On The Use Of KStar Algorithm For Predicting Object-Oriented Software Maintainability

Zighed Narimane
Departement of computer science.
Badji Mokhter Annaba
Badji Mokhter Annaba
Aannaba, Algeria
narimanezighed@gmail.com

Bounour Nora
Departement of computer science.
Badji Mokhter Annaba
Badji Mokhter Annaba
Annaba, Algeria
nora_bounour@yahoo.fr

Abstract— **This paper presents an ongoing work on using KStar algorithm to predict Object-Oriented software maintainability. The maintainability is measured as the number of changes made to code throughout the maintenance period by using Object-Oriented software metrics. We build a prediction model based on data collected from two different Object-Oriented systems. However, to figure out the advantages of KStar algorithm we made five experiments with the Weka machine learning workbench and to compare our proposed model with the other algorithms which are (linear Regression, Neural Network, Decision Tree, SVM), the prediction accuracy of all models is evaluated and compared using cross-validation and different types of accuracy measures. As a result, KStar yields better results and it demonstrated to be the best of them to predict more accurately than the other typical techniques.**

Keywords— ***Machine Learning Techniques, KStar, Software Maintainability, Object-Oriented Metrics.***

I. INTRODUCTION

Software maintenance has been one of the most complexes and hard tasks in the software development life cycle. Furthermore, the largest cost associated with any software product over its lifetime is the software maintenance cost [12]. Predicting software maintainability can be used to estimate the cost and effort required during the maintenance period. There are several definitions for the term "software maintainability", according to the IEEE Standard Glossary of software engineering terminology software maintainability is defined as “the ease with which a software system or component can be modified” [9].

Although the availability of maintainability prediction models that have been developed for measuring maintenance effort, the main aim is obtaining high prediction accuracies which are not the case of all these proposed models. A large number of studies have examined the link between OO software metrics and maintainability, they have found that in general these metrics can be used as predictors of maintenance effort [1], [5], [14].

In this paper, we propose the use of a novel Machine Learning (ML) technique “KStar algorithm” for software maintainability prediction. As well we present a comparative study of the use of Kstar algorithm and the most common ML techniques which are chosen from the machine learning platform Weka [8] such as linear Regression ‘LR’, Neural Network ‘MLP NN’, SVM ‘SMOreg’, Decision Tree ‘RepTree’. Thereafter, we investigate the accuracy and performances of ML-based models in predicting the software maintainability using the dataset previously collected. The results show that KStar algorithm gives better. The results suggest that for the QUES our model can predict maintainability more accurately than the other typical modeling techniques, while for UIMS is as accurate as the best modeling technique.

The remainder of this paper is organized as follows. Section 2 summarizes the related works conducted on software maintainability prediction using most commonly ML techniques. Section 3 provides an overview information of the datasets and the set of metrics used. Section 4 focuses on the various research methodologies used in predicting software maintainability and our constructed model. Section 5 includes the results of the model’s evaluation and analysis of the results. In section 6 we analyze the results of Kstar model using the UIMS and QUES data sets and section 7 concludes the paper.

II. RELATED WORKS

Several studies have been carried out for proposing different models to predict software maintainability. In this section, we provide an overview of the main existing predictive models with the common modeling techniques used.

Van and Gray in [14] have constructed a model for predicting the software maintainability of Object-Oriented systems using Bayesian Networks. Since the majority of prediction models have usually formed or built using a previous version of the system or a similar software project developed by the organization. The authors used the Object-Oriented metrics and datasets presented by Li and Henry [13]. The maintainability is measured as the number of code’s changes during a maintenance period. To build the Bayesian Network the Bayda tool was used. Bayda allows users to build a special type of Bayesian Networks called Bayesian naive classifier. In which a single node representing a dependent variable and it’s connected to all the other nodes that represent the predictor variables. The Bayesian network maintainability prediction model is built where a single node (change) is connected to the predictive variables OO (10 metrics). Then the precision of the model prediction is evaluated and compared with models-based regression. The results suggest that the Bayesian network model can predict maintainability more accurately than regression-based models for one system (UIMS) and almost as accurately as the best regression-based model for the other system (QUES).

