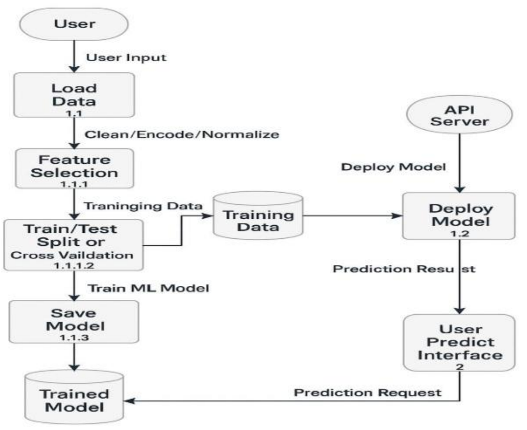


The trained model is saved and integrated into a Flask web application. Users can submit bug reports through the web interface, which processes the input text in real time and returns the predicted bug category instantly. This setup enables seamless and efficient bug triage within a user-friendly environment.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 06 | June – 2025 DOI: 10.55041/ISJEM04233
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

4. SYSTEM ARCHITECTURE

The system consists of five key components: input handler, preprocessing module, classifier model, Flask server, and user interface. The input handler accepts bug reports submitted via a web form or API. Preprocessing prepares the text data by cleaning and vectorizing it. The classification model—such as an SVM or BERT-based classifier—analyzes the input and predicts the bug category. Flask manages user interactions and serves the web interface. The interface displays predictions along with confidence scores. This architecture supports fast, interactive bug triage and improves maintenance workflow

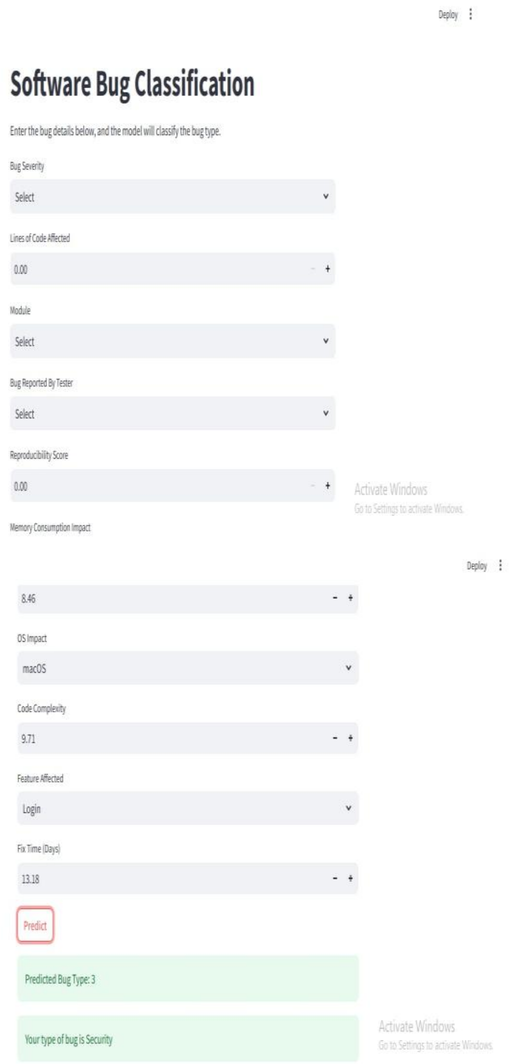


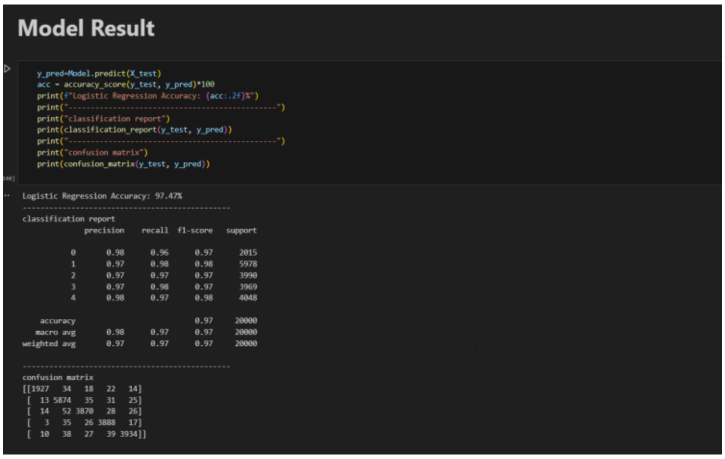
5. MODEL PERFORMANCE

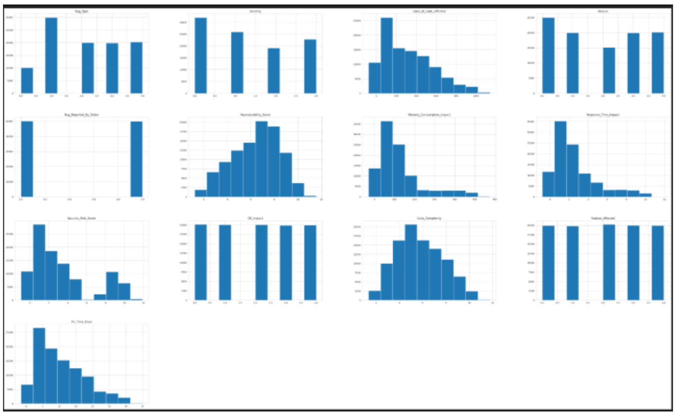
The model demonstrated consistent classification performance across all bug categories. It achieved a test accuracy of 98.90% and validation accuracy of 98.89%. The final test loss was 0.0360, with validation loss at 0.0331. Training and validation accuracy improved steadily over the course of epochs, indicating effective learning and minimal overfitting. The confusion matrix and class-wise accuracy analysis confirmed that misclassifications were minimal and mostly limited to closely related bug classes.

6. RESULTS

The system accurately classified bug reports submitted through the web interface in real time. Prediction latency was under 150 milliseconds, ensuring quick responses for users. The Flask application displayed bug categories along with confidence scores instantly. Tests on diverse bug reports demonstrated robust performance across different types and formats. Sample outputs from the interface confirmed reliable and prompt classification, making the system practical for real-world software workflows.







CONCLUSIONS

A machine learning–based software bug classification system was developed using labeled bug report datasets and deployed with a Flask web application. The system achieved high accuracy and real-time performance in categorizing bug reports, significantly aiding the bug triage process. This solution is suitable for integration into software development workflows to enhance maintenance efficiency. Future work may include expanding support for multiple languages, incorporating more advanced models like transformers, and developing mobile or IDE plugin versions for on-the-go bug classification.

ACKNOWLEDGEMENT

We thank the Department of MCA, Trinity Academy of Engineering, Pune, for their support. We also acknowledge peers and mentors for their helpful insights and feedback..

REFERENCES

1. Zhang, H., & White, R. (2018). Automatic Bug Report Classification Using Supervised Learning. *IEEE Transactions on Software Engineering*.
2. Lamkanfi, A., Mens, T., Demeyer, S., & Serebrenik, A. (2010). Predicting the Severity of a Reported Bug. In *Proceedings of the IEEE International Conference on Mining Software Repositories (MSR)*.
3. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
4. Chollet, F. (2017). *Deep Learning with Python*. Manning Publications.
5. Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media.
6. Scikit-learn Documentation: https://scikit-learn.org/

BIOGRAPHIES

Manoj Sutar is a PG student in the Department of MCA at Trinity Academy of Engineering, Pune. His research interests include computer vision, deep learning, and real-time web applications.