**Abstract**

Software bugs are a part of building software itself that user can lead to inefficiency, security holes and high maintenance costs. We are a machine learning-based project on automation in the classification of software bugs. Utilizing feature engineering and tree-based algorithm Random Forests [x.], the proposed system automatically sorts bugs into predefined categories based on a library of historical data.x This will not only help to make the process of bug triaging more efficient, but also improve the overall quality of the software itself by reducing manual effort involved in categorizing bugs. These findings suggest an exciting avenue for real-world application to enhance workflows for software testing and development.

**1.Introduction**

Detecting and classifying software bugs are vital processes in the software development lifecycle, significantly affecting the quality and dependability of software systems. People can classify bugs, but that is slow and susceptible to error, especially as bugs become ever more complex in the modern age. Automated malfunction classification based on machine learning (ML) is a promising method to overcome those challenges.

Aiming to explore the development of an ML powered bug classification system. Then, it analyzes structured data (like the bug description and logs) and assigns the bug to a predefined category (like functional, security, performance, etc.). The implementation uses tree-based algorithms, specifically Decision Trees and Random Forests for classification and has inbuilt handling for challenges such as class imbalance and scalability.

**2.Related Work in Automated Bug Classification**

**Overview of Existing Approaches**

**Machine Learning-Based Bug Classification Techniques**

1. **Traditional Machine Learning Approaches** 
   * Early research focused on applying classical machine learning algorithms to bug classification:
     + Support Vector Machines (SVMs)
     + Decision Trees
     + Naive Bayes Classifiers
   * These methods primarily used structured features extracted from bug reports and software metrics
2. **Feature Engineering Strategies** 
   * Patil et al. (2019) demonstrated significant improvements in bug classification through:
     + Sophisticated feature extraction techniques
     + Combining code complexity metrics
     + Incorporating text-based features from bug descriptions
   * Key feature categories included:
     + Code structural metrics
     + Historical bug data
     + Developer-related attributes
     + Textual description analysis

**Advanced Machine Learning Approaches**

1. **Deep Learning Innovations** 
   * **Tan and Chuah (2021) systematic literature review highlighted:** 
     + Emergence of neural network-based approaches
     + Transformer models for bug report analysis
     + Sequence-to-sequence learning techniques
   * Advantages:
     + Better handling of unstructured data
     + Improved feature representation
     + Capability to capture complex patterns in bug reports
2. **Ensemble and Hybrid Methods** 
   * Random Forest and Gradient Boosting techniques showed promising results
   * Key benefits:
     + Reduced overfitting
     + Improved generalization
     + Handling of complex, multi-dimensional datasets

**3.Problem Statement**

In software development, identifying and classifying bugs efficiently is crucial for maintaining code quality and ensuring timely releases. Traditional methods of bug classification often rely on manual inspection, which can be time-consuming and prone to human error. To address this challenge, we propose a machine learning approach using Random Forest to automate the classification of software bugs.

**4.Methods and Dataset for Automated Bug Classification**

**4.1 Datasets**

To train and validate the model, publicly available datasets from Kaggle will be utilized:

* **Bug Hunter Dataset**: [Bug Hunter Data](https://www.kaggle.com/datasets/vellyy/bug-hunter/data)
* **Bugzilla Bug Reports**: [Bugzilla Reports](https://www.kaggle.com/datasets/qicongliu/bugzilla-bug-reports)
* **Cleaned Bugzilla**: [Cleaned Bugzilla](https://www.kaggle.com/datasets/qicongliu/cleaned-bugzilla)

These datasets include structured and unstructured data, such as bug descriptions, logs, and metadata, enabling feature extraction and analysis.

**4.2 Algorithms**

The system uses tree-based machine learning model, including:

**Decision Trees:** To make decisions by splitting data into subsets based on the most significant attribute, aiming to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

**Random Forests**: To enhance generalization and reduce overfitting by aggregating predictions from multiple decision trees.

**4.3 Improvements Over Existing Approaches**

While existing implementations focus on detecting bugs, this project extends functionality to:

1. **Classify Bug Categories**: Categorizing bugs into functional, security, or performance-related classes.
2. **Prioritize Bugs**: Assigning severity levels to enable prioritization of critical issues.
3. **Feature Engineering**: Leveraging software metrics and bug descriptions for improved classification accuracy.
4. **Class Imbalance Handling**: Using oversampling techniques and weighted loss functions.

**5. Challenges**

**5.1 Dataset Challenges**

* **Data Quality**: Incomplete and inconsistent bug descriptions.
* **Class Imbalance**: Unequal representation of bug categories in datasets.

**5.2 Scalability**

* Ensuring the model’s performance with large datasets and real-time classification requirements.

**5.3 Integration**

* Compatibility with existing bug tracking tools and DevOps workflows.

**6. Evaluation Metrics**

The project employs both qualitative and quantitative methods for evaluation:

**6.1 Qualitative Evaluation**

* **Visualization**: Plots like confusion matrices, ROC curves, and feature importance charts.
  1. **Quantitative Evaluation**
* **Performance Metrics**: Accuracy, Precision, Recall, F1-score, and Macro/Micro Averaging.