Elish [5] used TreeNet to construct a software maintainability prediction model. They explored their experience on two famous data sets in the field of maintainability known as UIMS and QUES [13]. The authors have proved that it also provides competitive results compared to other models.

Aggarwal and al [1] used Neural Network as a technique to construct a prediction model to estimate the maintenance effort of OO software at the class level by estimating the number of rows modified per class, On the other hand, the authors in [2] aim to explore empirically the relationship between OO metrics and maintainability estimation. The values of the metrics studied were collected from a total of 110 classes from two software systems. The constructed ANN model belongs to the whole of the Multilayer Perceptron and the results of the latter's evaluation showed that the Mean Absolute Relative Error (MARE) was 0.265 of the model and ANN is capable of providing an adequate model for predicting maintenance effort and that OO metrics can be useful in guiding prediction.

Jindal and al [11] also used Neural Networks to develop a prediction model, but by using another type of neural network "Radial Basis Function Network".

In 2016 a recent study was conducted by Jain, Tarwani and Chug [10] to propose the use of Genetic Algorithms for predicting the software maintainability, and to compare its performance with various automatic learning techniques they extracted the OO metrics from four open source projects jTDS (64 classes), jWebUnit (22 classes), jXLS (78 classes) and SoundHelix (67 classes), using the tool (Chidamber and Kemerer Java Metrics). The Weka tool was also used to construct the prediction model. In their study, maintainability was measured by counting the number of changes at the code level, which was calculated by comparing each class of the two versions using the Beyond Compare tool. Finally, to evaluate the accuracy of predictions found they used Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) as precision measures of prediction which were proposed by Kitchenham. According to the results, the genetic algorithm gives more accurate predictions com-pared to other models of automatic learning-based prediction.

III. Experimental Data Collection

We perform our empirical experiment based on the metrics data collected from two commercial software packages UIMS and QUES. In this section, we summarize empirical data collection, which represent the independent and dependent variables.

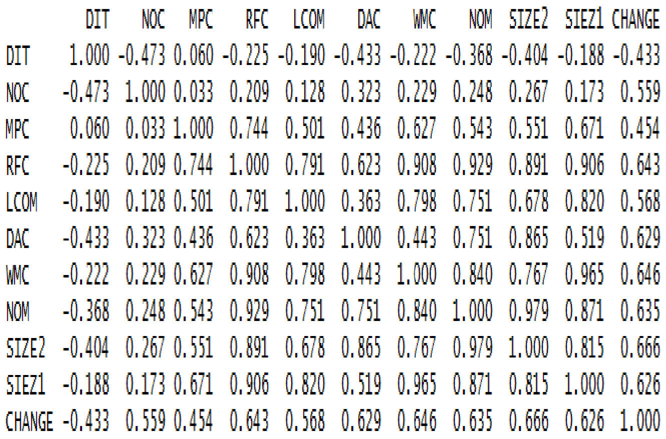
A. Metrics Selection

The independent variables which we have used in this study are values of ten Object-Oriented metrics and the dependent variable used is maintainability measure. In this study, we indicate software maintainability using the number of revised lines of code “LOC” (i.e., added, deleted, or changed) LOC during the maintenance phase as measure.

