



International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS)

(Open Access, Double Blind Peer-reviewed, Refereed and Indexed Journal)

www.iasir.net

Application of Machine Learning Algorithms in Predicting Maintainability

Sandhya Tarwani

Assistant Professor, HMR Institute of Technology and Management
New Delhi, India

Abstract— Software maintenance phase of Software Development Lifecycle (SDLC) is the most expensive and complex phase that requires nearly 60-70% of the total project cost. Due to this, many software fail to get repair within real time constraint. Ascribe to technology advancements and changing requirements, software must be well developed and maintained to get adapted. Hence, it is necessary to predict software maintainability in the early phases of the lifecycle so that optimization of resources can be possible and cost can be reduced. Software Maintainability is the quality attribute of software product that explains the ease with which modifications can be performed. The main aim of this study is to propose the use of General Regression neural network (GRNN) for the software maintainability prediction and measure its performance with various other machine learning techniques such as Decision Tree, Support Vector Machine, Linear regression and Radial basis function neural network. The empirical study is conducted with the help of two open source datasets and change between the lines of two versions is considered as maintenance effort. Results of this study show that GRNN algorithm performs better than other classifiers; hence it can be used as sound alternative in the prediction of software maintainability. This study would be helpful in achieving better resource allocation hence it will be useful for developers and maintainers.

Keywords—Software Maintainability, Machine learning algorithm, Prediction accuracy measures, Software Metrics, Neural Networks

I. INTRODUCTION

Software maintenance is the most complex and expensive phase of the software development lifecycle. It is the period of modification of the software after its development and its duration extends till the time it is no longer in use. According to the observations by Pressman [1], 70% of the total efforts are being used in maintenance phase, 17% of the work is used to correct faults, 18% is used for the portability and more than half share of 65% is used to implement the changes made in the requirements due to many reasons. Requirement change is common problem now days that software developer faces due to competition in market.

Software maintainability is the ease with which software can be modified, updated and corrected. According to Institute of Electrical and Electronics Engineers (IEEE) standards [2], it is the ease with which a software system or component can be modified to correct faults, improve performance or other attributes. There are various metrics proposed for the calculation of software maintainability like Software Maturity Index (SMI) [3] that considers measurement of software maintainability with respect to change or addition of modules. It can be said that software maintainability is the combine measure of various attributes of the software like readability, understandability etc. It is evident from literature that there exists a strong relation between design metric and software maintainability prediction [4, 5, 6, 7, 8, 9].

In this paper, empirical study has been conducted to predict the software maintainability with the help of open source software. The main aim is to investigate the relationship between a number of metrics and maintainability with the help of General Regression Neural Network (GRNN). Many OO metrics have been designed to measure the various characteristics of the software like inheritance, coupling, lines of code etc. In this study, we have collected 17 independent metrics against one dependent metrics that is CHANGE. The following research questions are the overview of our study which has been conducted.

RQ1: Whether GRNN algorithm can be applied to software maintainability prediction?

RQ2: Does GRNN perform better than other machine learning classifiers?

RQ3: Does GRNN perform better in terms of time complexity than other machine learning classifiers?

A set of five machine learning algorithms is used to form the prediction model with the help of DTReg¹ (Decision Tree and Regression) tool. It is predictive modeling software created for any input data. The rest of the paper is organized as follows: section II explains the related work conducted in this field. Section III gives the description of machine learning algorithm used. Section IV discusses the empirical data collection. Section V depicts the results along with prediction accuracy formulas and section VI deals with conclusion and future work.

¹ <https://www.dtreng.com/>

II. RELATED WORK

Various studies have been conducted and proposed in literature till now for predicting software maintainability prediction. In 2002, Thwin and Quah [10] predicted the software maintainability with the help of neural network applied on three real time industrial systems. In 2003, Dagpinar and Jhanke[11] predicted the software maintainability with the help of OO software metrics. Misra [12] used twenty design/code metrics to indicate the effect of them on software maintainability. In 2005, Koten and Gray [13] accurately predicted the maintainability with Bayesian network as compared to other regression-based models. In 2006, Zhou and Leung [14] in his study found that Multivariate adaptive regression splines (MARS) outperform other machine learning algorithms in predicting maintainability. In 2009, Elish and Elish[15] used Treenet model for achieving prediction accuracy for maintainability. Although in literature many authors have proposed various models but in this study, we have applied GRNN on two open source software for maintainability prediction and compared its performance with other machine learning algorithms.