**7. Project insights**

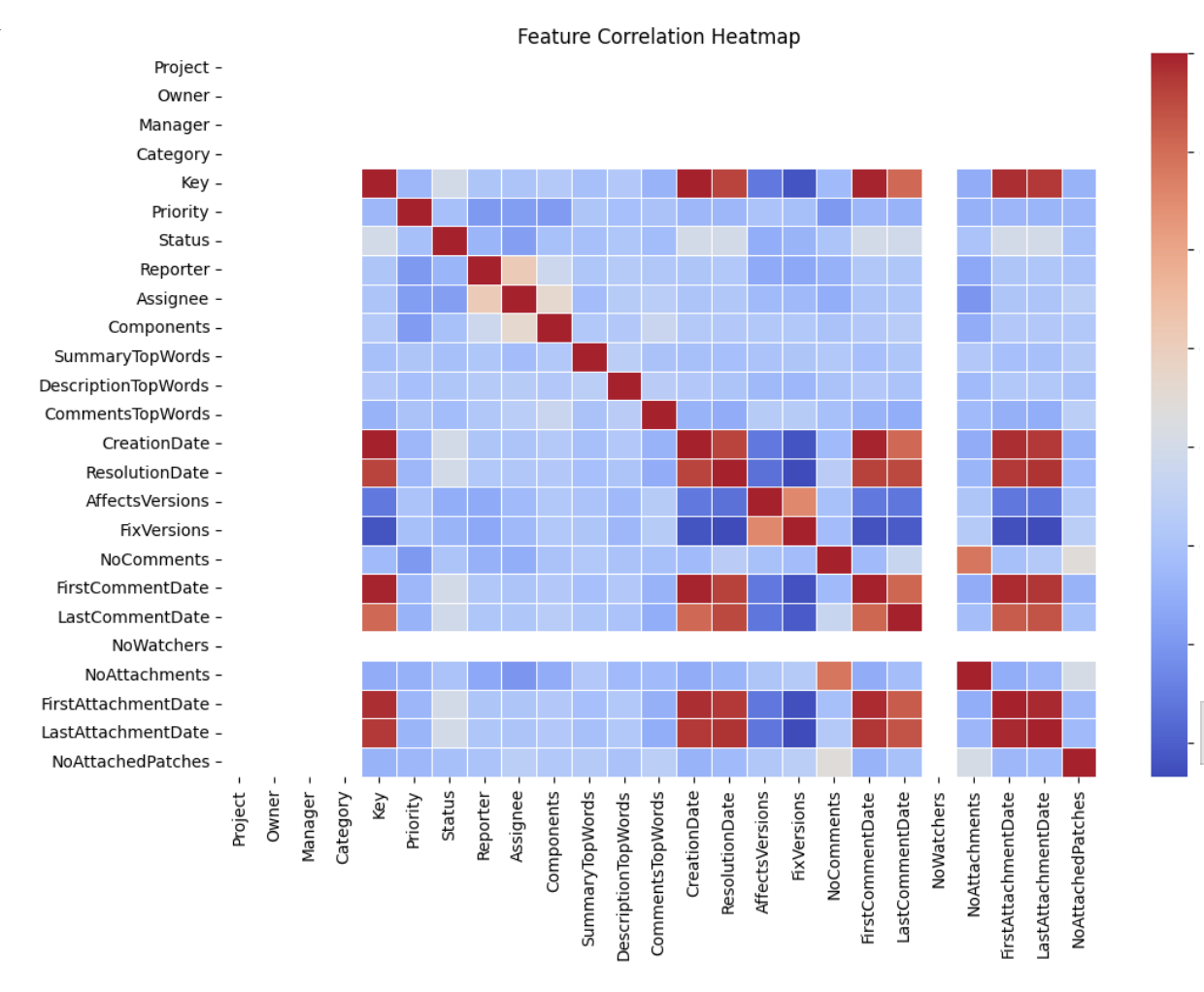


Figure Co-relation matrix

A close-up of numbers

Description automatically generated

Figure 2 Model comparission

A graph with numbers and squares

Description automatically generated

Figure 3 Confusion matrix-random forest

A screenshot of a graph

Description automatically generated

Figure 4 Evaluation scores of random forest

**7. Limitations**

* **Dataset Dependency**: Limited availability of high-quality datasets for training and evaluation.
* **Scalability Issues**: Potential computational overhead for large-scale systems.
* **Interpretability**: Difficulty in explaining model decisions in complex scenarios.
* **Real-Time Constraints**: Ensuring near-instantaneous bug classification for integration into CI/CD pipelines.

**8. Conclusion and Future Work**

This project demonstrates the feasibility and potential of automating software bug classification using machine learning. By addressing the challenges of dataset quality, class imbalance, and scalability, the proposed system aims to improve software quality and efficiency significantly. Future work will involve:

* Exploring advanced ML techniques, such as transformers, for handling unstructured bug descriptions.
* Integrating the model with popular bug tracking tools like Jira.
* Extending the classification system to predict the root cause of bugs.

**References**

[1] R. K. et al., “Feature Engineering for Bug Classification,” *Automated Classification of Software Bug Reports*. [Online]. Available: <https://www.researchgate.net/publication/336457672_Automated_Classification_of_Software_Bug_Reports>  
[2] L. Breiman, “Random Forests,” *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001. [Online]. Available: <https://link.springer.com/article/10.1023/A:1010933404324>  
[3] J. R. Quinlan, “Induction of Decision Trees,” *Machine Learning*, vol. 1, no. 1, pp. 81-106, 1986. [Online]. Available: <https://link.springer.com/article/10.1007/BF00116251>  
[4] “Deep Learning for Software Bug Classification: A Systematic Literature Review,” *Expert Systems with Applications*. [Online]. Available: <https://www.sciencedirect.com/science/article/abs/pii/S0957417421000324>