Software Fault Proneness Prediction Using Support Vector Machines

Yogesh Singh, Arvinder Kaur, Ruchika Malhotra

***Abstract*— Empirical validation of software metrics to predict quality using machine learning methods is important to ensure their practical relevance in the software organizations. In this paper, we build a Support Vector Machine (SVM) model to find the relationship between object-oriented metrics given by Chidamber and Kemerer and fault proneness. The proposed model is empirically evaluated using public domain KC1 NASA data set. The performance of the SVM method was evaluated by Receiver Operating Characteristic (ROC) analysis. Based on these results, it is reasonable to claim that such models could help for planning and performing testing by focusing resources on fault-prone parts of the design and code. Thus, the study shows that SVM method may also be used in constructing software quality models.**

Keywords: Metrics, Object-oriented, Software Quality, Empirical validation, Fault prediction, Support vector machine, Receiver Operating Characteristics analysis

I. INTRODUCTION

As the complexity and the constraints under which the software is developed are increasing, it is difficult to produce software without faults. One way to deal with this problem is to predict important software quality attributes such as fault-proneness, effort, testability, maintainability, and reliability during early phases of software development. The software metrics [4, 9, 10, 14, 15, 24, 25, 29-32, 41] may be used in predicting these quality attributes.

Manuscript received March 21, 2009.
Prof. Yogesh Singh is with Guru Gobind Singh Indraprastha University, Delhi, India (email: ys66@rediffmail.com)
Dr. Arvinder Kaur is with Guru Gobind Singh Indraprastha University, Delhi, India (e-mail: arvinderkaurtakkar@yahoo.com.)
Ruchika Malhotra (Corresponding Author phone: 91-011-26431421) is with Guru Gobind Singh Indraprastha University, Delhi, India (email: ruchikamalhotra2004@yahoo.com)

The behaviour of several quantitative models ranging from using simple linear discriminant analysis to more complex logistic regression, decision tree, and SVM have been proposed. The regression and machine learning approaches are inherently different, raising the question to analyze the performance of these methods.

Several empirical studies have been carried out to predict the fault proneness models such as [1, 2, 5, 7, 11, 12, 14, 16, 20, 21, 23, 27, 34, 35, 40, 44]. There is a need to empirically validate the machine learning methods in predicting software quality attributes. Therefore, more data-based empirical studies that can be used to verify the capability of machine learning methods are needed. The evidence gathered through these empirical studies is considered to be the strongest support for testing a given hypothesis.

Thus, there is a need for both 1) empirically validating the results of machine leaning methods such as SVM and 2) finding the relation between OO metrics given by Chidamber and Kemerer [15] and fault proneness models. Now we briefly describe the work done in this study. In this paper, we investigate the following issue:

- Are the fault proneness models predicted using SVM method feasible and adaptable?
- How accurately and precisely do the OO metrics predict faults?

In order to perform the analysis we validate the performance of the SVM method using public domain KC1 NASA data set [42]. The 145 classes in this data were developed using C++ language.

The contributions of this paper are summarized as follows: First, we performed the analysis of public domain NASA data set [42], therefore analyzing valuable data in an important area where empirical studies and data are limited. Second, we applied SVM method to predict the effect of OO metrics on fault proneness. To the best of our knowledge, there has been no such

previous research using SVM method to predict software fault proneness. The proposed results showed that SVM method predict faulty classes with high accuracy. However, since our analysis is based on only one data set, this study should be replicated on different data sets to generalize our findings.

The paper is organized as follows: Section 2 summarizes the metrics studied, and describes sources from which data is collected. Section 3 presents the overview of the SVM method. The results of the study are given in section 4. The model is evaluated in section 5. Conclusions of the research are presented in section 6.

II. RESEARCH BACKGROUND

In this section, we present the summary of metrics studied in this paper (Section A), and empirical data collection (Section B).

A. Dependent and Independent Variables

The binary dependent variable in our study is fault proneness. The goal of our study is to explore empirically the relationship between OO metrics and fault proneness. Fault proneness is defined as the probability of fault detection in a class [2]. We use machine learning methods to predict the probability of fault proneness. Our dependent variable will be predicted based on the faults found during software development life cycle. The metrics given by [15] are summarized in Table 1.