TABLE I. Description of the set of metrics used

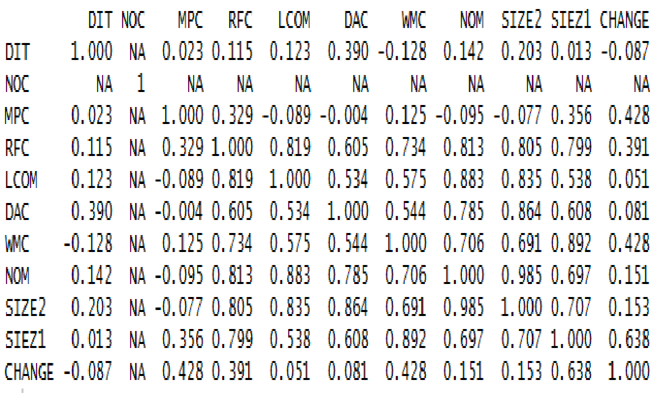
Metric	Description
WMC	The sum of McCabe’s cyclomatic complexity of all local methods in a given class.
DIT	The length of the longest path from a given class to the root in the inheritance hierarchy.
RFC	The number of methods that can potentially be executed in response to a message being received by an object of a given class.
NOC	The number of classes directly inherit from a given class.
LCOM	The number of pairs of local methods in a given class using no attribute in common
MPC	The number of send statements defined in a given class
DAC	The number of abstract data types defined in a given class
NOM	The number of methods implemented within a given class
SIZE1	The number of semicolons in a given class
SIZE2	The total number of attributes and the number of local methods in a given class
CHANGE	Number of lines changed in the class (insertion and deletion are independently counted as 1, change of the contents is counted as 2)

Fig 1. and Fig 2. show the correlation coefficients between the OO metrics values in the data sets UIMS and QUES and the maintainability measure “change”. Correlation Coefficient measures the intensity of relationship between dependent and independents variables. We have used R language to calculate the correlation matrices.



	DIT	NOC	MPC	RFC	LCOM	DAC	WMC	NOM	SIZE2	SIEZ1	CHANGE
DIT	1.000	-0.473	0.060	-0.225	-0.190	-0.433	-0.222	-0.368	-0.404	-0.188	-0.433
NOC	-0.473	1.000	0.033	0.209	0.128	0.323	0.229	0.248	0.267	0.173	0.559
MPC	0.060	0.033	1.000	0.744	0.501	0.436	0.627	0.543	0.551	0.671	0.454
RFC	-0.225	0.209	0.744	1.000	0.791	0.623	0.908	0.929	0.891	0.906	0.643
LCOM	-0.190	0.128	0.501	0.791	1.000	0.363	0.798	0.751	0.678	0.820	0.568
DAC	-0.433	0.323	0.436	0.623	0.363	1.000	0.443	0.751	0.865	0.519	0.629
WMC	-0.222	0.229	0.627	0.908	0.798	0.443	1.000	0.840	0.767	0.965	0.646
NOM	-0.368	0.248	0.543	0.929	0.751	0.751	0.840	1.000	0.979	0.871	0.635
SIZE2	-0.404	0.267	0.551	0.891	0.678	0.865	0.767	0.979	1.000	0.815	0.666
SIEZ1	-0.188	0.173	0.671	0.906	0.820	0.519	0.965	0.871	0.815	1.000	0.626
CHANGE	-0.433	0.559	0.454	0.643	0.568	0.629	0.646	0.635	0.666	0.626	1.000

Fig. 1. Correlation coefficient in UIMS .



	DIT	NOC	MPC	RFC	LCOM	DAC	WMC	NOM	SIZE2	SIEZ1	CHANGE
DIT	1.000	NA	0.023	0.115	0.123	0.390	-0.128	0.142	0.203	0.013	-0.087
NOC	NA	1	NA	NA	NA	NA	NA	NA	NA	NA	NA
MPC	0.023	NA	1.000	0.329	-0.089	-0.004	0.125	-0.095	-0.077	0.356	0.428
RFC	0.115	NA	0.329	1.000	0.819	0.605	0.734	0.813	0.805	0.799	0.391
LCOM	0.123	NA	-0.089	0.819	1.000	0.534	0.575	0.883	0.835	0.538	0.051
DAC	0.390	NA	-0.004	0.605	0.534	1.000	0.544	0.785	0.864	0.608	0.081
WMC	-0.128	NA	0.125	0.734	0.575	0.544	1.000	0.706	0.691	0.892	0.428
NOM	0.142	NA	-0.095	0.813	0.883	0.785	0.706	1.000	0.985	0.697	0.151
SIZE2	0.203	NA	-0.077	0.805	0.835	0.864	0.691	0.985	1.000	0.707	0.153
SIEZ1	0.013	NA	0.356	0.799	0.538	0.608	0.892	0.697	0.707	1.000	0.638
CHANGE	-0.087	NA	0.428	0.391	0.051	0.081	0.428	0.151	0.153	0.638	1.000