III. ALGORITHMS USED

A. General Regression Neural Network (GRNN)

GRNN are the neural network that performs regression where target value is continuous. It is based on kernel regression networks [16] that do not require back propagation for iterative training procedures. It draws function estimates directly from training sets. The architecture consists of input, hidden, pattern (summation) and output layer as shown in fig. 1. For each predictor values, there is one neuron in the input layer that is feed to each of the neurons in the hidden layer. The predictor and target values are stored in the neurons of hidden layer. The pattern layer consist of two neurons namely denominator summation unit (DSU) and numerator summation unit (NSU). The output layer consists of predicted target value that is obtained by dividing value resulted in NSU by the value in DSU.

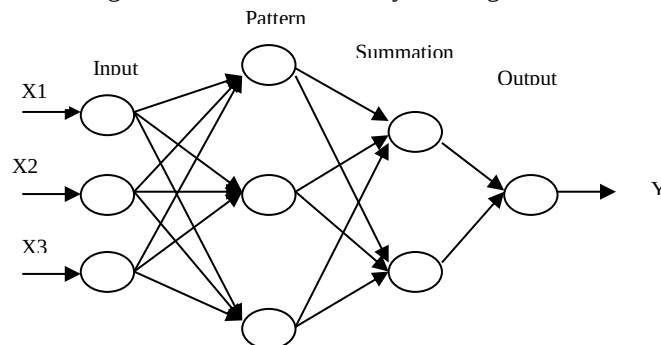


Figure 1 General Regression Neural Network (GRNN)

B. Decision Tree

Decision tree is a predictive model that predicts the value of target variable based on the values of various input variable. It is a simple method based on classification.

C. Linear Regression

Linear regression is a modeling approach that is used to figure out the relationship between dependent and independent variables. It uses linear predictor functions to model the relationship.

D. Support Vector Machine (SVM)

SVM is used to analyze the data for classification and regression analysis and therefore they are considered as supervised learning models. It is used for separating lines for distinguishing between objects of different class to handle classification task.

E. Radial basis function neural network (RBFN)

It is an artificial neural network used in function approximation, classification, etc. It uses radial basis functions as activation functions. It consists of inner layer, hidden layer with non-linear RBF activation function and a linear output layer which consists of inputs and neuron parameters.

Analysis of all the algorithms is done on the basis of prevalent accuracy measures.

IV. EMPIRICAL DATA COLLECTION

A. Metrics

In software engineering, software metrics is the answer that is used to measure different aspects of the software like inheritance, coupling etc. We have extracted 17 different OO metrics with the help of plug-in Metric reloaded in IntelliJ Idea IDE². Beyond Compare³ tool is used to calculate the change in two version of the software.

Table I describes all the independent metrics used in this study.

B. Open Source Projects

In this study we have used two open source java software projects downloaded from sourceforge.net⁴.

² <https://www.jetbrains.com/idea/>

³ <http://www.scootersoftware.com/download.php>

⁴ <http://sourceforge.net/>

- 1) jTDS (version 1.2.8 and 1.2.9): It is an open source jdbc 3.0 type 4 driver for Microsoft SQL Server and Sybase Adaptive Server Enterprise. It contains 64 classes.
- 2) jWebUnit (version 3.1 and 3.2): It is a java based framework for testing web applications. It helps in the verification of application correctness with the help of high-level java API. It contains 22 classes.

The dependent variable change is calculated as follows:

- 1) Change is counted as one if a new line is added or deleted in the dataset.
- 2) Modification of a line deals with addition and deletion simultaneously, therefore change is counted as two.
- 3) Blank line deletion or addition and modification in comment are not counted as change.

Descriptive statistics such as minimum, maximum, mean, standard deviation of OO metric is calculated but due to space shortage, table is not inserted only few observations are provided below:

- 1) The mean value of DIT and MIF is lowest in both datasets which indicates lowest inheritance.
- 2) The mean value of AHF is 100 in jwebunit which indicates all attributes are hidden.
- 3) The mean value of MHF is 100 in jwebunit which says it has little functionality.
- 4) The mean value of PF is 85.3917 in jTDS which indicates maximum methods are overridden in all derived classes.
- 5) The mean value of LCOM is low in both datasets, therefore it is easy to maintain and use.

