

An explainable AI-based approach for Java code smell detection to improve software maintainability

Wedaarachchi R. ^{1*}, Herath G.A.C.A. ² and Wasalthilaka W.V.S.K. ³

¹Department of Software Engineering, Faculty of Computing,
Sabaragamuwa University of Sri Lanka, Sri Lanka

²Department of Computing and Information Systems, Faculty of Computing,
Sabaragamuwa University of Sri Lanka, Sri Lanka

³Faculty of Computer Science, Technical University of Dresden

*rwedaarachchi@std.appsc.sab.ac.lk

Code smells are early indicators of design flaws that compromise long term software maintainability. Traditional static analysis tools often generate false positives or fail to detect context dependent smells because they rely on fixed thresholds. As software systems become sophisticated, machine learning (ML) approaches provide a viable alternative for capturing deeper structural and semantic patterns. In order to detect three maintainability critical code smells in Java applications Complex Method, Long Parameter List, and Multifaceted Abstraction; this study proposes an explainable machine learning driven methodology. Using the DACOS dataset, containing 10,341 method level and 4,424 class level samples, separate binary classification pipelines were developed for each smell to guarantee sufficient granularity and domain specific feature engineering. Models, including XGBoost, LightGBM, Random Forest, Gradient Boosting, Decision Tree, SVM, KNN, and Logistic Regression, were trained using an 80/20 stratified split for preserving class distribution, SMOTETomek for resampling, and Grid-SearchCV for hyperparameter tuning to enable an unbiased performance assessment. From intending individual model performance, the top three models for each code smell were integrated to build weighted voting ensemble model to further improve robustness. The findings suggest relatively consistent performance across the cross-validation folds. For Multifaceted Abstraction, XGBoost achieved an accuracy of 0.9401 (SD=0.0043) and an F1 score of 0.9073 (SD=0.0059), while for Complex Method and Long Parameter List, XGBoost achieved accuracy 0.8222 (SD=0.0101) and 0.8104 (SD=0.0085), respectively. Key features were highlighted in the instance level explanations provided by LIME. These insights increase transparency and enable better informed refactoring approaches. However, the results are derived from only a single dataset and a limited number of code smell type, which may limit their generalizability and external validity. Overall, this study presents the potential of a reliable, precise, and interpretable ML based technique for automated code smell detection within the evaluated context to software maintainability.

Keywords: *Code smells, Explainable AI, Java, Machine learning, Software maintainability*