B. Empirical Data Collection

This study makes use of the public domain data set KC1 from the NASA Metrics Data Program [42]. The data in KC1 was collected from a storage management system for receiving/processing ground data, which was implemented in the C++ programming language. This system consists of 145 classes that comprise of 2107 methods, with 40K lines of code. KC1 provides both class-level and method-level static metrics. At the method level, 21 software product metrics based on product's complexity, size and vocabulary are given. Five types of defects such as the number of defects and density of defects are also given. At the class level, values of 10 metrics are computed including seven metrics given by Chidamber and Kemerer [15]. These seven OO metrics are taken in our study (see Table 1) for analyses.

TABLE 1
METRICS STUDIED

Metric	Definition
Coupling between Objects (CBO)	CBO for a class is count of the number of other classes to which it is coupled and vice versa.
Lack of Cohesion (LCOM)	For each data field in a class, the percentage of the methods in the class using that data field; the percentages are averaged then subtracted from 100%.
Number of Children (NOC)	The NOC is the number of immediate subclasses of a class in a hierarchy.
Depth of Inheritance (DIT)	The depth of a class within the inheritance hierarchy is the maximum number of steps from the class node to the root of the tree and is measured by the number of ancestor classes.
Weighted Methods per Class (WMC)	A count of methods implemented within a class.
Response for a Class (RFC)	A count of methods implemented within a class plus the number of methods accessible to an object class due to inheritance.
Source Lines Of Code (SLOC)	It counts the lines of code.

III. MODEL PREDICTION USING SUPPORT VECTOR MACHINE (SVM) METHOD

SVM are useful tools for performing data classification, and have been successfully used in applications such as face identification, medical diagnosis, text classification [43], pattern recognition [13], chinese character classification [45], and identification of organisms [33]. SVM constructs an N-dimensional hyperplane that optimally separates the data set into two categories. The purpose of SVM modeling is to find the optimal hyperplane that separates clusters of vector in such a way that cases with one category of the dependent variable on one side of the plane and the cases with the other category on the other side of the plane [37]. The support vectors are the vectors near the hyperplane. The SVM modeling finds the hyperpalne that is oriented so that the margin between the support vectors is maximized. When the points are separated by a nonlinear region, SVM handles this by using a kernel function inorder to map the data into a different space when a hyperplane can be used to do the separation. Details on SVM can be found in [17, 18].

The recommended kernel function is the Radial basis Function (RBF) [37]. Thus, we used RBF function in SVM modeling to predict faulty classes in this study. The RBF kernel maps non-linearly data into a higher dimensional space, so it can handle non linear relationships between the dependent and the independent variables. Figure 1 shows the RBF kernel. One category of the dependent variable is shown as rectangles and the other as triangles. The shaded circles and rectangles are support vectors.

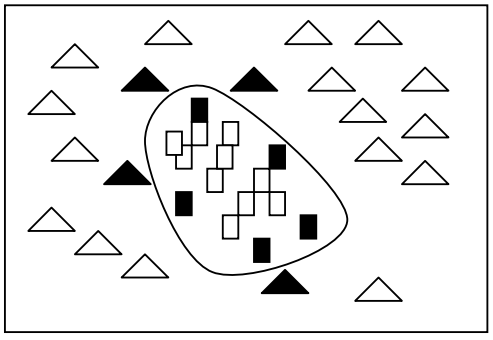

Figure 1: Radial basis function

Given a set of $(xi, yi),\ldots\ldots,(xm, ym)$ and $y_i \in \{-1,+1\}$ training samples. α_i =(i=1,……,m) is a lagrangian multipliers. K(xi,yi) is called a kernel function and b is a bias. The discriminant function D of two class SVM is given below [45]:

$$D(x) = \sum_{i=1}^{m} y_i \alpha_i K(x_i, x) + b \qquad (1)$$

Then an input pattern x is classified as [45]:

$$x = \begin{Bmatrix} +1 \text{ if } D(x) > 0 \\ -1 \text{ if } D(x) < 0 \end{Bmatrix} \qquad (2)$$

The performance of the models predicted was evaluated using sensitivity, specificity, precision, completeness, and Area Under the Curve (AUC). Details on these measures can be found in [19]. The ROC curve, which is defined as a plot of sensitivity on the y-coordinate versus its 1-specificity on the x coordinate, is an effective method of evaluating the quality or performance of the predicted models [22].

