An Empirical Comparison of Machine Learning Algorithms for Classification of Software Requirements

Law Foong Li[1], Nicholas Chia Jin-An[2]
Department of Computing
School of Computing and Creative Media
UOW Malaysia KDU University College
40150 Glenmarie Shah Alam, Malaysia

Zarinah Mohd Kasirun[3]
Department of Software Engineering
Faculty of Computer Science and Information Technology
University of Malaya
50603 Kuala Lumpur, Malaysia

Chua Yan Piaw[4]
Institute of Educational Leadership
Faculty of Education, University of Malaya
50603 Kuala Lumpur, Malaysia

***Abstract*—Intelligent software engineering has emerged in recent years to address some difficult problems in requirements engineering. Requirements are crucial for software development. Moreover, the classification of natural language user requirements into functional and non-functional requirements is a fundamental challenge as it defines the fulfillment criteria of the users' expected needs and wants. Therefore the research of this article aims to explore and compare random forest algorithm and gradient boosting algorithm to determine the accuracy of functional requirements and non-functional requirements in the process of requirements classification through the conduct of experiments. Random forest and gradient boosting are ensemble algorithms in machine learning that combines the decisions from several base models to improve the prediction performance. Experimental results show that the gradient boosting algorithm yields improved prediction performance when classifying non-functional requirements, in comparison to the random forest algorithm. However, the random forest algorithm is more accurate to classify functional requirements.**

Keywords*—Machine learning; ensemble algorithms; requirements classification; functional requirements; non-functional requirements

I. Introduction

Requirements are introductory building blocks for developing software projects. They are often classified into functional and non-functional requirements [1], [2]. In definition, functional requirements describe the system functionality whilst non-functional requirements describe system properties and constraints. This distinction has determined how requirements are being handled in practice; during elicitation, documentation, and validation [3].

Additionally, requirements are crucial in determining the success of a project; as it establishes a formal agreement between client and software provider working towards the same goal. However, the task of requirements categorization normally expends significant human effort and time when performed manually [4], [5]. The field of software engineering (SE) has witnessed remarkable progress in the past two decades attributable to the advancement of machine learning [6] and natural language processing.

Machine learning in natural language processing has become ever more accessible, leading to more innovations in software engineering. Many techniques and algorithms have been created and adapted into different systems, which has improved performance and overall computational efficiency. Numerous attempts have been made to construct automation for the assistance of extraction and classification of requirements using supervised [7], [8], [9] and semi-supervised learning techniques [10].

This paper aims to explore and compare the machine learning algorithms of random forest algorithm and gradient boosting algorithm. Both algorithms are employed to predict respective labelled data of the functional and non-functional requirements.

This paper is organized as follows: Section II describes the background and Section III discusses the related works. Followed by, Section IV presents the research methodology used. Section V exhibits the results of the study. Section VI presents the findings of the study. Section VII highlights the limitations of the study. Section VIII outlines future work to be undertaken. Finally, Section IX concludes the presented work.

II. Background

There are two primary types of learning schemes in machine learning: supervised learning, where the output has been given a priori labelled or the learner has some prior knowledge of the data; and unsupervised learning, where no prior information is given to the learner regarding the data or the output [11]. The following terms and tools employed in the study are briefly described as follows:

A. Python Pandas

Pandas is a software library written for the Python programming language intended for data manipulation and analysis. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames store and manipulate tabular data in rows of observations and columns of variables [11].

B. Scikit Learn

Scikit Learn is one of the most popular Python toolboxes. It provides a wide selection of supervised and unsupervised learning algorithms. It has advanced functions not commonly offered by other libraries including ensemble methods. Both random forest and gradient boosting are ensemble methods [11]. Both algorithms predict (regression or classification) by combining the outputs from individual trees.

C. Label

Label, also known as target array, defined as the number of categories the machine learning algorithm has to predict. It is generally contained in a NumPy array or Pandas Series [12]. For the purpose of this study, the algorithm of A1 only has two (2) labels which are functional or non-functional requirements. The algorithm of A2 has more than two (2) labels, also known as multiple labels, such as security, performance, usability, etc.

