International Journal of Open Source Software and Processes
Volume 13 • Issue 1

Cross-Project Software Refactoring Prediction Using Optimized Deep Learning Neural Network With the Aid of Attribute Selection

Rasmita Panighrahi, School of Engineering and Technology, GIET University, Gunupur, India*
Sanjay Kumar Kuanar, School of Engineering and Technology, GIET University, Gunupur, India
Lov Kumar, Birla Institute of Technology and Science, Hyderabad, India

ABSTRACT

Cross-project refactoring prediction is prominent research that comprises model training from one project database and testing it for a database under a separate project. While performing the refactoring process on the cross project, software programs want to be restructured by modifying or adding the source code. However, recognizing a piece of code for predicting refactoring purposes remains a challenge for software designers. To date, the entire refactoring procedure is highly dependent on the skills and software inventers. In this manuscript, a deep learning model is utilized to introduce a predictive model for refactoring to highlight classes that need to be refactored. Specifically, the deep learning technique is utilized along with the proposed attribute selection phases to predict refactoring at the class level. The planned optimized deep learning-based method for cross-project refactoring prediction is experimentally conducted on open-source project and accuracy is found to be 0.9648 as compared to other mentioned methods.

KEYWORDS

Cross Project, Deep Learning, Levy Flight, Neural Network, Optimization, Quantum Particle Swarm Optimization, Software Refactoring

1. INTRODUCTION

Application related to software domains are constantly preserved and modified for inserting new necessities like debugging, or changing new elements (Zhang, 2019). Demands continue as per the revolution of the market environment is vulnerable to stakeholder needs. Consequently, software applications need to be constantly updated to ensure that participants are satisfied. During software maintenance, developers are requested to enhance a new feature, replace or/ and remove prevailing codes. These changes are necessary to correct mistakes or to accommodate new needs lifted by the marketplace or stakeholders (Deeb et al. 2021). Modifying these features requires modifying software systems to touch the essential requirement; the refactoring procedure is refactoring (Aniche et al.,

DOI: 10.4018/IJOSSP.300756

*Corresponding Author

2020). Identifying a target code piece for restructuring purposes is problematic for software testers. Therefore, the entire redesign procedure is highly dependent on the skills and software testers. Software Refactoring involves modifying an application's interior structure or its structure deprived of altering the exterior functionality or its functionality. Redesign enhances comprehension, complexity, and maintenance (Hegedűs et al.,2018, Pal,2021). The semantics of code after and before the refactoring process stays similar. The redesign process comprises of changing methods, classes and variables in the software application. It is a technical challenge for software developers to find objects or regions for an extensive complex system to be redesigned. Good quality software is the result of good design. Code refactoring plays a prominent part in improving software quality by altering the interior structure without affecting the intended behavior (AlOmar et al., 2021).

The redesign models counteract the erosion of the software design at the beginning of the software development project in line with the model-driven engineering platform (Badri et al, 2019). The traditional model based on refactoring methods work at a higher level by using the model metric limit values as indicators of the above design and performing local adjustments. Many refactoring studies use quality metric limit values to mark the opportunities of refactoring something (Singh et al., 2018). However, the threshold selection is subjective and cannot be universally applied. Additionally, a software defect (code smells) expressed by individual metrics is high because it results from the primary achievement of design goals (Alizadeh et al.,2018). The prediction of the changes prone model in Cross project is comparatively unfamiliar area and is done with a small number of empirical evidences and studies. It is a model that possesses training from a sub-database of one project and testing it to a database under a diverse code project. Predicting cross-project change-proneness compared to untested (Agnihotri & Chug, 2020). The Cross-project speculation comprises utilizing data from other open source projects for similar model design features and then using a predictive model to identify shifting classes (open source project and different directional projects). Recently, few engineers and researchers have made an effort to smear predictions from one project to another. This method of utilizing information among various projects to create disability models is often called Cross-Project Defect Prediction (CPDP) (Paltoglou et al.,2018, Baqais & Alshayeb,2020). It includes using a project prediction system, designed for another project. The choice of training samples depends on the distribution requirements of the data sets.

Extensive research has been done on the association between reusable software programs and quality ratings (Limsettho et al., 2018). All of these test results show that the refactoring has a direct impact on getting better the quality of software. Consequently, forecasting rapid refactoring model ought to be investigated, and building an accurate model becomes obligatory. These techniques can offer developers with specific information about which part of the code should be redesigned and at what time. Still, Software designers possess the real shortcomings of choosing the exact time and software code for duplicate purposes as performance requires budget and time Tsoukalas et al., 2021). Therefore, engineers should be confident which part of the code should be changed before initiating the refactoring process to accommodate the innovative desires. Various methods have been developed and developed to assist developers in the process of remodelling such as, invariant mines, logic meta-programming, search based (Chug & Tarwani, 2021), Sandrasegaran & Vengusamy, 2021) and code smell identification techniques Patnaik et al., 2020, March). Furthermore, machine learning is harnessed in the field of cross project refactor prediction and demonstrates significant performance in terms of forecasting. Numerous machine learning procedures are used to predict code in method and class level prediction (Sharif et al.,2021). (Martins et al., 2021) proposed refactoring solutions for test smells. Machine learning algorithms will be fed with past implicit or tacit data from software projects. They used a large dataset to make ML methods more generalizable. The idea of gathering software data to detect test smell was presented. We trained seven ML algorithms to recognize test odors. As a result, the algorithms are good at detecting test scents. Instead of machine learning algorithms, dissimilar source code measures are calculated at the level of the method using its features to predict the need to perform refactoring the project. However, manual extraction methods fail to