IV. ANALYSIS RESULTS

This section presents the analysis results, following the procedure described in Section 3. The results of the model predicted and evaluated are presented.

Results of prediction model: Table 2 shows the sensitivity, specificity, precision, completeness, and cutoff point of each metric and the model predicted. SLOC and CBO metrics have the highest values of the sensitivity and completeness. However, in case of NOC metric all the classes were predicted to be non faulty hence the results are not shown, as testing all the classes will be a high waste of the testing resources.

The model was applied to 145 classes and Table 3 presents the results of correctness of the fault proneness model predicted. As shown in Table 2, the cut off point for the model build to predict fault proneness is 0.44. Out of 59 classes, actually fault prone, 45 classes were predicted to be fault prone. The sensitivity of the model is 76.27 percent. Similarly, 70 out of 86 classes were predicted not to be fault prone. Thus, the specificity of the model is 81.39 percent. The completeness of the model is 85.66 percent. Thus,

the accuracy of the model is very high.

V. MODEL EVALUATION USING ROC ANALYSIS

The accuracy of the models predicted is somewhat optimistic since the models are applied on same data set from which they are derived. To predict accuracy of the model it should be applied on different data sets thus we performed 10-cross validation of the SVM model. Details on cross validation can be found in [39]. Table 4 summarizes the results of 10-cross validation of models using the SVM method. The AUC of the model predicted is 0.89.

In line with other predictive models, likewise findings of this study need to be externally validated. Although regression analysis is widely used method in literature, our results show that performance of the SVM model is good. In a previous study, we validated SVM using open source data set [38]. In [38], results also showed good performance of the SVM method. Therefore, it appears that the model predicted using machine-learning methods might lead to development of optimum software quality models for predicting fault prone classes.

Planning and resource allocating for inspection and testing is difficult and it is usually done on empirical basis. The model predicted in the above section could be of great help for planning and executing testing activities. To illustrate how the prediction model can be applied in practice, consider Figure 2. The values for the predicted fault proneness were taken from the results of validation of the models. On X-axis, we plot

TABLE 2
SENSITIVITY, SPECIFICITY, PRECISION, AND COMPLETENESS

Metric	Sensitivity	Specificity	Precision	Completeness	Cutoff
CBO	76.27	68.6	71.7	82.24	0.40
WMC	55.93	65.1	61.37	71.18	0.41
RFC	52.54	66.27	60.68	67.13	0.37
SLOC	62.7	75.58	70.34	76.32	0.32
LCOM	59.32	63.95	62.75	65.88	0.33
NOC	-	-	-	-	-
DIT	71.18	29	46.2	78.34	0.38
Model IV	76.27	81.39	78.62	85.66	0.44

TABLE 3
PREDICTED CORRECTNESS OF MODEL

	Predicted	
	Po<=0.43	Po>0.43
Observed	Not faulty	Faulty
Not faulty	70	16
Faulty	14(92)	45(550)

TABLE 4
. RESULTS OF 10-CROSS VALIDATION OF MODELS

SUPPORT VECTOR MACHINE	
	Model
Cutoff point	0.45
Sensitivity	69.49
Specificity	82.55
Precision	77.24
Completeness	79.59
AUC	0.89

the probability of classes sorted in increasing order of their cutoff points. On Y-axis each column presents two lines. The first and second line shows the sensitivity and completeness of the predicted model, respectively. For example, in Figure 2 at X=0, the sensitivity of the model is 100% and the completeness is 100%.

VI. CONCLUSIONS

The goal of our research is to empirically analyze the performance of the SVM method. We find the individual and combined effect of each metric on fault proneness of classes. Based on public domain NASA data set KC1, we empirically analyzed the performance of the SVM method using ROC analysis.

We analyzed OO design metrics given by Chidamber and Kemerer. Our main results are as follows:

- The CBO, RFC, and SLOC metrics were found to be related to fault proneness. NOC and DIT metric was not found to be significantly related to fault proneness. Thus, the results of machine learning method (SVM) show that usefulness of NOC and DIT metric is poor.
- The SVM method yielded good AUC using ROC analysis. This study confirms that construction of the SVM models is feasible, adaptable to OO systems, and useful in predicting fault prone classes. While research continues, practitioners and researchers may apply machine learning methods for constructing the model to predict faulty classes.

As in all empirical studies, the relationship we established is valid only for certain population of systems. In this case, the authors can roughly characterize this population as "object-oriented, large-sized systems."

