I. INTRODUCTION

Unit testing is one of white box testing methods that aims to check the implemented source codes of a particular component or function independently (low-level design). In unit testing, code coverage is a metric commonly used to measure the written unit test codes quality [1]. Code coverage has a relationship with the number of potential bugs that exist in the software. The higher the code coverage in an application, the less the number of potential bugs in the application. In addition, code coverage plays an important role in software fault revelation [2]. Regardless of which criteria exist in the code coverage that has a very big influence on the fault revelation, code coverage can be used as a guide to maintain and improve the quality of the software and reduce the possibility of software failures and bugs.

Unit testing requires various resources such as time and expensive costs to achieve high quality software indicated by high code coverage [1, 3, 4]. Since high resources are required, a tool is essential to automate the creation unit test codes. Currently, research related to automation unit test generation (AUTG) only focuses on one language such as Java with the tools EvoSuite and Randoop [5-7]. In addition, the use of machine learning in AUTG is still limited to predict branch coverage that can be achieved from the test data generated by the AUTG tools [6]. The datasets used in the research on this topic mostly were public project repositories on the internet as there is still no dataset that can specifically be used for research on this research area.

The use of preprocessing in machine learning aims to provide some transformations to the dataset before it is being used as input for the machine learning models. In the case of natural language processing, especially in the case study of text classification, the selection of preprocessing or text preprocessing used has an influence on the performance of the resulting models [8, 9]. For example, the preprocessing text used in the Turkish news classification case study will be different from the preprocessing text in the English news classification case study [8, 10]. In the case study of news classification in Turkish, it is recommended to use the following text preprocessing: tokenization, lowercase and stemming [8]. Meanwhile, in the case study of English news classification, tokenization is not required to obtain the best performance [8, 10].

In the case of "source code" as the input of the machine learning models, based on our review of literature, there has been no gold standard or recommendation in terms of appropriate text preprocessing methods. Also, there is no study that discussed in depth about the impacts of different preprocessing methods to "source code" input on machine learning model performances. There are few studies that tried to explore this area, but there are some limitations to these studies. For example, in the research conducted by Reyes [11] and Gilda [12], the text preprocessing methods used were not the same between the two studies and they are simplified and not exhaustive. In addition, these two studies did not explain in depth regarding the preprocessing methods selection justifications and the impacts of the selected methods used on the accuracy of the machine learning model. Furthermore, the datasets used for these two studies were not shared publicly, thus future studies will not be able to replicate or continue extending the study.

This research contribution is twofold. First, the study provides analysis to better understand on the influences of several text preprocessing methods to the performance metrics of machine learning models in the case of a simple code coverage classification (i.e., branch/non-branch). The preprocessing methods used are regular expression (regex), and vectorization including Vector Count, TF-IDF, and Word2Vec. Further, this study aims to investigate the effect of using different preprocessing method and its combination on the machine learning model performance. This analysis and investigation will provide several recommendations for future studies that use "source code" as the input. The second contribution is to generate and provide a public dataset containing "source code" that will be used in this study and future related research.