D. n-gram

In the fields of computational linguistics and probability, an n-gram is a contiguous sequence of n-items from a text or speech corpus [11]. The items can be phonemes, syllables, letters, words or base pairs. The n-gram, n=1 is referred to as a "unigram"; n=2 as a "bigram"; and n=3 as a "trigram". For instance:

n-gram 1 = the, phone, rang
n-gram 2 = the phone, phone rang
n-gram 3 = the phone rang

E. Accuracy

Accuracy is one metric for evaluating classification models. This is the measure of the correct number of classifications divided by the total number of classifications [11].

III. Related Works

Machine learning has increasingly gained attention in software engineering. However, there are insufficient research works available in scholarly literature regarding carrying out an accurate comparison of machine learning algorithms for classification of software requirements, which could be used as a reference to conduct similar works in the future. These studies [7], [9], [10] concentrate on the classification of software requirements by tackling the machine learning approach through different models. Furthermore none of these studies will use the ensemble approach.

Kurtanović and W. Maalej [7] classified requirements as functional and non-functional requirements using support vector machine, abbreviated as SVM algorithm. Most present studies focuses on the classification of either functional or non-functional requirements. For example, Slankas and Williams [9] evaluated multiple classifiers to identify non-functional requirements and found the support vector machine had the highest effectiveness.

The approach proposed by Casamayor et al. [10] for the non-functional requirements identification is focused on semi-supervised text classification. The accuracy rates are above 70% for this proposed approach, significantly higher than the results obtained through the supervised method of using the standard collection of documents.

Instead, this study provides an implication for deciding accurate algorithm to categorize functional and non-functional requirements individually by attempting the ensemble approach which makes allowance for better predictions compared to a single model in order to close the research gap as mentioned earlier in this section.

IV. Research Methodology

A mixture of natural language processing algorithms and machine learning algorithms was used in the study. The machine learning algorithms were used to predict functional and/or non-functional labelled data, as well as labels such as security, usability, efficiency, etc. Alternatively, the natural language processing algorithms were used to generate a sentence(s) from the user requirements.

The main emphasis of this article is the machine learning algorithms. Therefore, no natural language processing algorithms will be discussed. In this article, algorithm A1 represents the algorithm that classifies the functional requirements and algorithm A2 represents the algorithm that classifies the non-functional requirements. Experiments are conducted to determine the machine learning algorithms. The random forest and gradient boosting algorithms was used to predict the respective labelled data of functional and non-functional requirements.

A. Data Preparation

The raw data was formatted and vectorised before passed into the random forest algorithm and the gradient boosting algorithm to perform model fitting and prediction. Then, the data frames were sorted with the relevant data to be used by the natural language processing algorithms at the next stage.

Once the file was read, it began to format the input data into a standard format. A few steps were taken to convert the raw data into a standard format as follows:

1) Remove all punctuations from the text.
2) Convert the text into lowercase.
3) Add a full-sentence column into the data frame.
4) Remove stop words.

The following vectorization steps were conducted to prepare the data for use by the algorithm (after the raw data has been formatted):

1) Create a new data frame that has only the required columns for the algorithm.
2) Split the training and test data into x and y coordinates respectively.
3) Vectorise the data to be used by the algorithm. This means that the words would be converted into unique

identifiers for the algorithm since the algorithm only accepts numerical data. For example, yes => 0 and no => 1.

4) Concatenate any features into the vectorised data.

B. Experimental Instruments

Python Pandas is well suited for different kinds of data. Furthermore, the DataFrame of Python Pandas can be created by loading datasets from existing external storage such as a SQL database, CSV files, list of dictionary, etc. Thus, the Python Pandas DataFrame is designated for the data manipulation in this study.

On the other hand, Scikit Learn was also chosen as a tool in this study because it is built on top of common data and Python Math Libraries. The design makes for ease of integration, whereby numpy arrays and pandas data frames can pass directly to the machine language (ML) algorithms of Scikit. In addition, it features numerous classification, regression, and clustering algorithms including support vector machines, random forest, gradient boosting, etc. Hence, two (2) ML algorithms in Python, which are random forest and gradient boosting, have been chosen for this study to classify functional requirements and non-functional requirements in turn.