Fig. 2. Correlation coefficient in QUES .

The correlation coefficient lies between -1 and 1. The positive linear relationship grows stronger as correlation coefficient nears 1, and the negative linear relationship grows stronger as correlation coefficient nears -1. No linear relationship is there if the correlation coefficient is 0. The results show that there is a significant correlation between CHANGE and the OO metrics, then all the metrics from the datasets are selected and will be used to build model.

B. *OO Software Datasets*

As we mentioned above, our empirical study was carried out using the datasets and metrics presented in [13] by Li and Henry. It concerns the two Object-Oriented systems: UIMS (User Interface Management System) with a total of 39 classes and QUES (Quality Evaluation System) with a total of 71 classes both systems were implemented in Classical-Ada language. The aim of this investigation is to predict the maintenance effort which is measured by using the number of lines changed per class which were changed during a 3 years of maintenance period. A line change could be an addition or a deletion. A change of the content of a line is counted as a deletion followed by an addition. This measurement is used in this study to estimate the maintainability of the OO systems.

The descriptive statistics of the UIMS and QUES datasets are shown in Table 2, the statistics were extracted from Weka tool .

TABLE II. CHARACTERISTIC OF DATSETS

Metrics	Min		Max		Mean		StdDev	
	U	q	U	q	U	q	U	q
DIT	0	0	4	4	2.15	1.91	0.90	0.52
NOC	0	0	8	0	0.94	0	2.01	0
MPC	1	2	12	42	4.33	18.02	3.40	8.33
RFC	2	12	101	156	23.20	54.29	20.18	32.78
LCOM	1	3	31	33	7.48	9.18	6.10	7.30
DAC	0	0	21	25	2.41	3.52	3.99	3.96
WMC	0	1	69	83	11.38	14.90	15.89	17.08
NOM	1	4	40	57	10.92	13.35	9.69	12.02
SIEZ2	1	4	61	82	13.53	18.02	13.10	15.20
SIZE1	4	115	439	1009	101.48	275.57	110.9	171.59
change	2	6	289	217	47.15	64.26	71.82	43.21

U: for UIMS dataset

q: for QUES dataset

A common problem that assumes often when using metrics and regression techniques is called multicollinearity. This problem arises when the independent variables are strongly correlated with each other which may affect the performance of the prediction model. To address the problem of multicollinearity we have transformed the data in uncorrelated variables. The task of choosing the most suitable representative set of metrics is known as feature selection. From feature selection techniques proposed in literature, we choose Correlation-based Feature Selection (CFS). Weka can detect and remove highly correlated input attributes automatically by setting weka. attributeSelection.CfsSubsetEval.

IV. PREDICTION MODELS

In this section, we introduce the commonly used modeling techniques: multivariate linear regression, artificial neural network, and regression tree. We describe the K* algorithm, which is regarded as a good technique for producing prediction models.

A. *KStar "K*"*

Lazy algorithms are a nonparametric approach. They defer the real work as long as possible, while other machine learning algorithms produce generalizations since they "meet" the data. Instance-based (IB) learners, also called Memory-based ones store the training instances in a lookup table and interpolate from these.

