A Partial Replication of Code Smell Detection Using Machine Learning Techniques

Reference work. F. Arcelli Fontana, M.V. Mäntylä, M. Zanoni, A. Marino: Comparing and experimenting machine learning techniques for code smell detection. Empirical Software Engineering 21(3):1143-1191 (2016). *Replication study*: D. Di Nucci, F. Palomba, D.A. Tamburri, A. Serebrenik, A. De Lucia: Detecting code smells using machine learning techniques: Are we there yet? SANER 2018: 612-621.

WHAT: Code smells are symptoms of poor design and implementation choices weighing heavily on the quality of produced source code. In the reference work the use of Machine-Learning (ML) techniques for code smell detection has been proposed. The results obtained have been interpreted as an indication that "using machine learning algorithms for code smell detection is an appropriate approach" and that the research problem in question is essentially solved.

WHY: The reference work has focussed on datasets containing instances affected by one type of smells or non smelly instances. This is, however, not realistic. Hence, we aim at replicating the study with datasets containing instances of more than one type of smells, besides non smelly instances. Moreover, in the reference work, one third of the instances in the dataset was composed of smelly elements. According to recent findings only a small fraction of a software system is usually affected by code smells: e.g., God Classes represent < 1% of the total classes[1].

HOW: To assess the actual capabilities of ML techniques for code smell detection, in the reference work a large-scale study was conducted where 32 different ML algorithms were applied to detect four code smell types, i.e., Data Class, Large Class, Feature Envy and Long Method. The authors have considered 74 systems from the Qualitas Corpus and reported that most of the classifiers exceeded 95% both in terms of accuracy and of F-Measure, with J48 and RANDOM FOREST obtaining the best performance.

WHERE: We built a new dataset closer to reality by merging the instances contained in the four original datasets. In this way, the new dataset includes instances affected by multiple smells, and smelly instances are more realistically less frequent. We performed exactly the same experiment as in the reference work. The performance of code smell prediction models on the new dataset is up to 90% lower than the one reported. The best performance (for all the smells) is achieved by the tree-based classifiers, i.e., RANDOM FOREST and J48: this confirms the results of the reference study. Thus, we believe that detecting code smells using machine learning techniques is extremely relevant and is still open.

DISCUSSION: Our study was only possible because the authors of the reference work publicly shared the datasets. Hence, sharing data in a findable, accessible, interoperable and reusable[2] way is an enabler of replication: while data sharing is not always possible it should be encouraged by the community. Another important enabler for our work is a well-known state-of-the-art tooling such as WEKA. Availability of data and analysis tools facilitate further replication studies, allow the community to reevaluate established theories and methodologies, improving them and eventually bringing new ones.

[1] F. Palomba, G. Bavota, M. Di Penta, F. Fasano, R. Oliveto, A. De Lucia: On the diffuseness and the impact on maintainability of code smells: a large scale empirical investigation. Empirical Software Engineering 23(3):1188-1221 (2018)
[2] https://www.force11.org/group/fairgroup/fairprinciples