

ABSTRAK

Long Method merupakan salah satu jenis *code smell* yang dapat menurunkan keterbacaan, keterpeliharaan, dan kemudahan pengujian kode program. Pada pengembangan aplikasi Android berbasis Kotlin, karakteristik sintaksis seperti *lambda*, *safe call*, *scope function*, *coroutine*, dan komponen Jetpack Compose dapat membentuk struktur *method* yang berbeda dari bahasa pemrograman berorientasi objek konvensional. Oleh karena itu, analisis fitur berbasis *static analysis* diperlukan untuk mengidentifikasi karakteristik kode yang berkaitan dengan label operasional *Long Method* pada kode sumber Android Kotlin. Penelitian ini bertujuan untuk menganalisis fitur *static analysis* berbasis Kotlin dalam mendukung deteksi label operasional *Long Method*. Dataset yang digunakan merupakan dataset *method-level* Android Kotlin dari repositori *open-source* F-Droid dengan total 320.602 *method* valid. Label operasional dibentuk menggunakan aturan $LOC > 60$, sehingga diperoleh 14.563 *method* berlabel *Long Method* dan 306.039 *method* berlabel *Non-Long Method*. Validasi label dilakukan dengan membandingkan hasil pipeline terhadap Detekt *LongMethod* pada ambang yang setara. Hasil validasi menunjukkan Cohen's kappa sebesar 0,804656, sehingga label operasional memiliki konsistensi kuat terhadap *static analysis tool* eksternal. Setelah fitur pembentuk label dan fitur yang berpotensi menjadi *proxy* langsung dikeluarkan, diperoleh 38 fitur kandidat. Seleksi fitur menggunakan *Recursive Feature Elimination* menghasilkan 25 fitur prediktif akhir atau RFE25. Analisis parsimoni menggunakan *L1 Logistic Regression* dan *one-standard-error rule* menunjukkan bahwa konfigurasi parsimonious masih mempertahankan 22 fitur dengan MCC validasi sebesar 0,764181. Hasil ini menunjukkan bahwa RFE25 masih dapat dipertanggungjawabkan sebagai *predictive feature set*. Model final yang digunakan adalah *HistGradientBoostingClassifier* tanpa *sample weight* dengan *threshold* 0,342650. Evaluasi akhir pada *held-out test set* menghasilkan MCC sebesar 0,751351, *F1-score* sebesar 0,760964, *precision* sebesar 0,742709, *recall* sebesar 0,780138, PR-AUC sebesar 0,850354, dan ROC-AUC sebesar 0,987469. Hasil penelitian menunjukkan bahwa fitur *static analysis* berbasis Kotlin dapat digunakan untuk mendukung deteksi label operasional *Long Method*.

Kata kunci— *Long Method*, Kotlin PSI, *code smell*, *feature selection*, *Recursive Feature Elimination*, Analisis Statis

ABSTRACT

Long Method is a type of code smell that may reduce code readability, maintainability, and testability. In Android development using Kotlin, language-specific constructs such as lambdas, safe calls, scope functions, coroutines, and Jetpack Compose components may produce method structures that differ from those commonly found in conventional object-oriented programming languages. Therefore, a static analysis-based feature analysis is required to identify code characteristics associated with the operational label of Long Method in Android Kotlin source code. This study aims to analyze Kotlin-based static analysis features to support the detection of operational Long Method labels. The dataset used in this study is an Android Kotlin method-level dataset derived from open-source F-Droid repositories, consisting of 320,602 valid methods. The operational label was defined using the rule $LOC > 60$, resulting in 14,563 methods labeled as Long Method and 306,039 methods labeled as Non-Long Method. Label validation was conducted by comparing the pipeline labels with Detekt LongMethod findings under an equivalent threshold. The validation result produced a Cohen's kappa value of 0.804656, indicating strong consistency between the operational labels and an external static analysis tool. After removing label-forming features and features that could act as direct proxies for the labeling rule, 38 candidate features were retained. Feature selection using Recursive Feature Elimination produced 25 final predictive features, referred to as RFE25. A parsimony analysis using L1 Logistic Regression and the one-standard-error rule showed that the parsimonious configuration still retained 22 features with a validation MCC of 0.764181. This result indicates that RFE25 remains defensible as the final predictive feature set. The final model used in this study was HistGradientBoostingClassifier without sample weighting, with a threshold of 0.342650. Final evaluation on the held-out test set produced an MCC of 0.751351, an F1-score of 0.760964, a precision of 0.742709, a recall of 0.780138, a PR-AUC of 0.850354, and a ROC-AUC of 0.987469. The findings indicate that Kotlin-based static analysis features can support the detection of operational Long Method labels.

Keywords— Long Method, Kotlin PSI, code smell, feature selection, Recursive Feature Elimination, static analysis