An Empirical Study on Discovering Software Bugs Using Machine Learning Techniques

G. Ramesh, K. Shyam Sunder Reddy, Gandikota Ramu, Y. C. A. Padmanabha Reddy, and J. Somasekar

Abstract Bug is a defect in software which needs to be identified early so as to avoid unnecessary burden caused by it later. Bug discovery from software modules has been around. However, of late, machine learning (ML) became a useful and appropriate solution to many real-world problems. In this context, usage of machine learning has become an important step forward in improving state of the art in bug detection. It is an artificial intelligence-based (AI) approach that makes it more effective due to the bulk of software modules. Many existing methods strived to incorporate ML for bug discovery. However, there is need for improvement with appropriate methodology. In this paper, we proposed a methodology that exploits two ML techniques known as decision tree (DT) and random forest (RF) for efficient means of discovering bugs from software modules. An empirical study is made using Python data science platform. Experimental results showed that RF performs better than DT in terms of accuracy of bug prediction.

Keywords Machine learning · Software bug discovery decision tree · Random forest

G. Ramesh (✉)
Department of CSE, GRIET, Bachupally, Hyderabad, Telangana, India
e-mail: ramesh680@gmail.com

K. S. S. Reddy
Department of Information Technology, Vasavi College of Engineering, Hyderabad, India

G. Ramu
Department of Computer Science and Engineering, Institute of Aeronautical Engineering, Dundigal, Hyderabad 500 043, India

Y. C. A. P. Reddy
Department of CSE, B V Raju Institute of Technology, Narsapur, Telangana, India

J. Somasekar
Department of CSE, Gopalan College of Engineering and Management, Bangalore, India

R. Buyya et al. (eds.), *Computational Intelligence and Data Analytics*,
Lecture Notes on Data Engineering and Communications Technologies 142,
https://doi.org/10.1007/978-981-19-3391-2_14

1 Introduction

Early detection of software bugs is to be given paramount importance in the real-world application development. It could lead to better performance in terms of saving time, effort and money. If any bug is not discovered early, it is carried forward to next phase in the life cycle of software. It becomes difficult later to fix the problem as it is expensive and needs more time and effort. Of late, machine learning domain has paved way to solve many complex real-world problems. It is widely used in different areas and applications. Software development industry is also exploring the benefits of machine learning. Discovery of bugs from software and related documents is an important part in software engineering. From the literature, it is known that different methods are used for bug detection in software related data sets. Most of the solutions are based on active machine learning techniques. In this paper, we proposed a methodology that exploits decision tree (DT) and random forest (RF) to have bug prediction models. We evaluated the models to know the better performing model. Our contributions in this paper are as followed.

1. A methodology is proposed to have DT and RF models to detect bugs in software.
2. DT and RF are used with specific approach to detect bugs.
3. An empirical study is made using Python data science platform and evaluation showed that RF performs better than DT.

The remainder of the paper is structured as follows. Section 2 reviews literature. Section 3 presents the proposed system. Section 4 presents experimental results, and Sect. 5 concludes the work.

2 Related Work

This section reviews literature on bug discovery using machine learning approaches. Ferreira et al. [1] employed machine learning to detect faults in wireless mesh networks associated with solar power distribution system. Tan et al. [2] used C5.0 and random forest (RF) for prediction of network faults. Duenas et al. [3] focussed on network failure prediction online by using event stream processing. Tran et al. [4] used RF to discover software bugs in the bug reports. Tran et al. [5] on the other hand explored different data analytics techniques used for detection of faults. Armbrust et al. [6] investigated on the dynamics of faults in the context of cloud computing. Hammouri et al. [7] explored machine learning methods to find bugs associated with software development. Zhang et al. [8] defined an approach known as KSAP to have bug report assignment in an efficient manner. Towards this end, they employed KNN search-based methodology. Sabor et al. [9] investigated on the automatic prediction of bugs and their severity levels with data pertaining to stack traces. Ramesh et al. [10] define an approach for technique for identifying the code smells. Pooja et al. [11] explored different techniques for analysing the software applications.

