A Class Balancing Methods Comparison in Software Requirement Classification Using a Support Vector Machine

Fachrul Pralienka Bani Muhamad, Esti Mulyani, Munengsih Sari Bunga
and Achmad Farhan Mushafa
Politeknik Negeri Indramayu, Indonesia

Keywords: Class Balancing, Classification, Feature Extraction, Pure Dataset, Software Requirements.

Abstract: Miss analysis of software requirements in the development stage whether functional or non-functional leads to a significant impact on the quality derivation and cost & time escalation. Especially in agile approaches, such as scrum, some non-functional requirements often go unnoticed, because of a high focus on business functionality that tends to be prioritized. Previous research has been carried out in classifying software requirements, especially non-functional requirements using the PROMISE dataset with the Bag of Words (BoW) feature extraction and the Support Vector Machine (SVM) algorithm. The results obtained from the combination of these methods provide a better accuracy value than the combination of feature extraction and other classification algorithms. However, the software requirement dataset tends to be imbalanced considering that there are several non-functional requirements types, so the data number of each class might differ. In another study, it was stated that the imbalance of datasets could give not optimal classification results, so it is necessary to balance the data. This research proposes class balancing on the dataset after the feature extraction is carried out. The output of the balanced class is used for the classification process. The PURE dataset is used in this research considering that the dataset is open to researchers. After experimenting with the combination of BoW feature extraction, as well as class balancing methods (i.e. SMOTE, Borderline SMOTE, and SVM SMOTE), and classified using the SVM algorithm, it was found that BoW with SVM SMOTE produces the best value average with an accuracy of 78.7%, precision of 80.2%, recall of 78.7%, and F1-Score of 78.9. It has higher results than software classification without a class balancing in enhancement average value accuracy of 0.03%, precision of 0.05%, recall of 0.03%, and F1-Score of 0.04%.

1 INTRODUCTION

Software development requires a comprehensive development process starting from the analysis of Software Requirements and selecting the technology to be used/developed to test the quality of the Software that has been created. According to H. F. Hofmann and F. Lehners (Aminu Umar, 2020) of the many processes in software development, software requirements specifications are very important which will determine the quality of the software itself. Software requirements are divided into two types, namely Functional requirements (FR) and Non-Functional requirements (NFR). FR is the main feature or process that the software will carry out. Meanwhile, NFR contains requirements that focus on the limitations or behavior of the software. NFR is based on the ISO 25010:2011 standard which consist of several categories including Security, Usability, Reliability, Portability, Performance, Compatibility, and Maintainability (Mulyawan et al., 2021).

Unlike the traditional approach, which relies on detailed processes and comprehensive planning, determining software requirements with the Agile approach can still be done by a Product Owner, but given the relatively large and changeable Product Backlog, and reduced levels of human concentration and focus when performing If the work is repetitive then more effort is needed to determine the software requirements. Therefore, we need a model that can provide information regarding the label description of software requirements, both functional and non-functional requirements. Our research is driven by the following research questions (RQ):

RQ1: What is the effect of class balancing methods in software requirements classification using a Support Vector Machine?

Muhamad, F., Mulyani, E., Bunga, M. and Mushafa, A.
A Class Balancing Methods Comparison in Software Requirement Classification Using a Support Vector Machine.
DOI: 10.5220/0011803100003575
In *Proceedings of the 5th International Conference on Applied Science and Technology on Engineering Science (iCAST-ES 2022)*, pages 366-369
ISBN: 978-989-758-619-4; ISSN: 2975-8246

RQ2: Which of the class balancing method give the best result?

RQ3: How far does the class balancing method enhance the classification result without class balancing?

The rest of the paper is organized as follows: Section 2 discusses related work. The section introduces the classification techniques used in this work in detail. Section 4 presents the experiment results and discusses their implications. In Section 5 We conclude this work.

2 RELATED WORK

In this Section we present some papers related to the analysis of requirements, including survey, classification and summarization of requirements.

Research by Ariful Haque, Abdur Rahman, and Saeed Siddik in 2019 entitled "Non-Functional Requirements Classification with Feature Extraction and Machine Learning: An Empirical Study". This study tries to automate the classification of Non-Functional requirements with four Feature Extractions and seven Machine Learning algorithms. This study uses the PROMISE dataset (Haque et al., 2019). The result of the research is that the Support Vector Machine algorithm with Feature Extraction Bag of Word produces higher Precision, Recall, and F1-Score values than other algorithms and Feature Extraction.

