Applying Machine Learning Techniques to Predict the Maintainability of Open-Source Software

Madhwaraj Kango Gopal, Amirthavalli M.

Abstract – Software maintainability is a vital quality aspect as per ISO standards. This has been a concern since decades and even today, it is of top priority. At present, majority of the software applications, particularly open source software are being developed using Object-Oriented methodologies. Researchers in the earlier past have used statistical techniques on metric data extracted from software to evaluate maintainability. Recently, machine learning models and algorithms are also being used in a majority of research works to predict maintainability. In this research, we performed an empirical case study on an open source software jfreechart by applying machine learning algorithms. The objective was to study the relationships between certain metrics and maintainability.

Index Terms— Machine Learning, Maintainability, Open Source Software, Object-Oriented, Design Metrics

I. INTRODUCTION

The ISO/IEC 25010:2011 breaks down software quality into eight characteristics. Maintainability is listed as the number one priority attribute due to its importance since decades. Today, many open source software are being created using the Object-Oriented (OO) paradigm. It becomes essential to evaluate the quality of such software especially maintainability. Design metrics are those values that are obtained from software applications at design time. These metrics can be used as indicators to evaluate software maintainability. Machine learning is a concept that takes its roots from artificial intelligence which gives power to software applications to improve themselves through experience. It targets on building predictive models and applications therefore get trained for themselves. Different types of machine learning have gained profound influence like supervised learning, unsupervised learning, reinforcement machine learning etc… Machine learning models represent a class of application software that learn from a given set of data and arrives at predictions on the new dataset based in its learning experience. Machine learning models are trained with an already existing dataset to make predictions on new datasets.

Manuscript published on 30 July 2019.
***** Correspondence Author (s)

Dr. Madhwaraj Kango Gopal, Professor in the Department of Computer Applications, New Horizon College of Engineering.
Ms. Amirthavalli, PhD scholar in the Department of Information Technology at SSN College of Engineering, Kalavakkam.

Recently, there is a growth in the application of machine learning models and algorithms on a variety of diverse applications in software quality [21,22]. Much earlier to this, researchers were using metrics extracted from several software applications and building predictive models. Software metrics have also gained importance in predicting maintainability [9, 23]. Some research works have been done by using certain design metrics from different suites. There are also works which use an entire metric suite and understand the influence of a suite in its entirety on software maintainability [19, 20]. Some of the popular OO metric suites are the Martin suite, CK suite, and the MOOD suite. In this research study, an entire metric suite i.e. Martin suite of metrics was extracted from a popular open source charting application jfreechart using the JDepend tool [12]. Machine learning algorithms were used on the extracted dataset. This research demonstrates the relationships between design metrics proposed by Martin [10] and maintainability.

The research work has been organized as follows. The related work has been presented in Section II. The case study design is explained in Section III. The machine learning techniques that were used with results and discussion are highlighted in Section IV. The important observations and conclusions are presented in Section V. The future directions are explored in Section VI and Section VII discusses the threats to validity.

Figure 1 shows the process diagram of the case study performed in this research work.

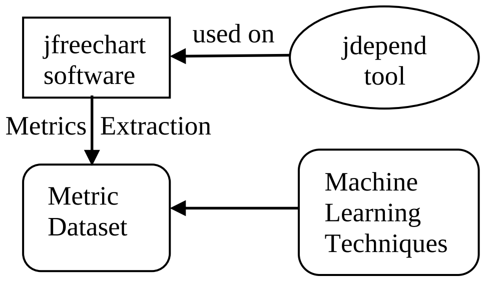

Figure 1 : The Process Diagram

II. RELATED WORK

Misra [9] identified that metrics can be used as important parameters towards predicting maintainability. MI metric [3] has received considerable attention in the past and has been validated in several research works. This metric can be used to check the maintainability of traditional code and was later adapted to OO too with certain limitations. Welker [1] validated the MI and found evidence of the practical use of MI in determining the maintainability of a software application. He further suggested the use of MI to measure the quality of any traditional source code.Recently, predictive models are being built using design metrics and this has gained importance due to the benefits it offers. This will also allow designers to make the necessary design or coding changes in their software applications and raise the maintainability levels in their software applications. Yuming et al. [5] evaluated the relationships between a few metrics and software maintainability. They found certain size metrics influencing software maintainability. Thwin et al. [4] have applied the neural network concepts in estimating software quality using OO metrics. Zhou et al. [6] proposed a modelling technique to build maintainability prediction models using metrics from some software applications. Aggarwal et al. [7] proposed an ANN model for maintainability prediction using OO metrics. Muthanna et al. [14] investigated the use of metrics to evaluate the maintainability of software by applying statistical concepts. Elish et al. [17] have performed a comparative study between three metric suites for prediction of faults in OO systems. They found evidence of prediction models constructed using the Martin suite to be more accurate when compared to models built using the CK and MOOD suites. Madhwaraj [19] performed a study to compare the performance of two OO suites and found that prediction models obtained from the Martin suite were performing better than the prediction models obtained from the Chidamber and Kemerer suite. Madhwaraj [20] has also performed a case study on jfreechart software and proposed a predictive model for maintainability. None of these research works have used machine learning algorithms and techniques. Therefore, an attempt was done to investigate the jfreechart by applying machine learning algorithms.