Gupta et al. [12] defined a novel XGBoost-based model to predict software bugs with supervised learning approach. Riza et al. [13] proposed an algorithm known as Knuth–Morris–Pratt to identify genomic repetitions. Sheneamer [14] focussed on finding code clones that are a kind of bugs in software development process. They used multiple similarity-based features in order to achieve this. From the literature, it is known that different methods are used for bug detection in software-related data sets. Most of the solutions are based on active machine learning techniques. In this paper, we proposed a methodology that exploits decision tree (DT) and random forest (RF) to have bug prediction models. We evaluated the models to know the better performing model.

3 Methodology

The proposed methodology includes the process of both DT and RF in prediction of bugs. DT uses the training data set in order to grow a tree based on features. Then, it finds the best split for every feature available. Then, it finds the required node that leads to best split. The node is split using the identified best fit. It is based on the stopping rules. After an iterative process for all nodes, a single decision tree is formed. The splitting rule is based on Eq. (1) and Eq. (2).

$$H(S) = -\sum_{x \in X} P(x) \log P(x) \tag{1}$$

$$IG(A, S) = H(S) - \sum_{t \in T} P(t) H(t) \tag{2}$$

The attribute that has highest information gain is considered to perform splitting. In case of RF, multiple decision trees are generated, and ensemble model is used to arrive at final predictions. The given data set is divided into number of subsets. A training data set is used to construct a tree. Then, the DT-based approach is followed for each tree. The tree is used to evaluate testing data set. After the iterative approach, a RF of tree is generated.

4 Experimental Results

Experimental results are presented in terms of cross-validation score, accuracy and consumption time.

As presented in Table 1, the cross-validation score is provided in presence of pre-processing and absence of it. Pre-processing is used to improve quality of training by filling missing values.

As shown in Fig. 1, the cross-validation score has its influence on both the number of bug reports and the presence of pre-processing.

Table 1 Shows the cross-validation score

	Cross-validation score					
	5	10	15	20	25	30
With pre-processing	0.61	0.6	0.6	0.6	0.65	0.63
Without pre-processing	0.48	0.49	0.5	0.5	0.48	0.5

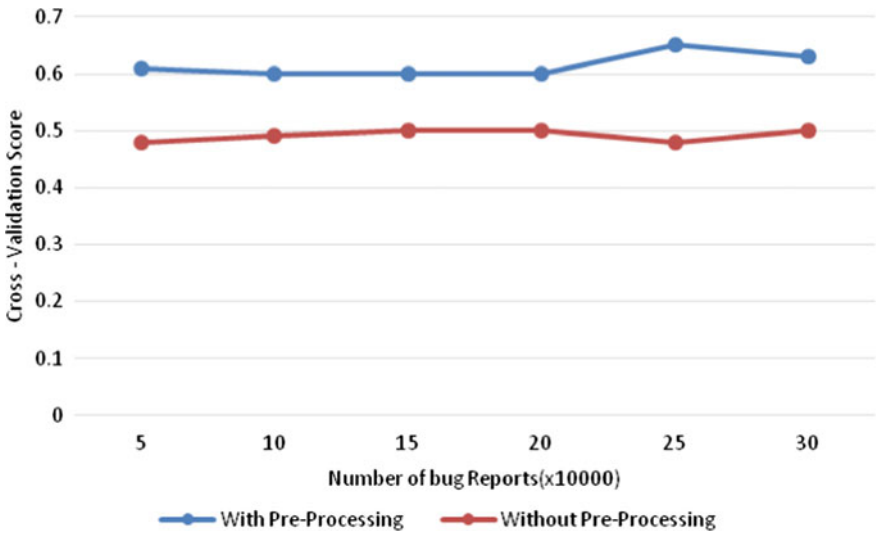

Fig. 1 Shows cross validation score against number of bug reports

As presented in Table 2, different number of bug reports (×10,000) have influence on the prediction models in terms of accuracy.

As shown in Fig. 2, the accuracy score is influenced by the number of bug reports. Random forest showed higher accuracy score.

As presented in Table 3, different number of bug reports (×10,000) have influence on the prediction models in terms of priority accuracy.

As shown in Fig. 3, the priority accuracy score is influenced by the number of bug reports. Random forest showed higher accuracy score.