V. RESULTS

Two open source datasets have been used in this study for software maintainability prediction along with five machine learning algorithms. Two prediction accuracy measures have helped in finding the best model for maintainability and are discussed below.

A. Prediction Accuracy Measures

Systems which are based on machine learning models uses prediction accuracy measures for evaluation. Kitchenham [21] proposed various parameters but we have used Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) as prediction accuracy measures. Error is measured with the help of difference between the actual and predicted value of CHANGE. Error measurement and accuracy are inversely related to each other. That is accuracy is high only when the error measurements will be low. The two parameters are described below:

1) Mean Absolute Error

It is the quantity used to measure the closeness of predictions with the actual events. It is calculated by taking average of absolute values of the difference between predicted and actual value as shown in (1).

$$MAE = \frac{|f_i - y_i|}{n} \quad (1)$$

Where f_i is predicted value, y_i is actual value and n is the total number of classes.

2) Root Mean Square Error

It is calculated by taking the standard deviation between the predicted value and the actual value as shown in (2). The individual difference between the predicted and actual value is called as residual. It is used to forecast error for a particular variable and not between variable.

$$RMSE = \sqrt{\frac{(f_i - y_i)^2}{n}} \quad (2)$$

Where f_i is predicted value, y_i is actual value and n is the total number of classes.

B. Cross validation

Cross validation is required when a model is built. This will help in generalizing the results with other independent data sets. When prediction accuracies are performed then validation plays an important role to measure how efficiently that model performs in practice. Prediction model consists of known data that is called training set and unknown data called testing set which helps in testing of the proposed model. Cross validation mainly requires partition of sample data sets as data used in maintainability prediction is of small size. 10-fold cross validation has been used in this study which divides the data sets into 10 equal sized sub-samples. Each sub-sample is used to test the model while other is used for training.

Table II summarizes the results of all the machine learning algorithms in which rows represent the type of algorithm used and columns represents the value of MAE and RMSE on two datasets used. Fig. 2 and fig. 3 represent the graphical representation of the results obtained.

Table I DESCRIPTION OF OO METRICS USED

Metric	Ref.	Definition
Coupling Between Object(CBO)	[17]	It is the count of number of classes that are coupled to a particular class. High CBO value results in high complexity therefore undesirable.
Depth of inheritance tree (DIT)	[17]	It is the maximum inheritance path from class to the root class.
Lack of cohesion of methods(LCOM)	[17]	It is a measure of how well the methods of a class co-operate to achieve aims of the class. A low value of LCOM is desirable as it is more cohesive hence easy to maintain and use.
Response for class(RFC)	[17]	It is the set of methods that are executed by a class along with all other methods called by that class. A large RFC value indicated the presence of more faults.
Number of children(NOC)	[17]	It is the number of immediate subclasses of a class. A high value indicates more use of base class which is desirable.

Weighted methods for class(WMC)	[17]	It is defined as sum of all the complexities of local methods of a class. A high value of WMC results in high number of faults.
Cyclomatic complexity(Ocavg)	[18]	It indicates complexity of the class.
Dependencies(Dcy &Dcy*)	[18]	It defines the cyclic and transitive dependencies of the class.
Javadoc function(jf)	[18]	It calculates the number of java functions of javadoc.
Javadoc Line of code(jLOC)	[18]	It calculates the number of lines of code of javadoc.
Javadoc methods(jm)	[18]	It calculates the number of java methods of javadoc.
Line of code(LOC)	[19]	It is used to measure the size of the program by counting the number of lines of the source code.
Attribute Hiding Factor(AHF)	[20]	It is used to measure how variables are encapsulated in a class. AHF=100% is the ideal case that is desirable in a code which indicates all attributes are hidden.
Method Hiding Factor(MHF)	[20]	It measures how methods are encapsulated in a class. A high value of MHF indicates very little functionality.
Polymorphism factor(PF)	[20]	It is the degree of measure of overriding methods in the class inheritance tree. It is calculated as number of actual override methods divided by number of total override methods.
Method inheritance factor(MIF)	[20]	It is calculated as number of inherited methods divided by total number of inherited methods in a class. A high value of MIF indicates a class that inherits many methods from its ancestor class.