III. CASE STUDY

This section gives information on the jfreechart software and also describes the dependent and independent variables that were investigated.

III.1 About jfreechart

jfreechart is a charting application that is very popular among the software community. The objective of the selection was to investigate an open source software application which has evolved with many versions and written in java language.

III.2 Dependent Variable and Independent Variables

This research study evaluates the maintainability of a software application with the MI metric. MI is a code metric which is used to evaluate the maintainability of software [2]. The Martin metrics were taken as independent variables.

The MI values obtained from jfreechart versions had very poor MI. This result allowed us to further investigate on which design metrics are contributing to the increase or decrease in MI values obtained across different versions.

IV. MACHINE LEARNING MODELS AND TECHNIQUES WITH RESULTS

The training data, testing data and the different machine learning techniques that were used to get better insight into the design metrics that influence MI have been introduced here. Out of the fifty two versions, 33% of the data was taken as the testing set and the remaining 66% was taken as the training set.

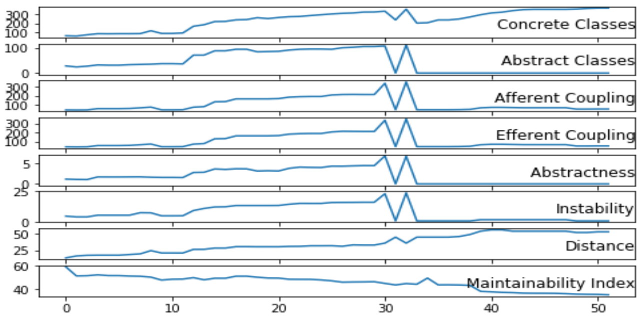

Figure 2 : Design metric and MI values obtained in all jfreechart versions

Figure 2 shows the design metric values that were obtained from fifty two versions of jfreechart software including the MI value. It is pretty clear that most of the jfreechart versions have less MI. Further, it also depicts a constant decrease in MI from different versions right from the first version till the last version. Figure 3 displays the metric values obtained from different versions of jfreechart.

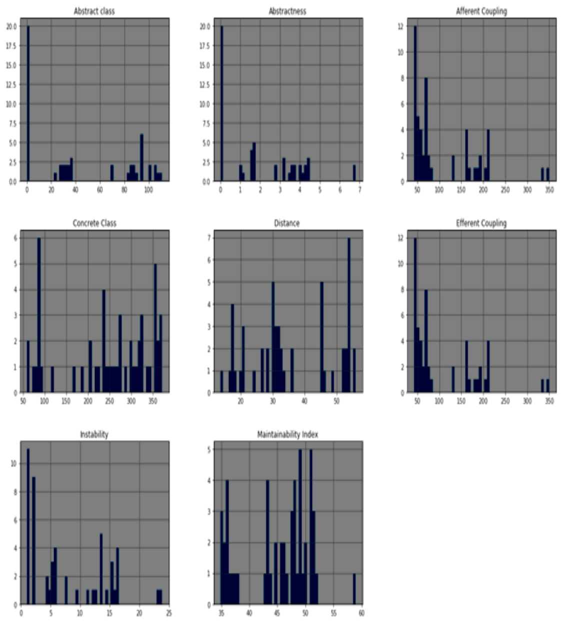

Figure 3 : Individual design metric histograms

IV.1 Linear Regression technique

Linear Regression performs the task of predicting a dependent variable value(y) depending on a given independent variable(x).