Among the Lazy algorithms we used KStar which is developed as an instance-based classifier, that tries to improve its performance for dealing with missing values, smoothness problems and both real and symbolic valued attributes, which is the class of a test instance and is based upon class of similar training instances similar to it, as determined by some similarity function. In IB learner problems, "each new instance is compared with existing ones using a distance metric, and the closest existing in-stance is used to assign the class to the new one" [4]. The principal difference of K* against other IB algorithms is the use of the entropy concept for defining its distance metric, which is calculated by mean of the complexity of transforming an instance into another; so, it is taken into account the probability of this transformation occurs in a "random walk away" manner. The classification with K* is made by summing the probabilities from the new instance to all of the members of a category. This must be done with the rest of the categories, to finally select that with the highest probability.

We used Weka data mining tool to conduct KStar Model. The training and testing dataset selection is being employed using k-fold cross validation procedure where the entire data set is divided randomly into k subsets (k = 10) and every time one of the k subsets is used as training data and the remaining (k -1) subsets are being used as testing data set so as to validate the prediction model for software reliability. Here, 10 cross validation is used where nine parts are used for training and one part is used for validation taken randomly ten times and results are recorded for each of the ten runs.

B. *Neural Network*

The Second model prediction was done by using Multi-Layer Perception Neural Net-work. The MLP is one of the most widely implemented neural network topologies. In terms of mapping abilities, the MLP is believed to be capable of approximating arbi-trary functions [7]. This has been important in the study of nonlinear dynamics, and other function mapping problems. MLPs are normally trained with the back-propagation algorithm that consists of an input layer, one or more hidden layers of nonlinear-ly activating nodes and an output layer. Each node in one layer connects with a certain weight to every other node in the following layer. The parameters of this model were initialized as follows. Back-propagation algorithm was used for training. Sigmoid was used as an activation function. Number of hidden layers was 5. Learning rate was 0.3 with momentum

0.2. Number of epochs to train through was 500. Validation threshold was 20.

C. *Decision Tree*

Regression tree (RepTree) is a variant of decision trees that predicts values of continuous variables instead of labels of classification [3]. RT has over 209 different computer configurations, adapted from Witten and Frank (2000). A regression tree is built through a recursive partitioning process. This is an iterative process of splitting the data into partitions, and then splitting it up further on each branches. Initially all of the records in the training set are put together in one node. The algorithm chooses an independent variable with values that minimize the sum of the squared deviations from the mean in the separate parts.

D. *Support Vector Machine*

The purpose of support vector machine (SVM) was originally developed for solving the classification problems [15] but it was extended to the domain of regression problems and recently it has been applied successfully in many software prediction models. It can, unlike ANN approaches, produce prediction models with excellent generalization performance. As a result, SVM is gaining in popularity in the machine learning community.

E. *Linear Regression*

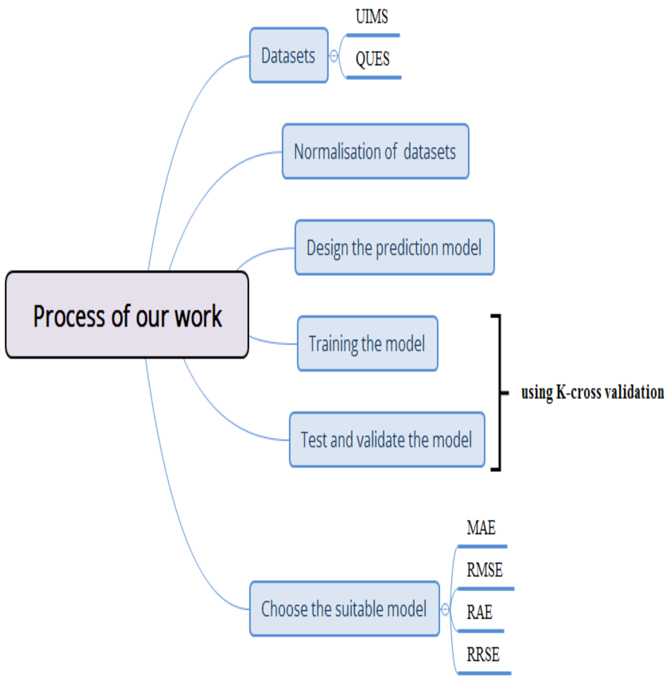
Multivariate Linear Regression (MLR) [6]. is the most commonly used technique for modeling the relationship between two or more independent variables and a dependent variable by fitting a linear equation to observed data. The main advantage of this technique is its simplicity. The general form of MLR model can be given by:

$$Y= a_0+a_1 x i_0+\ldots..+akxik$$

$$y= a0+a1xi0+\ldots..+akxik+ei$$

Which are: $a_0,\ldots,a_k$, the parameters to be estimated, Yi the dependent variable to be predicted, yi the actual value of the dependent variable, and e_i is the error in the prediction of the i case.

Fig3 presents the process which we went through for developing the prediction models. This process is applied to each of the other machine learning algorithms taken into consideration.



Process of our work
Datasets
UIMS
QUES
Normalisation of datasets
Design the prediction model
Training the model
Test and validate the model
using K-cross validation
Choose the suitable model
MAE
RMSE
RAE
RRSE

Fig. 3. Process of our work

V. MODEL PERFORMANCE EVALUATION

In order to evaluate the performance of the proposed model, we selected the standard and commonly used prediction accuracy measures metrics which they were defined as below in table 3:

TABLE III. EFFICACY MEASURES USED FOR EVALUATING PERFORMANCE OF MODELS

Measure	Definition
MAE (1)	Mean Absolute Error measures the average magnitude of the errors in a set of forecasts, without considering their direction.
RMSE (2)	Root Mean Squared Error shows differences between values predicted by a model and the values actually observed from the thing being modeled.
RAE (3)	Relative Absolute Error very similar to the relative squared error in the sense that it is also relative to a simple predictor
RRSE (4)	Root Relative Squared Error takes the total squared error and normalizes it by dividing by the total squared error of the simple predictor.

- *MAE*

$$\text{MAE}= \frac{1}{n}\sum_1^n |pi - ai| \quad (1)$$

- RMSE

$$\text{RMSE}=\sqrt{\frac{1}{n}\sum_{i=1}^n |pi - ai]} \quad (2)$$

- RAE

$$\text{RAE}=\frac{\frac{1}{n}\sum_{1}^{n}|pi-ai|}{\frac{1}{n}\sum_{1}^{n}|ai-A|} \quad (3)$$

- RRSE

$$\text{RRSE}=\frac{\sum_{i=1}^{n}(pi-ai)2}{\sum_{i=1}^{n}(pi-A)2} \quad (4)$$

Where pi is the prediction value, ai true value and Ai is given by the formula: $\frac{1}{n}\sum_{i=1}^{n} ai$.

These measures show how closely the actual and predicted values are correlated with each other.

VI. EMPIRICAL RESULTS

In this section, we analyze the results of KStar model and the other results using the two datasets. The prediction performance of the KStar model was assessed and compared with those of the linear regression the artificial neural network , the regression tree, and the support vector models. The results show that the KStar models can effectively predict the maintainability of OO software systems. For both UIMS and QUES data set, the KStar model is as accurate as the best prediction model. Thus, overall, it is concluded that the KStar model is better than the other models built using typical modeling techniques and it can be a useful modeling technique for software maintainability prediction.