C. Experimental Procedure

This section outlines the experimental procedure that classifies software requirements into two (2) different types, which are functional and non-functional requirements.

Fig. 1 illustrates the model fitting outline before model prediction. First, the system will read the selected file and check for errors in the file type chosen. When errors are found, it will prompt the user with an error message and allow for file re-selection. Once all errors are expunged, the system will format and prepare the data for use by algorithm A1 and A2. Following, the system will perform the model fitting, followed by the model prediction. Finally, the model predictions of algorithm A1 and algorithm A2 will be saved. The system will then return to the user, the trained or fitted model of algorithm A1 and algorithm A2 to make the prediction.

The purpose of conducting experiments on both algorithms is to determine which algorithm; random forest or gradient boosting is more accurate for algorithm A1 and algorithm A2. As mentioned at the beginning of this section, algorithm A1 represents the algorithm that classifies functional requirements and algorithm A2 represents the algorithm that classifies non-functional requirements.

There will be two (2) experiments in this study. Each experiment will have different sets of data. For example, a file that comprises a mixture of functional requirements and non-functional requirements will be used in conducting the first experiment to determine which algorithm, random forest or gradient boosting, is more accurate for algorithm A1. Objectively, the purpose of conducting the second experiment is to determine which algorithm, random forest or gradient boosting, is more accurate for algorithm A2. Hence, a file that consists of only non-functional requirements with their sub-category(s) will be employed to realize the purpose of the second experiment.

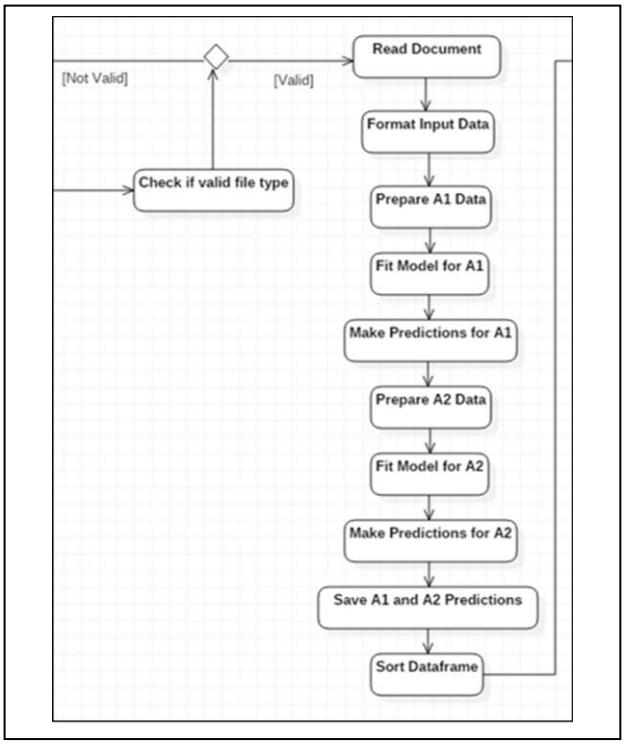

Fig. 1. Experimental Procedure of this Study.

In practice, a two (2) test per model configuration change is conducted in each experiment. The two (2) tests are as below:

1) The first test will be conducted five (5) times for each n-gram range and the averages are collected to find out which of the n-gram ranges are more accurate.

2) The second test will be conducted ten (10) times for each n-gram range and the averages are collected to reduce any variances that could come from the data splitting processes.

Nonetheless, there might be the circumstance of a new setting that leads to retesting the n-gram range to determine which n-gram range is most accurate. In such circumstance, more tests, to be conducted repetitively, was required. Such circumstance could also be a test to find a new setting, such as the number of estimators, also known as the number of decision trees built by the algorithm, or the depth of the decision trees. So, whenever a change is made in the number or depth of estimators, for example, the n-gram test needs to be conducted again to see the implications on the n-gram range.