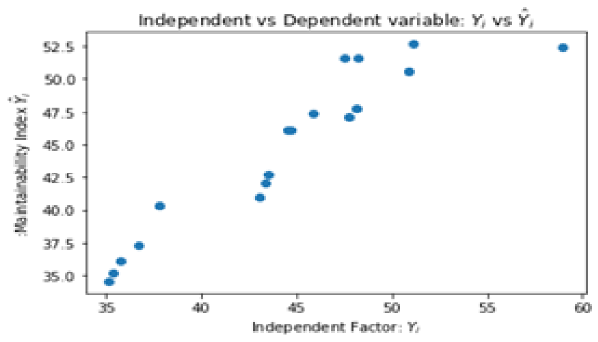

Figure 4 : Regression Curve

The results of the linear regression model obtained is depicted in Figure 4. It is very much evident that the regression curve does not form a straight line which suggests that the model has a lot of errors. Certain goodness of fit tests were done to zero-in on the errors in the numerical form. The MSE(Mean Squared Error) was found to be 5.19 and the RMSE(Root Mean Squared Error) value was 2.27. These values further substantiate the fact that errors are evident in the model.

IV.2 Gradient Boosting Regression Algorithm

Gradient boosting is one of the several techniques employed while predictive models. This is an efficient algorithm that converts a relatively poor hypotheses problem into a very good hypotheses. It is also an optimization algorithm that uses back propagation techniques to correct or minimize errors. In our study, this algorithm was used to minimize the errors obtained in our linear regression model. After the algorithm was applied, it is evident from Figure 5 that leaving two extreme cases, there is a good model that has been generated i.e. the errors have been minimized to a large extent. The MSE obtained now was 2.36 and the RMSE value was 1.12. The errors obtained in this model are definitely less when compared with the earlier model. (Figure 4).

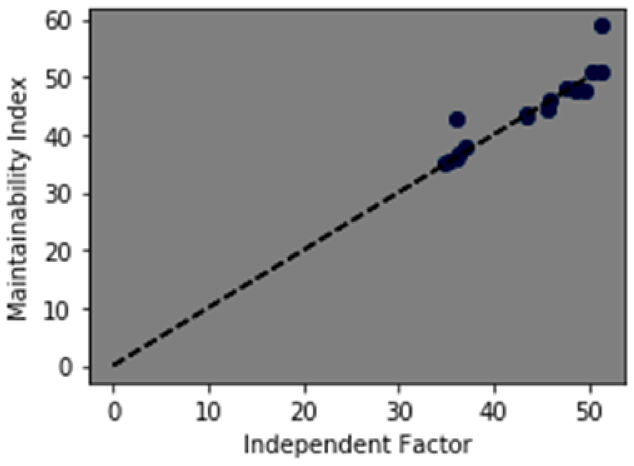

Figure 5 : Linear Regression Curve after applying Gradient Boost Algorithm

V. Important Observations and Conclusions

In this study, the Instability metric was seen as the most important contributor influencing MI. Therefore, coupling (both efferent and afferent) should be given utmost priority when new versions of software are released. The results obtained in this case study were different when compared with the case study done using statistical analysis [20] i.e. the prediction model that was obtained showed four design metrics namely concrete classes, abstract classes, instability and distance influencing MI. In the current study, we found only the Instability metric as one that influences MI the most. There is a need to perform more research studies by collecting huge datasets from different open source software applications. Further, rather than applying mere statistical techniques, using machine learning algorithms and techniques would help in arriving at better conclusions.

VI. Further Directions

Large software datasets need to be generated from different types of Object Oriented software. Several metric suites can be extracted from many software applications across different domains of software. Machine learning algorithms would certainly help in identifying metrics that are specifically influencing maintainability to a larger extent. Further, like maintainability which was studied in this research, other external quality attributes like usability, reliability and understandability etc… can also be researched in future works.

VII. Threats to Validity

The sample set taken for analysis is a small dataset and the results and inferences obtained cannot be taken as the final metrics towards predicting maintainability. Generally, machine learning algorithms are used on large and very large datasets to arrive at specific conclusions. Since the dataset used in this study is a small one, the results and conclusions need to be verified with many more studies. Research studies should be conducted on many more object oriented design metrics in arriving at specific conclusions.

REFERENCES

1. K.D. Welker, The software maintainability index revisited, *Journal of Defense Software Engineering*, pp. 18-21, 2001.
2. P. Oman and J. Hagemeister, Metrics for assessing a software system's maintainability, *Proceedings of the IEEE International Conference on Software Maintenance*, pp. 337-344, 1992.
3. P. Oman and J. Hagemeister, Constructing and testing of Polynomials predicting software maintainability, *Journal of Systems and Software, Vol. 24, No 3, pp. 251-266, 1994.*
4. M.M.T. Thwin and T.S. Quah, Application of neural networks for software quality prediction using object-oriented metrics,*Journal of Systems and Software*, Vol 76, No 2, pp. 147-156, 2005.
5. Y. Zhou and X.U. Baowen, Predicting the maintainability of open source software using design metrics, *Wuhan University Journal of Natural Sciences*, Vol 13, No 1, pp. 14-20, 2008.
6. Y.Zhou and H.Lueng, Predicting object-oriented software Maintainability using multivariate adaptive regression splines, *Journal of Systems and Software*, Vol 80, No 8, pp. 1349-1361, 2007.