As presented in Table 4, different number of bug reports (×10,000) have influence on the prediction models in terms of consumption time.

Table 2 shows accuracy of the models

	Accuracy score					
	5	10	15	20	25	30
Random forest	0.82	0.8	0.76	0.75	0.78	0.75
Decision tree	0.68	0.68	0.65	0.65	0.68	0.62

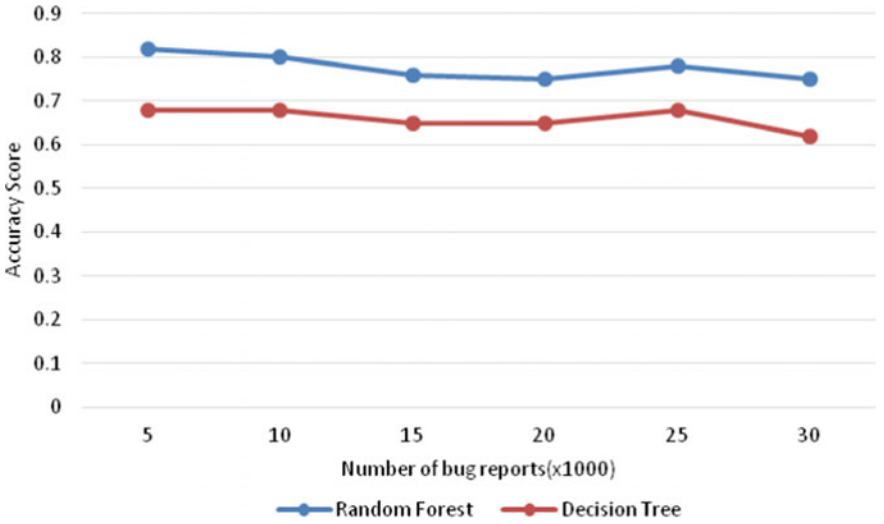

Fig. 2 shows accuracy score against number of bug reports

Table 3 Shows priority accuracy of the models

	Priority accuracy score					
	5	10	15	20	25	30
Decision tree	0.69	0.69	0.69	0.69	0.65	0.62
Random forest	0.72	0.72	0.73	0.75	0.75	0.74

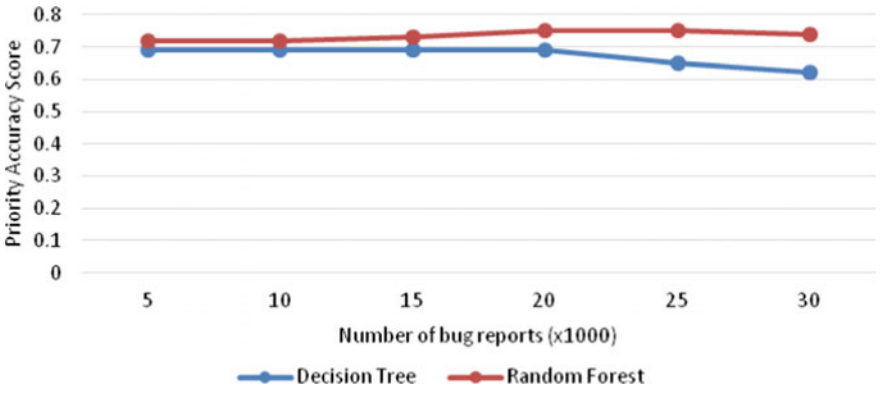

Fig. 3 Shows priority accuracy score against number of bug reports

Table 4 Shows consumption time of the models

	Consumption time (S)					
	5	10	15	20	25	30
Random forest	1	2	3	4	5	7
Decision tree	5	7	12	20	25	35

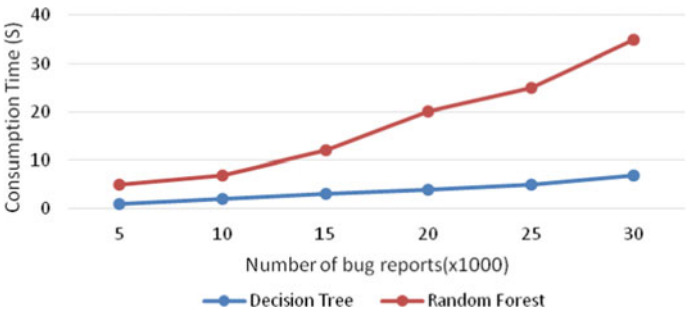

Fig. 4 shows consumption time against number of bug reports

As shown in Fig. 4, the consumption time is influenced by the number of bug reports. Random forest showed higher consumption time.