V. Results of Study

The results of algorithm A1 that applied random forest and gradient boosting technique with two parameters which are the number of trees and maximum depth of the trees are shown respectively in Table I and Table II. Separately, Table III and Table IV illustrates the results of algorithm A2 that applied the same technique and parameters as presented in Table I and Table II. There are some null values of maximum depth of the trees appears in Table I and Table III which means no limits or infinite depth of the trees.

The final results of algorithm A1 and A2 are revealed in Table V and Table VI. Table V shows that the random forest technique has an advantage in the average accuracy compared with the gradient boosting algorithm. The random forest algorithm achieved a higher average accuracy which is 0.826 in comparison with the gradient boosting algorithm which is 0.789. Thus, the measured values deviated by the range of 0.037. It can be concluded that the random forest is more accurate for algorithm A1, which is responsible to classify functional requirements.

In contrast, Table VI illustrates that the gradient boosting algorithm is more accurate for algorithm A2. The gradient boosting algorithm obtained a higher average accuracy which is 0.591 compared to the random forest algorithm which only has 0.582 of the average accuracy.

It is noteworthy to indicate that the number of labels of raw data has influenced the average accuracy of the algorithm significantly. In this study, the random forest and gradient boosting algorithms was used in the first experiment to predict functional and non-functional labelled data. Meanwhile, the random forest and gradient boosting algorithms was used in the second experiment to predict non-functional labelled data such as security, usability, efficiency, and so on. The efforts to predict multiple labels of raw data in the second experiment is tougher than that of the binary labels in the first experiment. The machine learning algorithms had to predict binary labels of raw data such as functional requirements and non-functional requirements in the first experiment, which would give 50% accuracy through blind guessing. In contrast, multiple labels of raw data from different categories of non-functional requirements, for instance, ten (10) labels, needed to be predicted in the second experiment which gave 10% accuracy through the prediction process. This is because the algorithm was required to choose from the available labels rather than to perform blind guessing. It also sets the baseline for comparing the accuracy of the algorithms.

In summary, the results of this study shows that the random forest algorithm is more accurate for algorithm A1 whereas the gradient boosting algorithm is more suited for algorithm A2 due to accuracy.

TABLE. I. RESULT OF ALGORITHM A1 USING RANDOM FOREST WITH NUMBER OF TREES AND MAXIMUM DEPTH OF THE TREES PARAMETERS SETTING

Number of Trees	Maximum Depth of the Trees	Precision	Recall	Accuracy	Predict Time
50	40	0.892	1	0.915	0.106
200		0.892	1	0.915	0.103
100	40	0.900	1	0.894	0.104
50	30	0.868	1	0.894	0.108
100	30	0.868	1	0.894	0.104
100		0.889	0.970	0.894	0.104
200	40	0.868	1	0.894	0.104
50	30	0.878	1	0.894	0.104
100		0.878	1	0.894	0.104
200		0.919	0.944	0.894	0.104

TABLE. II. RESULT OF ALGORITHM A1 USING GRADIENT BOOSTING WITH NUMBER OF TREES AND MAXIMUM DEPTH OF THE TREES PARAMETERS SETTING

Number of Trees	Maximum Depth of the Trees	Precision	Recall	Accuracy	Predict Time
150	11	0.914	0.970	0.915	0.001
150	3	0.889	0.970	0.894	0.001
150	11	0.912	0.939	0.894	0.001
300	3	0.889	0.970	0.894	0.001
300	11	0.912	0.939	0.894	0.001
450	11	0.889	0.970	0.894	0.001
450	3	0.848	1.000	0.894	0.001
450	3	0.935	0.879	0.872	0.001
150	7	0.886	0.939	0.872	0.001
300	7	0.910	0.909	0.872	0.001

TABLE. III. RESULT OF ALGORITHM A2 USING RANDOM FOREST WITH NUMBER OF TREES AND MAXIMUM DEPTH OF THE TREES PARAMETERS SETTING

Number of Trees	Maximum Depth of the Trees	Precision	Recall	Accuracy	Predict Time
50	20	0.586	0.586	0.586	0.125
50	40	0.621	0.621	0.621	0.109
50		0.621	0.621	0.621	0.123
100	20	0.586	0.586	0.586	0.110
100	40	0.621	0.621	0.621	0.121
100		0.655	0.655	0.655	0.120
200	20	0.621	0.621	0.621	0.109
200	40	0.655	0.655	0.655	0.110
200		0.586	0.586	0.586	0.109
50	20	0.621	0.621	0.621	0.109