A. *Results using QUES dataset*

TABLE IV. EFFICACY MEASURES USED FOR EVALUATING PERFORMANCE OF MODELS

Technique	MAE	RMSE	RAE%	RRAE%
K*	0.081	0.133	52.44	65.00
RepTree	0.108	0.154	69.66	75.24
LR	0.117	0.151	74.89	74.00
MLP	0.120	0.152	76.84	74.21
SVM	0.113	0.160	72.49	78.52

B. *Results using UIMS dataset*

TABLE V. EFFICACY MEASURES USED FOR EVALUATING PERFORMANCE OF MODELS

Technique	MAE	RMSE	RAE	RRAE
K*	0.105	0.220	63.43	88.05
RepTree	0.144	0.235	86.55	93.92
LR	0.120	0.209	72.30	83.38
MLP	0.178	0.337	107.15	134.36
SVM	0.120	0.211	72.32	84.12

VII. CONCLUSION

In this paper, we employ a novel exploratory modeling technique for software maintainability prediction. First, we empirically evaluated the capability and suitability of the application of KStar Algorithm to build maintainability prediction model by means of Object-Oriented software metrics. Thus, an experiment was conducted to evaluate the performance of the KStar model to predict the software maintainability in terms of the possible number of changes. We used public data sets introduced by Li and Henry. In addition, number of common modeling techniques are examined to find the machine learning algorithm that gives accurate results. The results show that KStar algorithm can effectively predict the maintainability of OO software systems. One direction of future work would be conducting additional empirical studies with other datasets to further support the findings of this paper

REFERENCES

[1] Aggarwal, K.K., Singh, Y., Kaur, A., Malhotra, R. : Application of Artificial Neural Network for Predicting Maintainability using Object Oriented Metrics. World Academy of Science, Engineering and Technology, pp.285-289, (2008).

[2] Aggarwal, K.K., Singh, Y., Kaur, A., Sangwan, O.P, : A Neural Net Based Approach To Test Oracle. ACM SIGSOFT Software Engineering Notes, pp.1-6, (2004).

[3] Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J. : Classification and Regression Trees. Chapman & Hall, London, (1993).

[4] Cleary, J.G., Trigg, L.E. : K*: An Instance-based Learner Using an Entropic Distance Measure. the Twelfth International Conference on Machine Learning, Tahoe City, California, July 9–12, pp.108–114, (1995).

[5] Elish, M., Elish, K.: Application of TreeNet in Predicting Object-Oriented Software Maintainability: A Comparative Study. In The 13th European Conference on Software Maintenance and Reengineering (CSMR), IEEE, Germany, pp.69-78, (2009).

[6] Freidman, J.: Multivariate adaptive regression splines. Annals of Statistics, pp.1-141, (1991).

[7] Haykin, S.: Neural networks: a comprehensive foundation. Prentice Hall, New Jersey, (1999).

[8] Hall, M. and al : The WEKA data mining software: an update. ACM New York, NY, USA, , June, pp. 10-18, (2009).

[9] International Software Testing Qualifications Board: IEEE Standard glossary of terms used in Software Engineering, (2011).

[10] Jain, A., Tarwani, S., Chug, A. : An Empirical Investigation of Evolutionary Algorithm for Software Maintainability Prediction. Students' Conference on Electrical, Electronics and Computer Science (SCEECS), IEEE, India, pp.1-6, (2016).

[11] Jindal, R., Malhotra, R., Jain, A. : Predicting Software Maintenance Effort Using Neural Networks. The 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), IEEE, India, (2015).

[12] Kumar, R., Dhanda, N. : Maintainability Measurement Model for Object Oriented Design. International Journal of Advanced Research in Computer and Communication Engineering, , pp.68-71, (2015).

[13] Li, W., Henry, S.: Object-oriented metrics that predict maintainability. Journal of Systems and Software, pp.111–122,(1993).

[14] Van Koten, C., Gray, A.R.: An application of Bayesian network for predicting object oriented software maintainability. Information and Software Technology Journal, pp.59-67, (2006).

[15] Vapnik, V., Golowich, S., Smola, A.: Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing. Advances in Neural Information Processing Systems 9. MIT Press, pp.281-287, (1997).