International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

A Comparative Study of Machine Learning Algorithm for Software Defect Prediction

Mr. Vijay Anand Yadav (M. Tech Scholar)
Dr. D L Gupta (Professor)
vijayanand301297@gmail.com
dlgupta@knit.ac.in

Department of Computer Science & Engineering KNIT SULTANPUR

Abstract
Software defects are among the leading causes of software failure, causing performance degradation, increased costs, and compromised reliability. Software Defect Prediction (SDP) models aim to identify fault-prone software components early in the development lifecycle using machine learning (ML) techniques. This research introduces a hybrid machine learning-based SDP model that integrates Random Forest, Support Vector Machines (SVM), Convolutional Neural Networks (CNN), and Logistic Regression to predict software bugs with improved accuracy. The proposed model is trained and evaluated using NASA and Eclipse datasets. Results show significant improvements in precision, recall, and F1-score over conventional models, offering a practical and scalable solution for real-world software quality assurance.

1. Introduction

Modern software development demands high reliability and rapid delivery. However, increasing complexity in software systems has led to a higher risk of defects, or "bugs," which can compromise functionality and security. Identifying and resolving these defects late in the Software Development Lifecycle (SDLC) is both time-consuming and costly.

Software Defect Prediction (SDP) provides a predictive solution by identifying defect-prone components using historical software data and statistical patterns. Machine learning (ML) plays a central role in this process, allowing automated systems to learn from labelled defect data and make predictions on unseen components. This paper focuses on building a robust SDP model that integrates multiple ML algorithms to improve defect prediction performance across diverse datasets.

2. Background and Motivation

Traditional defect identification relies heavily on testing and code reviews. These manual processes are expensive and become inefficient as the codebase grows. Moreover, not all software modules are equally fault-prone; hence, prioritizing which modules to test becomes a crucial decision.

The need for automated, intelligent tools to predict software defects led to the development of SDP models using machine learning. While existing approaches using individual algorithms like Decision Trees or Naïve Bayes have shown promise, they struggle with imbalanced data and fail to generalize across projects.

To address these challenges, we propose a **hybrid Software Defect Prediction Model** that combines multiple ML classifiers leveraging their strengths and applies them to real-world software datasets. Our model supports early fault localization, improves code quality, and reduces the cost and effort of testing.

3. Related Work

Numerous machine learning approaches have been applied to software defect prediction. These include classical methods such as Logistic Regression, Naïve Bayes, and Decision Trees, as well as more advanced algorithms like Random Forest and SVMs. Recent research has introduced deep learning models, such as CNNs and RNNs, which excel at handling large, complex datasets and unstructured information like bug reports.

However, individual classifiers often perform inconsistently across datasets. Researchers have thus turned to hybrid models and ensemble methods to combine the predictive strengths of various classifiers. Our proposed model builds on this idea by integrating Random Forest, CNN, and SVM classifiers with Logistic Regression for final prediction.

4. Software Defect Prediction Model (Proposed Approach)

The proposed SDP model consists of three primary phases:

4.1 Data Preprocessing

Datasets Used: NASA Promise datasets (e.g., PC1, KC1, MC1) and Eclipse datasets (JDT, PDE UI, Equinox).

Data Cleaning: Removal of null values, duplicates, and non-informative features.

Feature Selection: Correlation-based Feature Selection (CFS) and Principal Component Analysis (PCA) are applied to reduce dimensionality and eliminate redundant features.

Class Imbalance Handling: SMOTE (Synthetic Minority Over-sampling Technique) is used to balance the defective and non-defective class distributions.

4.2 Model Architecture

The proposed SDP model follows a hybrid, ensemble-based architecture:

Step 1: Train base classifiers independently:

- **Random Forest (RF)**: Provides strong performance in high-dimensional datasets with bagging.
- **Support Vector Machine (SVM)**: Captures non-linear patterns using kernel tricks.
- **Convolutional Neural Network (CNN)**: Extracts hierarchical features from structured and semi-structured data.