Table II MEAN ABSOLUTE ERROR AND ROOT MEAN SQUARED ERROR VALUE FOR BOTH DATASETS

Algorithm Used	jTDS Dataset		jwebunit Dataset	
	MAE	RMSE	MAE	RMSE
Decision Tree (DT)	29.298607	58.996514	1.0909091	1.8781035
Linear regression (LR)	53.244576	91.995872	29.398621	81.085369
Radial Basis function Neural network(RBFN)	33.715479	60.106959	1.4905189	2.3251425
General Regression Neural Network (GRNN)	29.288607	58.916514	0.3689558	0.9559137
Support Vector Machine (SVM)	53.244576	91.995872	0.9964273	1.5082089

The three research questions raised in the introduction section are discussed below.

RQ1: Whether GRNN algorithm can be applied to software maintainability prediction?

Prediction accuracy measure will help in explaining that whether GRNN is suitably applied to the datasets or not. It can be seen from Table II that the value of MAE and RMSE in case of GRNN is least in both the data sets. Therefore, it can be concluded that GRNN can be applied to maintainability prediction.

RQ2: Does GRNN perform better than other machine learning classifiers?

Friedman ranking test [24] is used across multiple test attempts to detect the differences in treatment. It ranks the rows in a table then interprets the results by considering the values of ranks by column. On conducting the test, it was found that it assigns lowest rank to the GRNN in terms of MAE and RMSE as shown in Table III and IV, therefore it can be concluded that GRNN performs better than other machine learning algorithms.

RQ3: Does GRNN perform better in terms of time complexity than other machine learning classifiers?

Time taken to form the machine learning model is noted and is given in Table V. It can be seen from the table that GRNN performs better in jTDS as compared to jwebunit but a clear conclusion cannot be drawn. Therefore, Friedman rank test is applied and as it can be seen in Table VI that mean rank of GRNN is more as compared to DT and LR. Therefore, it can be concluded that in terms of time complexity, GRNN is not best machine learning model.

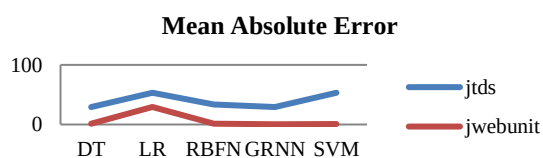


Figure 2 Mean Absolute Error

Table III RESULT OF FRIEDMAN TEST ON MAE

MAE	
Algorithms	Mean Rank
DT	2.25
LR	4.75
RBFN	3.00
GRNN	1.25
SVM	3.75

TableV TIME TAKEN BY EACH ALGORITHM (IN SECONDS)

	DT	LR	RBFN	GRNN	SVM
Jtds	0.26	0.30	5.19	0.33	0.30
Jwebunit	0.28	0.24	2.94	0.17	1.28

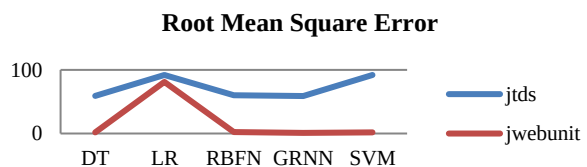


Figure 3 Root Mean Squared Error

Table IV RESULT OF FRIEDMAN TEST ON RMSE

RMSE	
Algorithms	Mean Rank
DT	2.25
LR	4.75
RBFN	3.00
GRNN	1.25
SVM	3.75

Table VI RESULT OF FRIEDMAN TEST ON TIME

Time	
Algorithms	Mean Rank
DT	2.00
LR	2.25
RBFN	5.00
GRNN	2.50
SVM	3.25

VI. CONCLUSION AND FUTURE WORK

In this study, two open source datasets have been used to predict the software maintainability with the help of machine learning algorithms. Prediction accuracy measure like MAE and RMSE are used and it was found that GRNN model outperforms the other machine learning algorithms used. In future, we are planning to conduct the same study on large datasets and different language to generalize the results. We have only predicted the software maintainability with the help of GRNN, in future we are planning to use this concept on other software engineering classification and estimation problem like effort and cost estimation.