Research by Zijad Kurtanovic and Walid Maalej's in 2017 entitled "Automatically Classifying Functional and Non-Functional Needs Using Guided Machine Learning". This study tries to calculate the level of automation of functional and non-functional requirements classification using Machine Learning Support Vector Machine and Lexical Features algorithms (Kurtanovic & Maalej, 2017). The result of this research is that the Support Vector Machine algorithm is successful in classifying Functional and Non-Functional requirements by getting the Precision and Recall values reaching ~92%.

Research by S Tiun, U A Mokhtar, S H Bakar, and S Saad in 2020 entitled "Classification of functional and non-functional requirements in software requirements using Word2vec and fast Text". This study tries to automate the classification of Non-Functional requirements with two combinations of Feature Extraction and four Machine Learning algorithms. This study uses the RE17 dataset (Tiun et al., 2020). The result of this research is that the performance of Word2vec Feature Extraction and fast text classification is not much better than other machine Learning algorithms such as Logistic Regression combined with Bag of Word but better than Support Vector Machine combined with Bag of Word.

3 METHOD

This paper proposed a technique for finding the best combination of balancing class and machine learning approaches to software requirement classification. We preprocessed the dataset by manually labeling each software requirement statement. First, we converted the original NFR dataset to a CSV file. The whole process of this research is divided into the following five steps and depicted in Figure 1.

3.1 Pure Dataset

The data collection in this paper uses a public dataset, and is used in general in research where the classification of software requirements for the dataset in question is the PURE (Public Requirement) dataset.

Table 1: Numbers and Percentages of Manually Labeled User Review Sentences in the Dataset.

Category	Statement	Proportion
Usability	102	12%
Reliability	53	6%
Security	106	12%
Performance	109	12%
Maintainability	35	4%
Portability	41	5%
Functional Requirement	439	50%
Total	**885**	**100%**

3.2 Method Details

There are 5 (five) steps in this study to produce the desired outcome i.e., Preprocessing, Feature Extraction, Class Balancing, Processing, and Evaluation.

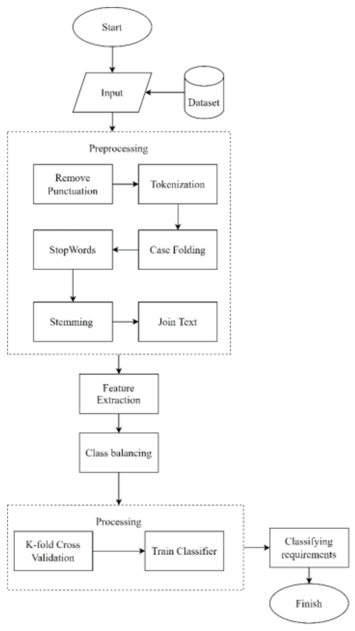

Figure 1: The Software Requirement Classification Process.

a. Step 1: Preprocessing

Before starting the feature extraction process, the word to be used must be cleaned first through the text preprocessing process, preprocessing was used in this study using library Natural Language Toolkit (NLTK), as shown in Figure 1:

- Remove Punctuation
  A process to clean sentences from punctuation marks or perform replacements on the target string based on the specified pattern. In the Remove Punctuation process used is to remove punctuation marks and numbers.

- Case Folding
  A process to changing capital letters to lowercase or regular letters (Rahimi et al., 2020).

- Tokenization
  A process of separating input data into tokens (Binkhonain & Zhao, 2019).

- StopWords
  A process of removing auxiliary verbs, prepositions, pronouns, adverbs, and conjunctions in sentences (Binkhonain & Zhao, 2019) like the, be, to, in, is, etc.

- Stemming
  A process of reducing inflected (or sometimes derived) words to their word stem, base or root form (Binkhonain & Zhao, 2019). For example, the words ‘goes’, ’gone’, and ‘going’ will map to ‘go’.