TABLE. IV. RESULT OF ALGORITHM A2 USING GRADIENT BOOSTING WITH NUMBER OF TREES AND MAXIMUM DEPTH OF THE TREES PARAMETERS SETTING

Number of Trees	Maximum Depth of the Trees	Precision	Recall	Accuracy	Predict Time
150	11	0.586	0.586	0.586	0.001
300	11	0.621	0.621	0.621	0.001
450	11	0.621	0.621	0.621	0.001
600	11	0.517	0.517	0.517	0.001
750	11	0.586	0.586	0.586	0.002
900	11	0.655	0.655	0.655	0.002
1050	11	0.552	0.552	0.552	0.002
150	11	0.517	0.517	0.517	0.001
300	11	0.586	0.586	0.586	0.001
450	11	0.586	0.586	0.586	0.001

TABLE. V. FINAL RESULTS OF ALGORITHM A1

Setting	Average Precision	Average Recall	Average Accuracy	Predict Time
Random Forest	0.802	0.965	0.826	0.111
Gradient Boosting	0.809	0.882	0.789	0.658

TABLE. VI. FINAL RESULTS OF ALGORITHM A2

Setting	Average Precision	Average Recall	Average Accuracy	Predict Time
Random Forest	0.582	0.582	0.582	0.105
Gradient Boosting	0.591	0.591	0.591	0.131

VI. DISCUSSION

There are several tuning parameters important for random forests and gradient boosting algorithm, however, only two parameters which are the number of trees and the tree depth were chosen in this discussion. Generally, a higher number of trees increased the performance and made the predictions more stable which could result in better accuracy. In comparison, more trees also meant more computational cost and after a certain number of trees, the improvement was negligible. The results in Table I show that there was no significant improvement on the accuracy; for example, the first and second record obtained the same accuracy rate which was 0.915, but both had different number of trees 50 and 200 for the first and second record respectively. It is concluded that as the number of trees grows, it does not always mean the performance of the forest is significantly better than previous forests which had fewer trees. The results of Table I indicates that the addition of trees is insignificant. This result is aligned with the previous studies finding [14] revealing that the smallest number of trees is sufficient to obtain the same level of accuracy. Barman et al. [15] also discovered that there is no significant difference between using a number of trees, and larger number of trees in a forest will not significantly improve the performance but in contrary, it will increase its computational cost.

Table II shows that the first record gained the highest accuracy which was 0.915 with 150 number of trees and 11 depth of the trees. The results in Table II yet again shows that larger number of trees has no significant improvement on the accuracy.

Table III displayed that the eighth record listed has the highest accuracy value of 0.6555 in contrast to the seventh and ninth record listed which possesses the same number of trees, but have differing depth of trees which are 20, 40 and infinite depth of trees respectively. The results indicate that depth of the trees has a significant effect on the accuracy. As the depth increases, the stability of prediction will decrease as each model tends to cause overfitting [16]. The depth of the tree meaning length of tree. Larger tree helps to convey more information whereas smaller tree gave less precise information. Hence, there needed to be a balanced ratio within the depth of trees to gain a better performance.

Alternatively, the findings in Table IV indicate the number of trees will determine the accuracy of prediction when the depth of the trees value is constant. The sixth record listed in Table IV gained the highest accuracy which was 0.655.

The parameters in random forest and gradient boosting are either to increase the predictive power of the model or to make it easier to train the model. Important parameters to fine tune would be the number of trees, the depth of trees and the number of features used for a split. However, the number of trees and the depth of trees were selected to perform the fine tuning the models of this study. Optimistically this article has given essential understanding to begin using the random forest and gradient boosting on projects.