7. K.K. Aggarwal, Y. Singh, A. Kaur and R. Malhotra, Application of artificial neural network for predicting maintainability using object-oriented metrics, *World Academy of Science, Engineering and Technology*, No 22, 2006.
8. P. Niemeyer and J. Knudsen , Learning Java, *O'Reilly & Associates* : 2005.
9. S.C. Misra, Modeling design/coding factors that drive maintainability of software systems, *Software Quality Journal*, Vol 13, pp. 297-320, 2005.
10. R. Martin, Agile Software Development Principles : Principles, Patterns and Practices, *Prentice Hall*, 2003.
11. Robert C Martin,"*UML 0.9 Lexicon*", B Class, Citeseer.
12. http://clarkware.com/software/JDepend.html.
13. H.M. Halstead, Elements of Software Science, N. Holland : *Elsevier*, 1997.
14. S. Muthanna, K. Kontogiannis, K. Ponnambalam and B. Stacey, A maintainability model for industrial software systems using design level metrics, *Proceedings of the Seventh Working Conference on Reverse Engineering*, pp. 248-256, 2000.
15. S.S. Yau and J.S. Collofello, Some stability measures for software Maintenance, IEEE *Transactions on Software Engineering*, Vol 6, No 6, pp. 545-552, 1980.
16. F. Zhuo, B. Lowther, P. Oman and J. Hagemeister, Constructing and testing software maintainability assessment models, *Proceedings of the First International Software Metrics Symposium*, IEEE CS Press, pp. 61-70, 1993.
17. M.O. Elish, A.H.A. Yafei and M.A. Mulhem, Empirical comparison of three metrics suites for fault prediction inpackages of object-oriented systems : a case study of eclipse, *Advances in Engineering Software*, Vol 42, No 10, pp. 852-859, 2011.
18. IEEE, IEEE Standard Glossary of Software Engineering Terminology, Report IEEE Std 610.12-1990, IEEE, 1990.
19. Madhwaraj K.G., Empirical comparison of two metrics suites for maintainability prediction in packages of object-oriented systems : A case study of open source software, *Journal of Computer Science*, Vol 10, No. 11, pp. 2330-2338, 2014.
20. Madhwaraj K.G., Predicting the maintainability of object oriented software using design metrics – An evolutionary case study of open source software, *International Review on Computers and Software*, Vol 9, No. 6, 2014.
21. Malhotra, R., & Bansal, A.J. (2012). Fault Prediction Using Statistical and Machine Learning Methods for Improving Software Quality *JIPS*, *8*, 241-262.
22. Arunima Jaiswal, Ruchika Malhotra, Software reliability prediction using machine learning techniques, *International Journal of System Assurance Engineering and Management*, 2018, Volume 9, Number 1, Pages 230-244.
23. Padhy, N., Panigrahi, R., & Neeraja, K. (2019). Threshold estimation from software metrics by using evolutionary techniques and its proposed algorithms, models. *Evolutionary Intelligence*, 1-15.

AUTHOR PROFILE

Dr. Madhwaraj Kango Gopal is working as a Professor in the Department of Computer Applications at New Horizon College of Engineering. He obtained a PhD in Computer Science from Anna University, Chennai and an M.Phil Degree in Computer Science from MS University, MCA from the University of Madras and a B.Sc from Bharathidasan University. His research interests include Software Quality Management, Machine Learning, Deep Neural Networks and Predictive Modelling. He has published many papers in International Journals indexed in the Scopus database.

Ms. Amirthavalli is a full-time PhD scholar in the Department of Information Technology at SSN College of Engineering, Kalavakkam. She obtained her ME in Computer Science from Anna University Chennai, M.Phil in Computer Science from Manonmaniam Sundaranar University, Tirunelveli, a Masters in Computer Application from Bharathidasan University and a Bachelor's degree in Electronics from University of Madras. Her research interests are Internet of Things, Internet Security, Predictive modelling of software etc…She has also published papers in International and National Conferences.