We plan to replicate our study to predict the models based on machine learning algorithms such as genetic algorithms. We will also focus on cost benefit analysis of models that will help to determine whether a given fault proneness model would be economically viable.

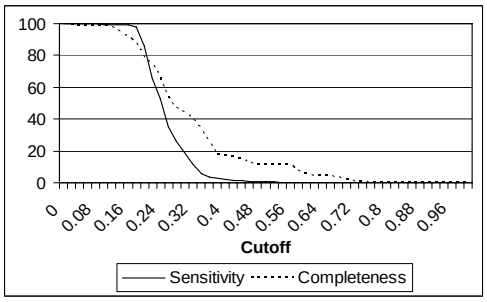

Figure 2: Sensitivity and Completeness of the Model

REFERENCES

[1] Aggarwal K.K., Singh Y., Kaur A., Malhotra R. (2007). Application of Artificial Neural Network for Predicting Fault Proneness Models, International Conference on Information Systems, Technology and. Management (ICISTM 2007), March 12-13, New Delhi, India.

[2] Aggarwal K.K., Singh Y., Kaur A., Malhotra R. (2008). Empirical Analysis for Investigating the Effect of Object-Oriented Metrics on Fault Proneness: A Replicated Case Study, Forthcoming in Software Process Improvement and Practice, Wiley.

[3] Aggarwal K.K., Singh Y., Kaur A., Malhotra R. (2006). Empirical study of object-oriented metrics. Journal of Object Technology, 5(8), 149-173.

[4] Aggarwal K.K., Singh Y., Kaur A., Malhotra R. (2005). Software reuse metrics for object-oriented systems. In Proceedings of the Third ACIS Int'l Conference on Software Engineering Research, Management and Applications (SERA '05), 48-55.

[5] Aggarwal K.K., Singh Y., Kaur A., Malhotra R. (2006). Investigating the Effect of Coupling Metrics on Fault Proneness in Object-Oriented Systems. Software Quality Professional, 8(4), 4-16.

[6] Barnett V., Price T. (1995). Outliers in Statistical Data. John Wiley & Sons.

[7] Basili,V., Briand, L., and Melo, W. (1996). A validation of object-oriented design metrics as quality indicators. IEEE Transactions on Software Engineering, 22(10), 751-761.

[8] Binkley, A., and Schach, S. (1998). Validation of the coupling dependency metric as a risk predictor. In Proceedings of the International Conference on Software Engineering, 452-455.

[9] Briand, L., Daly, W., and Wust J. (1998). Unified framework for cohesion measurement in object-oriented systems. Empirical Software Engineering, 3(1), 65-117.

[10] Briand, L., Daly, W., and Wust J. (1999). A unified framework for coupling measurement in object-oriented systems. IEEE Transactions on Software Engineering , 25(1), 91-121.

[11] Briand, L., Daly, W., and Wust J. (2000). Exploring the relationships between design measures and software quality. Journal of Systems and Software, 51(3), 245-273.

[12] Briand L., Wüst J., Lounis H. (2001). Replicated Case Studies for Investigating Quality Factors in Object-Oriented Designs, Empirical Software Engineering: An International Journal , 6(1), 11-58.

[13] Burges, C. (1998). A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2, 121–167.

