


A Hybrid Approach to Identify Code Smell Using Machine Learning Algorithms

Archana Patnaik, Department of Computer Science and Engineering, School of Engineering and Technology, GIET University, Gunupur, India

Neelamdhhab Padhy, Department of Computer Science and Engineering, School of Engineering and Technology, GIET University, Gunupur, India

 <https://orcid.org/0000-0002-8512-3469>

ABSTRACT

Code smell aims to identify bugs that occurred during software development. It is the task of identifying design problems. The significant causes of code smell are complexity in code, violation of programming rules, low modelling, and lack of unit-level testing by the developer. Different open source systems like JEdit, Eclipse, and ArgoUML are evaluated in this work. After collecting the data, the best features are selected using recursive feature elimination (RFE). In this paper, the authors have used different anomaly detection algorithms for efficient recognition of dirty code. The average accuracy value of k-means, GMM, autoencoder, PCA, and Bayesian networks is 98%, 94%, 96%, 89%, and 93%. The k-means clustering algorithm is the most suitable algorithm for code detection. Experimentally, the authors proved that ArgoUML project is having better performance as compared to Eclipse and JEdit projects.

KEYWORDS

Code Smell, Dirty Code, Feature Selection, Machine Learning, Software Refactoring

1. INTRODUCTION

The primary cause of code complexity is the time frame, mismanagement, unclear shortcuts during the software development process, lack of testing, documentation issues, lack of understanding, communication issues, lack of teamwork, monitoring issues, workload and late refactoring. Lack of cooperation and coordination often cause these problems. Project transition even harmed the whole project due to nasty coding. Code smell refers to the deeper issue inside a program's source code. These problems occurred because code smell may not affect the result, but it still harms the source code's performance. The absolute violation of basics in developing software results in decreased code quality by increasing the technical debt to identify code smells automatically. Wekanose is a tool used to determine the code smell from any coding using weka software. Other code detection tools are PMD, iplasma, Jdeodrant, Decoder, Checkstyle, etc.

Figure 1, illustrates the dirty code with data clump code complexity where groups of variables are combined to form objects at the class level. It increases the execution time of the program by allocating data values to the variables. In the above Figure data members like ccno, expmonth, expyear and amt consists of some random data values, which further leads to code complexity. It can be avoided by deleting the assigned values.

DOI: 10.4018/IJOSSP.2021040102

Figure 1. Dirty Code

```
class card {  
    void cc(int account_no, String expiry_month, int expiry_year, double amount) {  
        account_no=144222222;  
        expiry_month="April";  
        expiry_year=2021;  
        amount=50000.00;  
        System.out.println("Credit card number="+accout_no);  
        System.out.println("Expiry month="+expiry_month);  
        System.out.println("Expiry year="+expiry_year);  
        System.out.println("Amount="+amount); } }
```

Feature selection is the automatic or manual selection of relevant features from the massive amount of data used to construct the model. It is used to improve the accuracy of a model by reducing its complexity. It is a process of selecting a set of best features in the form of a subset before implementing any generalized algorithms. Various parameters involved for feature selection are correlation, entropy, mutual information. Different types of feature selection methods are Recursive Feature Elimination, Chi-squared test, feature evaluation, etc. Machine learning involves a machine to learn from data by predicting things being programmed automatically. We have used different supervised, unsupervised and anomaly detection algorithms to identify the smelly data from the realtime datasets. In our research, the prime focus is on code smell detection using the identification of outliers. We have used different unsupervised anomaly detection methods like PCA, GMM, autoencoder, K-means clustering, and Bayesian network to identify outliers in the dirty code. We have also focused on the performance of the system by comparing its accuracy. Software quality is defined as the robustness or fitness of a software product's quality. It is analyzed by the following parameters reusability, correctness, portability and maintainability. Software quality assurance produces high-quality software by saving time and cost. Code smell affects the source code by violating the good program designing principles having a negative impact on the software quality. The primary solution to this problem is to develop the refactored code. Refactoring is used to change the internal structure of code without altering its external functionalities. Different types of techniques are replacing parameter, inline method, extract class etc.

RQ1: Which type of feature selection method is used for analyzing the open-source projects?

In this work, feature selection methods reduce complexity and increase the proposed model's efficiency. Recursive Feature Elimination (RFE) is used for selecting the relevant data by removing the weakest features of the dataset.

RQ2: Which type of anomaly detection is more preferable for analyzing the concept of code smell?

This work illustrated the anomaly detection technique for identifying outliers by comparing the dirty code with clean code. We have used five different algorithms to identify the extreme code point that slightly deviated from the original data samples. Cluster-based anomaly detection methods give the best results for code smell detection.

RQ3: What are the most commonly found code smell and suitable refactoring approach for developing clean code?

13 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage: www.igi-global.com/article/a-hybrid-approach-to-identify-code-smell-using-machine-learning-algorithms/280096

Related Content

DistProv-Data Provenance in Distributed Cloud for Secure Transfer of Digital Assets with Ethereum Blockchain using ZKP

Navya Gouruand NagaLakshmi Vadlamani (2019). *International Journal of Open Source Software and Processes* (pp. 1-18).

www.irma-international.org/article/distprov-data-provenance-in-distributed-cloud-for-secure-transfer-of-digital-assets-with-ethereum-blockchain-using-zkp/238007

DistProv-Data Provenance in Distributed Cloud for Secure Transfer of Digital Assets with Ethereum Blockchain using ZKP

Navya Gouruand NagaLakshmi Vadlamani (2019). *International Journal of Open Source Software and Processes* (pp. 1-18).

www.irma-international.org/article/distprov-data-provenance-in-distributed-cloud-for-secure-transfer-of-digital-assets-with-ethereum-blockchain-using-zkp/238007

Open Culture for Education and Research Environment

Ernesto Damiani, Paul G. Mezey, Paolo M. Pumiliaand Anna M. Tammario (2007). *Open Source for Knowledge and Learning Management: Strategies Beyond Tools* (pp. 219-244).

www.irma-international.org/chapter/open-culture-education-research-environment/27813

On the State of Free and Open Source E-Learning 2.0 Software

Utku Kose (2014). *International Journal of Open Source Software and Processes* (pp. 55-75).

www.irma-international.org/article/on-the-state-of-free-and-open-source-e-learning-20-software/124004

Building Open-Source Resources for Online Learning in a Higher Education Environment

Shalin Hai-Jew (2013). *Open-Source Technologies for Maximizing the Creation, Deployment, and Use of Digital Resources and Information* (pp. 115-135).

www.irma-international.org/chapter/building-open-source-resources-online/70122