Quantification of Dynamic Metrics for Software Maintainability Prediction

Dhyanendra Jain[1], Ashu Jain[2], Amit Kumar Pandey[3]
Department of Information Technology, Northern India Engineeirng College, New Delhi, India
[1]dhyanendra.jain@gmail.com
[2]ajainashu@gmail.com
[3]amitpandey33@gmail.com

***Abstract*—Software maintenance is one of the important stages of the software development life cycle (SDLC) which needs to be performed carefully to reduce the cost and the effort associated with the software. Software maintainability is the ability to perform the changes in the software without the recurring bugs. If we are able to predict which classes of the object oriented (OO) software need maintenance, it will be very helpful to turn down the cost and the manpower required by the software. In this article, dynamic metrics of the software are used as the internal attributes and the number of changes made from one version of software to another, is taken as the external attribute for the prediction of software maintainability. Dynamic metrics capture more information as they are collected during the execution of the software, hence they are more accurate. For the prediction purpose, machine learning algorithms have been used. Out of the algorithms used, decision table algorithm performs the best in terms of MAE and RMSE.**

Keywords—software maintainability, machine learning, dynamic metrics,mean absolute error, root mean squared error, object oriented.

I. INTRODUCTION

Software maintenance is an exceptionally expensive action that incorporates error rectification, updation, deletion of out of date abilities, and improvements. The change in software is required due to the adjustment in the prerequisites of the client, change in the technology, change in the stage where the software will be sent. Since change is inescapable, mechanisms must be created for assessment, controlling and making adjustments.

So any work done to change the lines of code in software after it is in operation, is thought to be maintenance work. The design is to safeguard the estimation of software even after the delivery of the software. Measure of effort which is spent on keeping up the software is considerably much higher than creating it. Maintenance may traverse for a long time, while development might be 1-2 years. Accordingly creating software is not as hard as it is difficult to maintain it. The maintenance cost can be controlled by checking the software metrics amid the development stage.

Software metrics are the quantitative estimation of the different highlights of software which incorporate coupling, cohesion, abstraction, polymorphism, and so forth. Software metrics are calculated at each phase of software development life cycle (SDLC) and enable us to make different judgments about the software, which thus helps in taking different specialized and administrative choices. Previously, analysts focused on the static metrics which can be caught by the static investigation of the software program. Static examination can be as walkthroughs and inspections. Static metrics just catch the structural conduct of the software, genuine code isn't executed. In any case, now the analysts are moving towards the course of dynamic metrics. Dynamic metrics are estimated by the dynamic investigation of the software, when the genuine code is executing, i.e. at run time. With the assistance of dynamic metrics, we can discover answers to such a large number of question like, what number of lines of code are in actual getting executed, what number of strategies have been called at run time, which class gets coupled to another class, and so forth.

In this article, dynamic metrics have been used for the prediction of software maintainability. Four machine learning models: Linear Regression, Radial Basis Function (RBF) Neural Network, MultiLayer Perceptron(MLP) Neural Network and Decision table, are used for analyzing the datasets collected from object oriented(OO) software. Three open source software projects – SoundHelix, orDrumBox, and jXLS (from sourceforge.net) have been used to collect the datasets.

The rest of the paper is organized as follows: Section 2 describes the related work done in the field of prediction of software maintainability. Section 3 gives the overview of the algorithms used. Section 4 discusses the empirical data collection. Section 5 presents the accuracy measures used for evaluating the prediction models. Section 6 includes the result. In section 7, conclusion and directions of future work are provided.

II. RELATED WORK

In this section, work related to dynamic metrics and software maintainability prediction is presented in brief. Gupta and Chabra[1] thought about the execution of different methodologies for the dynamic examination of OO frameworks. They inferred that Aspect Oriented Programming (AOP) is the most proficient strategy for the dynamic examination and easier to execute. Briand et al[2] concentrated on the effect of software quality indicators on software maintenance efforts and they found that software quality markers beneficially affect software maintenance. Dagpinar and Jhanke[3] utilized OO software metrics for the software maintainability prediction. The outcomes demonstrated that size and import direct coupling metrics fill in as the critical indicators for software maintainability but inheritance, cohesion and indirect/export coupling don't. Jain et al[4] used the evolutionary algorithm for the prediction of software maintainability.

They used mean absolute error and root mean squared error as the accuracy parameters. Thwin and Quah[5] utilized two neural network models, which are, ward neural network and general regression neural network (GRNN) for the estimation of software reliability and practicality utilizing OO software metrics. They found that GRNN network foresee more precisely than the ward network model. Jain and Chug[6] compared the performance of dynamic metrics with that of static metrics for the purpose of maintainability prediction.

In this paper, dynamic metrics are used for software maintainability prediction using machine learning classifiers.

III. ALGORITHMS USED

For the prediction purpose, four machine learning algorithms are used for training and testing the model. A brief description of the classifiers is given in this section.