5 Conclusion and Future Work

In this paper, we proposed a methodology that exploits two ML techniques known as decision tree (DT) and random forest (RF) for efficient means of discovering bugs from software modules. The methodology describes how the two techniques are able to find bugs. DT is the approach which is based on forming a decision tree to have predictions. RF on the other hand uses multiple DTs and makes an ensemble of them in order to have better prediction of bugs. An empirical study is made using Python data science platform. Experimental results showed that RF performs better than DT in terms of accuracy of bug prediction. The methodology explored in this paper has several limitations. First, it has limitations in terms of number of ML methods. Second, it needs further improvement in terms of processing of data prior to using algorithms. In future, we overcome these limitations with further improvements in methodology.

References

1. Ferreira VC, Carrano RC, Silva JO, Albuquerque CVN, Muchaluat-Saade DC, Passos DG (2017) Fault detection and diagnosis for solar-powered wireless mesh networks using machine learning. In: Proceedings of IFIP/IEEE symposium on integrated network and service management (IM'17), pp 456–62
2. Tan JS, Ho CK, Lim AH, Ramly MR (2018) Predicting network faults using Random forest and C5.0. Int J Eng Technol 7(2.14):93–6
3. Duenas JC, Navarro JM, Parada HA, Andion J, Cuadrado F (2018) Applying event stream processing to network online failure prediction. Commun Mag 56(1):166–170
4. Tran HM, Nguyen SV, Ha SVU, Le TQ (2018) An analysis of software bug reports using Random forest. In: Proceedings of 5th international conference on future data and security engineering (FDSE'18). Springer, pp 1–13
5. Tran HM, Nguyen SV, Le ST, Vu QT (2017) Applying data analytic techniques for fault detection. Trans Large Scale Data Knowl Cent Syst (TLDKS) 30–46
6. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M (2010) A view of cloud computing. ACM Commun 53(4):50–58
7. Hammouri A, Hammad M, Alnabhan M, Alsarayrah F (2018) Software bug prediction using machine learning approach. Int J Adv Comput Sci Appl 9. https://doi.org/10.14569/IJACSA.2018.090212
8. Zhang W, Wang S, Wang Q (2015) KSAP: an approach to bug report assignment using KNN search and heterogeneous proximity. J Inf Softw Technol 70:68–84
9. Sabor KK, Hamdaqa M, Hamou-Lhadj A (2019) Automatic prediction of the severity of bugs using stack traces and categorical features. Elsevier J Inf Softw Technol
10. Ramesh G, Mallikarjuna Rao C (2018) Code-smells identification by using PSO approach. Int J Recent Technol Eng (IJRTE) 7(4). ISSN: 2277-3878
11. Pooja ASSVL, Sridhar M, Ramesh G (2021) Application and analysis of phishing website detection in machine learning and neural networks. In: Luhach AK, Jat DS, Bin Ghazali KH, Gao XZ, Lingras P (eds) Advanced informatics for computing research. ICAICR 2020. Communications in computer and information science, vol 1394. Springer, Singapore
12. Gupta A, Sharma S, Goyal S, Rashid M (2020) Novel XGBoost tuned machine learning model for software bug prediction. 2020 international conference on intelligent engineering and management (ICIEM), pp 376–380
13. Riza LS, Rachmat AB, Munir TH, Nazir S (2019) Genomic repeat detection using the Knuth-Morris-Pratt algorithm on R high-performance-computing package. Int J Adv Soft Comput Appl 11(1):94–111
14. Sheneamer AM (2021) Multiple similarity-based features blending for detecting code clones using consensus-driven classification. Expert Syst Appl 183