VII. LIMITATION OF STUDY

As with all research, there were a few limitations to this study that must be acknowledged. Firstly, the system is limited by the amount of data due to the issue of confidentiality; similar to that encountered in similar research [13]. As more quality data is collected, the accuracy of algorithm A1 and algorithm A2 will increase. Secondly, the system has a limitation on the format of data read, as it requires data to be in a specific format to function properly. It is also limited by its inability to determine how many separate requirements there are in a sentence, as well as its inability to read from .docx and .txt files.

VIII. FUTURE WORK

With the limitations of study stated in Section VII. For future work, the following aspects are identified:

1) Collect more data to be used in experiments.

2) Generalize the reading format to allow the reading of data in a more general format.

3) Implement a function to determine how many different requirements exist in a sentence.

IX. CONCLUSION

This article explores and compares the random forest algorithm and the gradient boosting algorithm to discover which is more accurate to classify functional requirements and non-functional requirements, by conducting experiments. Among the investigated machine learning algorithms, the results of this study have shown empirically that the gradient boosting algorithm yields better prediction performance in terms of accuracy when sorting non-functional requirements, in comparison to the random forest algorithm.

For future work, more machine learning algorithms will be investigated by engaging the ensemble strategy in order to improve the overall classification accuracy.

ACKNOWLEDGMENT

This project has received funding from the KDU Research Grant under grant no. KDURG/2017/010.

REFERENCES

[1] G. Kotonya, & I. Sommerville, Requirements engineering with viewpoints, Software Engineering Journal, 11(1), 5-18, 1996.

[2] R. Y. Lee, Requirements Elicitation Software Engineering: A Hands-On Approach (pp.81-102). Paris: Atlantis Press, 2013.

[3] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc. , “The detection and classification of non- functional requirements with application to early

aspects". In 14th IEEE International Requirements Engineering Conference (RE'06). IEEE, 2006. pp. 38-49.

[4] J. Cleland-Huang, R. Settimi, X. Zou, & P. Solc, Automated classification of non-functional requirements. Requirements Engineering, 12(2), 103-120, 2007.

[5] A. Rashwan, O. Ormandjieva, & R. Witte, "Ontology-based classification of non-functional requirements in software specifications: a new corpus and svm-based classifier". In 2013 IEEE37th Annual Computer Software and Applications Conference. IEEE, July 2013. pp. 381-386.

[6] D. Zhang, & J. J. Tsai, Machine learning and software engineering, Software Quality Journal, 11(2), 87-119, 2003.

[7] Z. Kurtanović and W. Maalej, "Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning," In 2017 IEEE 25th International Requirements Engineering Conference (RE). IEEE, September 2017, pp. 490-495.

[8] M. Lu, & P. Liang, Automatic classification of non-functional requirements from augmented app user reviews. In Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering. ACM, June 2017. pp. 344-353.

[9] J. Slankas, & L. Williams, Automated extraction of non-functional requirements in available documentation. In 2013 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE). IEEE, May 2013, pp. 9-16.

[10] A. Casamayor, D. Godoy, & M. Campo, Identification of non-functional requirements in textual specifications: A semi-supervised learning approach, Information and Software Technology, 52(4), 436-445, 2010.

[11] I. Laura and S. Santi, Introduction to Data Science A Python Approach to Concepts, Techniques and Applications. Springer, Switzerland, 2017.

[12] V.Jake, Python Data Science Handbook. O'Reilly Media, Inc, Sebastopol, United States, 2016.

[13] B. Athuraliya and C. Farook, ""Revyew" Hotel Maintenance Issue Classifier and Analyzer using Machine Learning and Natural Language Processing," In 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON). IEEE, 2018, pp. 274-280.

[14] P. Latinne, O. Debeir, & C. Decaestecker, Limiting the number of trees in random forests. In International workshop on multiple classifier systems. Springer, Berlin, Heidelberg, July 2001. pp. 178-187.

[15] T. M. Oshiro, P. S. Perez, & J. A. Baranauskas, How many trees in a random forest? In International workshop on machine learning and data mining in pattern recognition .Springer, Berlin, Heidelberg, 2012, July. pp. 154-168.

[16] C. B. Liu, B. P. Chamberlain, D. A. Little, & Â. Cardoso, Generalising random forest parameter optimisation to include stability and cost. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, Cham. 2017, September. pp. 102-113.