A. *Linear Regression*

The general thought of regression is to analyze two things: (1) does an arrangement of predictor variables make a decent showing with regards to predicting an outcome variable? (2) Which variables specifically are noteworthy predictors of the result variable, and how do they– demonstrated by the greatness and indication of the beta estimates– affect the result variable? These regression gauges are utilized to clarify the connection between one ward variable and at least one autonomous variable.

B. *Multilayer Perceptron(MLP)*

Multi Layer perceptron (MLP) is a feedforward neural network with at least one layer amongst input and output layer. Feedforward implies that input streams in a single course from input to output layer (forward). This sort of network is prepared with the back propagation learning algorithm.

C. *Radial Basis Function(RBFN) NeuralNetwork*

A RBFN performs classification by estimating the input's closeness to cases from the training set. Each RBFN neuron stores a "model", which is only one of the cases from the training set. When we need to group another input, every neuron registers the Euclidean separation between the input and its model. Generally, if the input more nearly looks like the class A models than the class B models, it is named class A.

D. *Decision table*

It breakdowns the given data into two attributes in a hierarchical tree by using the best possible classification question. Using this hierarchical tree, test instance is classified.

IV. EMPIRICAL DATA COLLECTION

A. *Open source Projects*

In this study, three open source projects have been used for the analysis of machine learning algorithms for software maintainability prediction. All the three projects have been coded in java language. The Description of datasets of these projects is depicted in table I.

DATASETS

Project	Version	Number of Classes
SoundHelix	0.7.1 and 0.8	67
jXLS	1.0.5 and 1.0.6	78
jDrumbox	0.9.08 and 0.9.081	218

V. DYNAMIC METRICS

For predicting the software maintainability, dynamic metrics have been used as the internal quality indicators. CHANGE is used as the external quality indicator which is calculated by counting the number of lines of code [10] added deleted or modified. The description of the metrics is given in table II [9].

DYNAMIC METRICS

Metric	Description
Loose Class Coupling (LCC)	Measures the low dependency of a class on other classes at run time.
Tight Class Coupling (TCC)	Measures the high dependency of a class on other classes at run time.
Lack of cohesion of methods (LCOM)	Calculates the lack of cohesion of methods at run time low value of LCOM is required.
Lines of Code (LOC)	Number of lines of code executed at run time
Lines of Comment (LOCm)	Number of lines of comments used at run time.
Number of Assertions per KLOC (NAK)	Number of declarations per KLOC at execution time.
Number of Children (NOC)	Number of immediate subclasses of a class at run time.
Number of Fields (NOF)	Total number of fields defined at run time
Number of Methods (NOM)	Total number of methods defined at run time.
Number of Static Fields (NOSF)	Total number of static fields defined at run time.
Number of Static Methods (NOSM)	Total number of static methods defined at run time.
Number of Test Methods (NTM)	Total number of test methods which get executed.
Weighted Method Count(WMC)	Calculates method complexity at run time.

VI. PREDICTION ACCURACY MEASURES

According to the rules of Kitchenham [7], we have utilized the Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to assess the machine learning models. Error is estimated by ascertaining the distinction between the actual and predicted estimation of dependent variable CHANGE. Lower the estimation of the error, higher will be the exactness of model. The equation portrayal of the measures is given as takes after [11]:

1. Mean Absolute Error

$$MAE = \frac{|fi - yi|}{n} \qquad (1)$$

Where *fi* is the actual value, *yi* is the predicted value and n is total number of classes.

2. Root Mean Squared Error

$$RMSE = \frac{\sqrt{(fi - yi)^2}}{n} \qquad (2)$$

Where *fi* is the actual value, *yi* is the predicted value and n is total number of classes.

VII. RESULTS

Table III and IV presents the values of mean absolute error and root mean squared error respectively on the considered datasets. On obtaining the values of MAE and RMSE, Friedman test[8] is applied to rank the algorithms. Ranks for MAE and RMSE are given in table V and VI.

Graphical representation of MAE and RMSE values are depicted in figure 1 and 2.

MEAN ABSOLUTE ERROR (MAE)

	Linear Regression	Multilayer Perceptron	RBF	Decision Table
SoundHelix	16.15	9.045	15.06	17.33
jXLS	44.23	0.85	1.49	0.79
jDrumBox	6.81	13.71	6.14	5.31

ROOT MEAN SQUARED ERROR (RMSE)

	linear Regression	Multilayer Perceptron	RBF	Decision Table
SoundHelix	35.833	26.67	34.45	45.48
jXLS	54.16	8.73	8.78	8.72
jDrumBox	14.93	24.13	14.72	14.66

From table V and table VI, it can be concluded that decision table algorithm performs the best both in terms of MAE and RMSE for software maintainability prediction.