- Joining Text
  In this process the words that become tokens are combined into 1 sentence.

b. Step 2: Feature Extraction

This step converts the pre-processed re- requirements document into a format that can be understood by the machine learning model (Ramos et al., 2018). In this step, the document is represented as a vector, where the value of this word is weighted through different techniques, such as binary methods. Techniques in vectorization such as the following:

- Bag of Words:
  This vector space model represents unstructured text as a numeric vector, where it establishes the presence of feature words from all the words of an instance. In the process, the software requirements are converted into numerical vectors in such a way that each document is represented by 1 vector (row) (Haque et al., 2019).

c. Step 3: Class Balancing

On such data learning classification methods generally perform poorly because the classifier often learns better than the majority class. The reason for this is that learning classifiers attempt to reduce global quantities such as the error rate, and do not take the data distribution into consideration. As a result, samples from the dominant class are well-classified whereas samples from the minority class tend to be misclassified (Poolsawad et al., 2014). This paper uses one strategy of sampling data. SMOTE generates synthetic examples for the minority class; where SMOTE offers three additional options to generate samples. Those methods focus on samples near the border of the optimal decision function and will generate samples in the opposite direction of the nearest neighbors class.

d. Step 4: Processing

Three classification techniques of class balancing i.e. SMOTE, SVM SMOTE, and BORDERLINE SMOTE has been used in the above step to balance class, which act as the input of machine learning algorithms of training classifiers. The experimentation has been conducted using SVM machine learning algorithms. The various balancing class was applied for requirements classification to compare the performance.

e. Step 5: Evaluation

Each of the classifiers trained in the previous section will output for a given requirement whether it belongs to a category or not. For example, in order to classify requirements according to category performance, the framework will return the list of requirements for which it received a fit with the answer. Also, the other documents will be classified accordingly. The combination of four textual feature extraction methods and SVM machine learning algorithms have been applied in this software requirements classification framework. The textual data has been converted into vector representations to be fed as input in machine learning algorithms.

4 RESULT

In this paper, the evaluation of the machine learning model focuses on the values of the parameters that will be used, including the average score of 2 fold of Accuracy, F1-Score, Precision and Recall.

Table 2: Comparison averaage score all method.

METHOD	AVERAGE SCORE			
	ACCURACY	F1-SCORE	PRECISION	RECALL
SMOTE	0.70	0.70	0.73	0.70
SVM SMOTE	**0.78**	**0.78**	**0.80**	**0.78**
BORDERLINE SMOTE	0.72	0.73	0.75	0.72
Without Class Balancing	0.75	0.74	0.75	0.75

5 CONCLUSION

It can be concluded that the class balancing method can enhance the SVM method in software requirements classification accuracy of 0.03%, precision of 0.05%, recall of 0.03%, and F1-Score 0.04%. Class balancing SVM SMOTE gives the best result among the rest of them.

REFERENCES

Aminu Umar, M. (2020). Automated Requirements Engineering Framework for Agile Development. *ICSEA 2020: The Fifteenth International Conference on Software Engineering Advances*, *c*, 147–150.

Binkhonain, M., & Zhao, L. (2019). A review of machine learning algorithms for identification and classification of non-functional requirements. In *Expert Systems with Applications: X* (Vol. 1). Elsevier Ltd. https://doi.org/10.1016/j.eswax.2019.100001

Haque, A., Rahman, A., & Siddik, S. (2019). Non-Functional Requirements Classification with Feature Extraction and Machine Learning : An Empirical Study. *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), 2019*(Icasert), 1–5.

Kurtanovic, Z., & Maalej, W. (2017). Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning. *Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017*, 490–495. https://doi.org/10.1109/RE.2017.82

Mulyawan, M. D., Kumara, I. N. S., Bagus, I., Swamardika, A., & Saputra, K. O. (2021). *Kualitas Sistem Informasi Berdasarkan ISO / IEC 25010 : 20*(1).

Poolsawad, N., Kambhampati, C., & Cleland, J. G. F. (2014). Balancing class for performance of classification with a clinical dataset. *Lecture Notes in Engineering and Computer Science, 1*(July 2014), 237–242.

Rahimi, N., Eassa, F., & Elrefaei, L. (2020). *SS symmetry An Ensemble Machine Learning Technique for. Ml*, 1–25.

Ramos, F., Costa, A., Perkusich, M., Almeida, H., & Perkusich, A. (2018). A non-functional requirements recommendation system for scrum-based projects. *Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE, 2018-July*(July), 149–154. https://doi.org/10.18293/SEKE2018-107

Tiun, S., Mokhtar, U. A., Bakar, S. H., & Saad, S. (2020). Classification of functional and non-functional requirement in software requirement using Word2vec and fast Text. *Journal of Physics: Conference Series, 1529*(4). https://doi.org/10.1088/1742-6596/1529/4/042077