[14] Cartwright, M., and Shepperd, M. (1999). An empirical investigation of an object-oriented software system. IEEE Transactions of Software Engineering, 26(8), 786-796.
[15] Chidamber, S., and Kamerer, C. (1994). A metrics suite for object-oriented design. IEEE Transactions on Software Engineering , 20(6), 476-493.
[16] Chidamber, S., Darcy, D., and Kemerer, C. (1998). Managerial use of metrics for object-oriented software: An exploratory analysis. IEEE Transactions on Software Engineering, 24(8), 629-639.
[17] Cortes, C., Vapnik, V. (1995). Support-vector networks. Machine Learning 20, 273–297.
[18] Cristianini, N., Shawe-Taylor, J. (2000). An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, Cambridge, UK.
[19] El Emam, K., Benlarbi, S., Goel, N. And Rai, S. 1999. A Validation of Object-Oriented Metrics, *Technical Report ERB-1063*, NRC.
[20] El Emam, K., Benlarbi, S., Goel, N., and Rai, S. (2001). The Confounding Effect of Class Size on The Validity of Object-Oriented Metrics. IEEE Transactions on Software Engineering, 27(7), 630-650.
[21] Gyimothy, T., Ferenc, R., Siket, I. (2005). Empirical validation of object-oriented metrics on open source software for fault prediction, IEEE Trans. Software Engineering, 31 (10), 897 – 910.
[22] Hanley, J., McNeil, BJ. (1982). The meaning and use of the area under a Receiver Operating Characteristic ROC curve. Radiology, 143: 29-36.
[23] Harrison, R., Counsell, S. J., and Nithi, R.V. (1998). An evaluation of MOOD set of object-oriented software metrics. IEEE Transactions on Software Engineering, 24(6), 491-496.
[24] Henderson-Sellers, B. (1996). Object-oriented metrics, measures of complexity. Englewood Cliffs, N.J.: Prentice Hall.
[25] Hitz, M., and Montazeri, B. (1995). Measuring coupling and cohesion in object-oriented systems. In Proceedings of the International Symposium on Applied Corporate Computing, Monterrey, Mexico.
[26] Hosmer, D., and Lemeshow, S. (1989). Applied logistic regression. New York: John Wiley and Sons.
[27] Khoshgaftaar, T.M., Allen, E.D., Hudepohl, J.P, Aud, S.J. (1997). Application of neural networks to software quality modeling of a very large telecommunications system," IEEE Transactions on Neural Networks, 8(4), 902-909.
[28] Kothari, C. R. (2004). Research Methodology. Methods and Techniques. New Delhi**:** New Age International Limited.
[29] Lake, A., and Cook, C. (1994). Use of factor analysis to develop OOP software complexity metrics. In Proceedings of the 6th Annual Oregon Workshop on Software Metrics, Silver Falls, Oregon.
[30] Lee, Y., Liang, B., Wu, S., and Wang, F. (1995). Measuring the coupling and cohesion of an object-oriented program based on information flow. In Proceedings of the International Conference on Software Quality, Maribor, Slovenia.
[31] Li, W., and Henry, S. (1993). Object-oriented metrics that predict maintainability. Journal of Systems and Software, 23(2), 111-122.
[32] Lorenz, M., and Kidd, J. (1994). Object-oriented software metrics. Englewood Cliffs, N.J.: Prentice-Hall.
[33] Morris, C., Autret, A., Boddy, L. (2001). Support vector machines for identifying organisms-a comparison with strongly partitioned radial basis function networks. Ecological Modeling 146, 57–67.
[34] Olague, H., Etzkorn, L., Gholston, S., and Quattlebaum, S. (2007). Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes. IEEE Transactions on software Engineering, 33(8), 402-419.
[35] Pai, G. (2007). Empirical analysis of Software Fault Content and Fault Proneness Using Bayesian Methods, IEEE Transactions on software Engineering, 33(10), 675-686.
[36] Porter, A., Selly, R. (1990). Empirically guided Software Development using Metric-Based Classification Trees, IEEE Software, 7(2), 46-54.
[37] Sherrod, P. (2003) DTreg Predictive Modeling Software.
[38] Singh, Y., Arvinder, K., Malhotra, R. (2009). Application of Support Vector Machine to Predict Fault Prone Classes, ACM SIGSOFT Software Engineering Notes, 34(9), 1-6.
[39] Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. J. Royal Stat. Soc., 36, 111-147.
[40] Tang, M.H, Kao,, M.H., and Chen, M.H. (1999). An Empirical Study on Object-Oriented Metrics, In Proceedings of Metrics, 242-249.
[41] Tegarden, D., Sheetz, S., and Monarchi, D. (1995). A software complexity model of object-oriented systems. Decision Support Systems 13 (3-4), 241-262.
[42] www.mdp.ivv.nasa.gov, NASA Metrics data Repository.
[43] Wang, X., Bi, D., and Wang, S. (2007). Fault recognition with Labeled multi-category', Third conference on Natural Computation, Haikou, China.
[44] Yuming, Z. and Hareton, L. (2006). Empirical analysis of Object-Oriented Design Metrics for predicting high severity faults. IEEE Transactions on Software Engineering, 32(10), 771-784.
[45] Zhao, L., Takagi, N. (2007). An application of Support vector machines to Chinese character classification problem. IEEE International Conference on systems, Man and Cybernetics, Montreal.