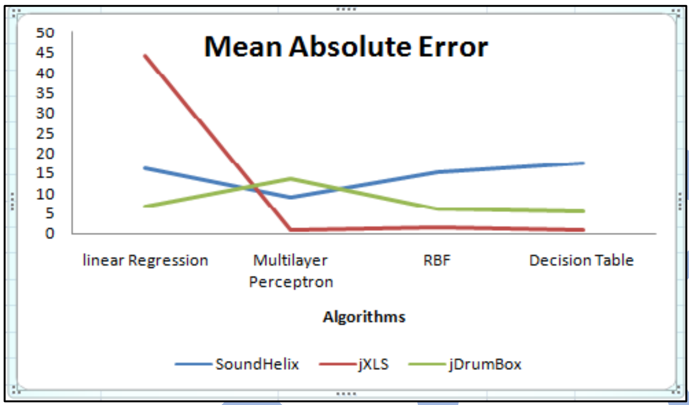

Fig 1. Mean Absolute Error

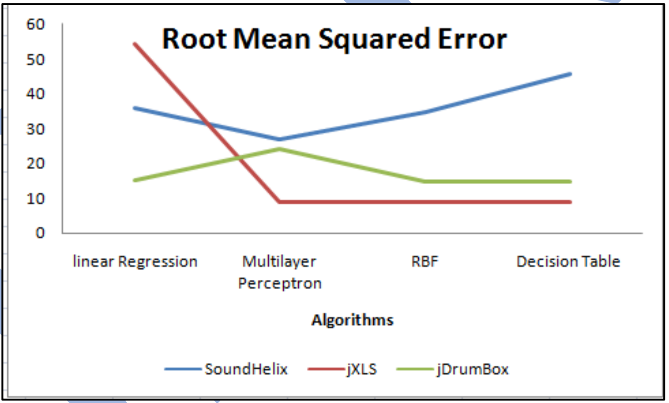

Fig. 2 Root Mean Squared Error

FRIEDMAN TEST RESULT ON MAE

MAE	
Algorithms	**Mean Rank**
Linear Regression	**3.33**
Multilayer Perceptron	**2.33**
RBF	**2.33**
Decision Table	**2.00**

FRIEDMAN TEST RESULTS ON RMSE

RMSE	
	Mean Rank
linear Regression	**3.33**
Multilayer Perceptron	**2.33**
RBF	**2.33**
Decision Table	**2.00**

VIII. CONCLUSION AND FUTURE SCOPE

In this article, dynamic metrics have been used as the independent variables and CHANGE is used as the dependent variable. Four machine learning algorithms have been applied on three datasets of different sizes. On applying the algorithms, decision table showed the minimum error both in terms of MAE and RMSE. Hence it can be concluded that decision table algorithm can be applied for software maintainability prediction with dynamic metrics as the internal quality attributes.

Our future work aims to generalize these results by considering the software projects of different languages and of big size. Also, we have analyzed only limited number of algorithms; hence there is a scope of other machine learning classifiers as well and of course the evolutionary algorithms.

REFERENCES

V.Gupta, and J. K. Chabra "Measurement of Dynamic Metrics Using Dynamic Analysis of Programs", Applied Computing Conference, Istanbul, Turkey, pp. 27-37, 2008.

L. Briand, C. Bunse, and J. Daly, "A controlled experiment for evaluating quality guidelines on the maintainability of object oriented design", *IEEE Transaction on software Engineering*, vol:27, no: 6, pp 513-530, 2001, DOI: 10.1109/32.926174.

M. Dagpinar, and J.H Jahnke, "Predicting Maintainability with Object-Oriented Metrics - An Empirical Comparison", *Proceedings of the 10th Working Conference.*

A. Jain, S.Tarwani and A. chug, "An empirical invetsigation of Evolutionary algorithm for software maintainability prediction", Electrical, Electronics and Computer Science (SCEECS), 2016 IEEE Students' Conference, july 2016

M. Thwin, and T. 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

A. Jain and A. chug, "Stepping Towards Dynamic Measurement for Object Oriented Software", Information Processing (IICIP), 2016 1st India International Conference, july 2017

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.

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.

Gaurav Kumar, Pradeep Kumar Bhatia, "Neuro-Fuzzy Model to Estimate & Optimize Quality and Performance of Component Based Software Engineering", ACM SIGSOFT Software Engineering Notes, Vol. 40, Issue 2, pp. 1-6, March 2015.

Gaurav Kumar, Pradeep Kumar Bhatia, "Optimization of Component Based Software Engineering Model Using Neural Network", BIJIT - BVICAM's International Journal of Information Technology, Vol. 6, No. 2, pp. 732-742, July - Dec. 2014.

Gaurav Kumar, Pradeep Kumar Bhatia, "Empirical Assessment and Optimization of Software Cost Estimation using Soft Computing Techniques", Book Title – Advanced Computing and Communication Technologies, Publisher - **Springer Singapore**, Vol. 452, pp. 117-130, June 2016.