REFERENCES

- [1] R.S. Pressman, Software Engineering: A Practitioner's Approach, 6 Edition, McGraw Hill, Book Co, 2005.
- [2] IEEE Std. 610.12-1990.: Standard Glossary of Software Engineering Terminology, IEEE Computer Society Press, Los Alamitos, CA, 1993.
- [3] "Software Engineering Standards", 1994 Edition, IEEE, 1994.
- [4] K.K. Aggarwal, Y. Singh, P. Chandra, and M. Puri, "Measurement of Software Maintainability Using a Fuzzy Model", *Journal of Computer Sciences*, vol. 1, no.4, pp. 538-542, 2005 ISSN 1549- 3636 © 2005 Science Publications.
- [5] A. Kaur, K. Kaur, and R. Malhotra, "Soft Computing Approaches for Prediction of Software Maintenance Effort", *International Journal of Computer Applications*, vol 1, no. 16, pp : 0975 – 8887, 2010.
- [6] D. Coleman, B. Lowther and P. Oman, "The Application of Software Maintainability Models in Industrial Software Systems", *Journal of System Software*, pp. 3-16, 1995.
- [7] C.J. Burgess and M. Lefley, "Can genetic programming improve software effort estimation? A comparative evaluation", *Information and Software Technology*, Vol. 43, pp. 863-873, 2001.
- [8] J.L. Dallal, "Object-oriented class maintainability prediction using internal quality attributes", *Information and Software Technology*, Vol. 55, pp. 2028-2048, 2013.
- [9] F.Zhuo, B. Lowther, P. Oman and J. Hagemeister, "Constructing and Testing Software Maintainability Assessment Models", *1st Int. Proc. Software Metrics Symp.*, Baltimore, MD, 1993, pp. 61-70.
- [10] M. Thwin, and T. Quah, "Application of neural networks for predicting software development processing", ©IEEE, vol. 5, pp.2312-2316.
- [11] M. Dagpinar, and J.H Jahnke, "Predicting Maintainability with Object-Oriented Metrics - An Empirical Comparison", *Proceedings of the 10th Working Conference reverse Engineering*, pp. 155-164, 2003.
- [12] S. Misra, "Modeling design/coding factors that drive maintainability of software systems", *Software Quality Journal*, vol. 13, no. 3, pp. 297-320, 2005.
- [13] C. V. Kotten and A. R. Gray, "An application of Bayesian network for predicting object-oriented software maintainability", *Information and Software Technology*, Vol. 48, pp. 59-67, 2005.
- [14] Y. Zhou and H. Leung, "Predicting object oriented software maintainability using multivariate adaptive regression splines", *The Journal of Systems and Software*, Vol. 80, pp. 1349-1361, 2006.
- [15] M.O. Elish, and K.O. Elish, "Application of TreeNet in Predicting Object-Oriented Software Maintainability: A Comparative Study", *European Conference on Software Maintenance and Reengineering*, pp 1534-5351, March 2009, DOI 10.1109/ CSMR.2009.57.
- [16] H. B. Celikoglu and H. K. Cigizoglu, "Public transportation trip flow modelling with generalized regression neural network", *Advances in Engineering Software*, Vol. 38, pp. 71-79, 2007.
- [17] S. Chidamber and C. Kemerer : "Towards a Metrics Suite for Object Oriented Design", *Conference on Object-Oriented Programming: Systems, Languages and Applications (OOSPLA 91)*, Published in SIGPLAN Notices, vol. 26, no. 11, pp. 197- 211, 1991.
- [18] B. Meyer, "Object-oriented software construction," in *Prentice Hall*, second edition, 1998.
- [19] L. Bennet P and S. E. Burton, "Problems in application software maintenance", *Communications of the ACM*, vol. 24, no.11, pp. 763-769, 1981.
- [20] F.B. Abreu and R. Carapuça, "Object-Oriented Software Engineering: Measuring and Controlling the Development Process," in *Proc. 4th Int. Conf. on Software Quality, McLean, Virginia, USA.*, 1994 doi=10.1.1.14.1382&rep=rep1&type=pdf.
- [21] B.A. Kitchenham, C.M. Pickard, S.G. MacDonell, and M.J. shepperd, "What accuracy statistics really measure", *Proceedings-Software, IEEE*, vol. 148, no. 3, pp. 81-85, 2001.
- [22] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings", *The Annals of Mathematical Statistics*, 11(1):86-92, (1940). DOI:10.1214/aoms/1177731944.