Detection of Shotgun Surgery and Message Chain Code Smells using Machine Learning Techniques

Thirupathi Guggulothu, University of Hyderabad, Hyderabad, India
Salman Abdul Moiz, University of Hyderabad, Hyderabad, India

ABSTRACT

Code smell is an inherent property of software that results in design problems which makes the software hard to extend, understand, and maintain. In the literature, several tools are used to detect code smell that are informally defined or subjective in nature due to varying results of the code smell. To resolve this, machine learning (ML) techniques are proposed and learn to distinguish the characteristics of smelly and non-smelly code elements (classes or methods). However, the dataset constructed by the ML techniques are based on the tools and manually validated code smell samples. In this article, instead of using tools and manual validation, the authors considered detection rules for identifying the smell then applied unsupervised learning for validation to construct two smell datasets. Then, applied classification algorithms are used on the datasets to detect the code smells. The researchers found that all algorithms have achieved high performance in terms of accuracy, F-measure and area under ROC, yet the tree-based classifiers are performing better than other classifiers.

KEYWORDS

Code Smells, Machine Learning, Software Refactoring, Stratified Sampling, Supervised Learning

INTRODUCTION

Code smells or bad code smells refers to an anomaly in the source code that may result in deeper problems which makes software difficult to understand, evolve, and maintain. According to (Booch, 2006) smell is a kind of structure in the code that shows a violation of basic design principles such as Abstraction, Hierarchy, Encapsulation, Modularity, and Modifiability. Even if the design principles are known to the developers due to inexperience, the competition that is in the market and deadline pressure are leading to violation of these principles. Fowler et al. (Fowler, 1999) have defined 22 informal code smells which are removed through refactoring techniques. These techniques are used to enhance the internal structure of the code without varying the external behaviour and to improve the quality of the software. The (Opdyke, 1992) authors have defined 72 refactoring techniques.

There are various methods and tools available in the literature to detect the code smells. Each technique and tool produces different (Fontana, 2012). Bowes et al. (Bowes, 2013), compared two code smell detection tools on message chaining and shown disparity of results between them. The three main reasons for varying results are: 1) The code smells can be subjectively interpreted by the developers, and hence detected in different ways. 2) Agreement between the detectors is low, i.e., different tools or rules detect a different type of smell for different code elements. 3) The threshold value for identifying the smell can vary for the detectors.
To address the above limitations, in particular the subjective nature, Fontana et al. (Fontana, 2016) proposed a machine learning (ML) technique to detect four code smells (Long Method, Data Class, Feature Envy, Large Class) with the help of 32 classification techniques. The authors have built 4 datasets, one for each smell. These datasets have been prepared based on the tools and manual labelling process. Tools are used to identify whether the code elements (instances) are smelly or not. But the tools may produce some false positive instances so, the authors manually validated the instances to avoid the biasness. In this paper, instead of using tools and manual validation, the authors have prepared two new method level code smell datasets of Fowler et al. (Fowler, 1999) from the literature; based on the detection rules and unsupervised learning i.e., clustering to validate the instances as smelly or not.

In the proposed work, an attempt is made to detect two code smells namely Shotgun surgery and Message chaining with supervised learning techniques. It is an application of machine learning (ML) classification approach used for code smell detection. It uses known data to determine how the new instances should be classified into binary classification i.e., based on the metrics used for a particular method, the ML approach helps in classifying a method to be prone to code smell or not. In this paper, the dataset instances are methods of 74 heterogeneous java systems. The metrics of object-oriented systems have been computed on method instances, which are the features or attributes of the dataset. For each smell, one dataset is prepared by using detection rules from the literature (Ferme, 2013). The researchers applied a random stratified sampling on the method instances to balance the datasets. Sample instances of the dataset are validated through unsupervised learning and added to the training dataset. Then applied some known classification algorithms on the trained datasets to detect the code smells, by using 10-fold cross validation method. To evaluate those algorithms, standard metric measures such as F-score, accuracy and the area under the ROC are used. The experimented algorithms have achieved high performance in both the smells.

The paper has been arranged as follows; The second section, introduces a work related to detection of code smells; The third section, defines two proposed approaches of code smell detections; The fourth section, detecting code smells using ML approach; The fifth section, presents experimental results; The sixth section, presents the code smell detection rules; and the final section, gives conclusion and future directions.

RELATED WORK

According to (Kessentini, 2014) approaches of code smell detection are classified into 7 categories (i.e., cooperative-based approaches, visualization-based approaches, search-based approaches, probabilistic approaches, metric-based approaches, symptoms based approaches, and manual approaches). In the manual approach developers and maintainers follow different reading guidelines to detect smells. As it requires human involvement, it consumes more time for large systems. In the metric-based approach, smell detection is based on source code metrics. Symptoms based approach uses different notations to detect smells. But the problem with this approach is, it requires analysis to convert symptoms or notations into detection algorithms. Probabilistic approach is based on applying fuzzy logic rules to detect smells. The visualization approach uses semi-automated processes to detect and visualize the smells with the integration of human capabilities. But the problem with this approach is that, it requires human effort, with increase in large systems. The search-based approach applies different algorithms to detect the smells. Most of the techniques use ML approaches. The success of this approach depends upon the training datasets. The cooperative approach performs different activities in a cooperative way.

Fontana et al. (2015) proposed a detection strategy for the code smells. The authors have derived metric thresholds to detect code smells, from a benchmark of 74 java software systems.

Fontana et al. (2016) experimented and compared the supervised ML algorithms to detect the code smells. The authors have used 74 java systems to prepare the training dataset which are manually
Related Content

Detection of Shotgun Surgery and Message Chain Code Smells using Machine Learning Techniques

IT Outsourcing in the Face of Global and Technology Challenges
[www.irma-international.org/chapter/it-outsourcing-in-the-face-of-global-and-technology-challenges/112477](www.irma-international.org/chapter/it-outsourcing-in-the-face-of-global-and-technology-challenges/112477)

Nigerian Undergraduate Students' Computer Competencies and Use of Information Technology Tools and Resources for Study Skills and Habits' Enhancement
[www.irma-international.org/chapter/nigerian-undergraduate-students-computer-competencies-and-use-of-information-technology-tools-and-resources-for-study-skills-and-habits-enhancement/183942](www.irma-international.org/chapter/nigerian-undergraduate-students-computer-competencies-and-use-of-information-technology-tools-and-resources-for-study-skills-and-habits-enhancement/183942)

Cloud Governance at the Local Communities
[www.irma-international.org/chapter/cloud-governance-at-the-local-communities/183818](www.irma-international.org/chapter/cloud-governance-at-the-local-communities/183818)

Modified LexRank for Tweet Summarization
[www.irma-international.org/article/modified-lexrank-for-tweet-summarization/163105](www.irma-international.org/article/modified-lexrank-for-tweet-summarization/163105)