Step 2: Use **Logistic Regression** to combine the prediction probabilities from RF, SVM, and CNN into a final classification output (defective or non-defective).

4.3 Prediction Workflow

1. Input software module metrics (e.g., lines of code, cyclomatic complexity).
2. Preprocess and transform data into structured input.
3. Each model (RF, SVM, CNN) generates a probability score.
4. Logistic Regression aggregates these scores and predicts final output.

5. Experimental Setup

Experiments were conducted using Python and MATLAB on benchmark datasets. The datasets contain multiple software modules labelled as "defective" or "non-defective" based on historical bug data. Key characteristics of datasets include:

- **Features**: Metrics such as LOC, coupling, cohesion, inheritance, complexity.
- **Output**: Binary class indicating presence or absence of a defect.

To evaluate performance, we applied **10-fold cross-validation**. Evaluation metrics include:

- **Accuracy**: (TP + TN) / Total
- **Precision**: TP / (TP + FP)
- **Recall (Sensitivity)**: TP / (TP + FN)
- **F1-Score**: 2 × (Precision × Recall) / (Precision + Recall)

6. Results and Analysis

The results reveal that the proposed hybrid model consistently outperforms standalone models across all datasets. Highlights include:

Dataset	Accuracy	Precision	Recall	F1-Score
PC1	93.25%	72.72%	78.04%	75.25%
KC1	91.14%	70.35%	76.40%	73.25%
JDT	90.42%	69.81%	74.88%	72.25%

Key observations:

- Random Forest performed well with fewer tuning parameters and was robust to overfitting.
- CNN, supported by dropout layers and word embeddings, extracted complex patterns from source code metrics.
- SVM struggled slightly with large, high-dimensional datasets but contributed positively in ensemble mode.
- Logistic Regression effectively combined model outputs and reduced variance.

Feature selection improved training efficiency, and SMOTE helped address the class imbalance problem.

7. Conclusion

This research presents a comprehensive machine learning-based Software Defect Prediction Model that integrates multiple algorithms into a hybrid framework. The model leverages the predictive strengths of Random Forest, SVM, CNN, and Logistic Regression to identify defect-prone software modules with high accuracy.

By applying the model to real-world datasets, we demonstrate its potential for practical use in software development environments. Early defect detection using this model allows developers to focus on risky modules, improving software quality and reducing costs.

Future enhancements could include integrating graph neural networks, attention mechanisms, and real-time deployment in CI/CD pipelines. Additionally, explainable AI techniques can be employed to make defect prediction decisions more transparent to developers.

References

1. Arora, I.; Tetarwal, V.; Saha, A. Open issues in software defect prediction. Procedia Comput. Sci. 2015, 46, 906–912. [Google Scholar] [CrossRef] [Green Version]

2. Ali, M.M.; Huda, S.; Abawajy, J.; Alyahya, S.; Al-Dossari, H.; Yearwood, J. A parallel framework for software defect detection and metric selection on cloud computing. Clust. Comput. 2017, 20, 2267–2281. [Google Scholar] [CrossRef]
3. Yadav, H.B.; Yadav, D.K. A fuzzy logic based approach for phase-wise software defects prediction using software metrics. Inf. Softw. Technol. 2015, 63, 44–57. [Google Scholar] [CrossRef]
4. Iqbal, A.; Aftab, S.; Ali, U.; Nawaz, Z.; Sana, L.; Ahmad, M.; Husen, A. Performance analysis of machine learning techniques on software defect prediction using NASA datasets. Int. J. Adv. Comput. Sci. Appl. 2019, 10, 300–308. [Google Scholar] [CrossRef] [Green Version]
5. Khalid, A., Badshah, G., Ayub, N., Shiraz, M., & Ghouse, M. (2022). Software Defect Prediction Analysis Using Machine Learning Techniques. Sustainability, 15(6), 5517. https://doi.org/10.3390/su15065517