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

Classification of Software Defects Using Orthogonal Defect Classification

Sushil Kumar, Shyam Lal College, University of Delhi, India
SK Muttoo, University of Delhi, India
V. B. Singh, Jawaharlal Nehru University, India*

ABSTRACT

Classification of software defects is an important task to know the type of defects. It helps to prioritize the defects to understand the cause of defects for improving the process of software defect management system by taking the appropriate action. In this paper, the authors evaluate the performance of naïve bayes, support vector machine, k nearest neighbor, random forest, and decision tree machine learning algorithm to classify the software defect based on orthogonal defect classification by selecting the relevant features using chi-square score. Standard metrics accuracy, precision, and recall are calculated separately for Cassandra, HBase, and MongoDB datasets. The proposed method improves the existing approach in terms of accuracy by 5%, 20%, 6%, 27%, and 26% for activity, defect impact, target, type, and qualifier, respectively, and shows the enhanced performance.

KEYWORDS

Chi Square Score, Machine Learning, Orthogonal Defect Classification, Software Defects

1. INTRODUCTION

Software defect classification is a very crucial task to help the software defect management process. The size and number of software systems are increasing day by day. The defects are being reported by the users of these software systems. Classifying defects into a category helps software developers to assign priorities to the defects, faster resolution, analysis of a defect prone module etc. (Endres1975; Shepherd1993; Wagner2008). Classifying these defects is a time consuming process which is done manually by software developers. In recent years, supervised learning methods have been used to automate the process of defect classification using orthogonal defect classification.

Orthogonal defect classification (ODC) (Chillarege et al.1992 1996) was developed by IBM in the 1990s to provide measuring software process by extracting valuable information from defects. It acts as a bridge between defect modeling and causal analysis. It groups defects based on their

DOI: 10.4018/IJOSSP.300749 *Corresponding Author

impact, trigger, activity, target, type, source, qualifier and age. ODC defect impact specifies a user experience when a defect occurs. Defect impact is further classified into usability, reliability, standard, install-ability, security maintenance etc. In recent years many works have been focused on orthogonal defect classification that tried to categorize the defects based on ODC attributes. ODC has been successfully used by many organizations to improve their process of software development (Butcher 2002;Soylemez and Tarhan 2013;Bridge and Miller 1998;Mays et al. 1990;Lutz and Mikulski 2004; Zheng et al.2006).

In this paper we have evaluated five classifiers namely Naïve Bayes, Support Vector Machine, K Nearest Neighbor, Random Forest and Decision Tree.

In brief, the main contributions of this paper are:

- Defect categorization from unstructured text provided in the description field of defect reports for 4096 defects from three datasets MongoDB, Cassandra and HBase
- Selection of most relevant features using chi square score
- Evaluate the performance of Naïve Bayes, Support Vector Machine, K Nearest Neighbor, Random Forest and Decision Tree dataset wise and with whole data

The rest of this paper is organized in various sections. Section 2 presents the background studies, definitions and related work focuses on software defect categorization using orthogonal defect classification. The next section discusses the dataset. In Section 4, we define the problem and explain our proposed approach; the results are discussed in section 5 in comparison with an existing approach. Section 6 discusses the threats to validity to our work. Last section concludes this paper with conclusion and future scope.

2. BACKGROUND AND RELATED WORK

The definition of defect given by IEEE standard (Board 1993) is "deficiency in a software product when that product does not meet its predefined specifications or requirements, and either needs to be replaced or repaired". A defect report contains various fields like bug id, description, summary, severity, priority, number of comments, URL etc. Defect reports are generally generated through a bug reporting system for each defect. Software managers use a bug reporting system to track the bugs for reporting and assigning. In this way it helps to speed up and improve the process of software project management. In the literature, several works has been found related to classification of software defects (Mays et al.1990; Schneidewind and Hoffmann 1979;Ostrand and Weyuker 1984; Basili and Perrcone 1984;Grady 1992) Various models and the framework have been proposed to understand and improve the classification process. (Wiseman, Y. 2005) wrote about the bugs in operating systems and may have different classification and categories. HP developed a classification taxonomy (Freimut and Ketterer 2005) which categorizes the defect's source on the basis of their type, origin and mode. Some works related to improving the process of software development has been found such as (Kumaresh and Baskaran 2010;Zhi-bo et al. 2011). In recent years a lot of research work has been proposed to automate the process of software defect categorization based on orthogonal defect classification.

(Thung and Jiang 2012) proposed a software defect classification methodology based on support vector machine. The authors classify the defects based on orthogonal defect classification attributes. The proposed methodology categorizes the defect broadly in three categories control and data flow, structural and non-functional. The approach extracts relevant information from the defect reports and classifies manually in the defined three categories. The proposed approach achieves an accuracy of 77.8% with total 500 defect reports were taken for the experimental work. (Zhou et al. 2016) propose a multistage approach, at first stage text mining techniques are applied on the summary field of bug