Int J Syst Assur Eng Manag (July-Sept 2010) 1(3):269–281
DOI 10.1007/s13198-011-0048-7

ORIGINAL ARTICLE

Empirical validation of object-oriented metrics for predicting fault proneness at different severity levels using support vector machines

Ruchika Malhotra · Arvinder Kaur · Yogesh Singh

Received: 24 March 2010 / Revised: 17 February 2011 / Published online: 19 April 2011

Abstract Empirical validation of software metrics to predict quality using machine learning methods is important to ensure their practical relevance in the software organizations. It would also be interesting to know the relationship between object-oriented metrics and fault proneness at different severity levels. 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, at different severity levels. The proposed models at different severity levels are empirically evaluated using public domain 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. The performance of the model predicted using high severity faults is low as compared to performance of the model predicted with respect to medium and low severity faults. Thus, the study shows that SVM method may also be used in constructing software quality models. However, similar types of studies are required to be carried out in order to establish the acceptability of the model.

R. Malhotra (✉)
Department of Software Engineering, Delhi Technological University, Bawana, Delhi 110042, India
e-mail: ruchikamalhotra2004@yahoo.com

A. Kaur · Y. Singh
University School of Information Technology, GGS Indraprastha University, Delhi 110403, India
e-mail: arvinderkaurtakkar@yahoo.com

Y. Singh
e-mail: ys66@rediffmail.com

Keywords Metrics · Object-oriented · Software quality · Empirical validation · Fault prediction · Support vector machine · Receiver operating characteristics analysis

Abbreviations

Coupling	Coupling is a measure of the degree of interdependence between classes
Cohesion	Cohesion is a measure of the degree to which the elements of a module are functionally related (Aggarwal et al. 2006a)
Severity	This value quantifies the impact of the fault on the overall environment
Fault proneness	The probability of the detection of a fault
Sensitivity	The ratio of predicted faulty classes and actual faulty classes
Specificity	The ratio of predicted non faulty classes and actual non faulty classes
Completeness	The number of faults in classes predicted fault-prone, divided by the total number of faults in the system
Precision	The number of classes that are predicted correctly (both faulty and not faulty), divided by the total number of classes
ROC curves	ROC curve, which is defined as a plot of sensitivity on the y-coordinate versus its 1-specificity on the x coordinate

1 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 quality attributes such as fault-proneness, effort, productivity, maintenance effort and reliability during early phases of software development. The software metrics may be used in predicting these quality attributes. Software faults vary considerably with respect to their severities.

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 evaluate the results of these methods.

Some empirical studies have been carried out to predict the fault proneness models such as (Basili et al. 1996; Binkley and Schach 1998; Briand et al. 2000; Cartwright and Shepperd 1999; Chidamber et al. 1998; Harrison et al. 1998; Li and Henry 1993; Yu et al. 2002; Yuming and Hareton 2006; Aggarwal et al. 2006b, Aggarwal et al. 2009; Olague et al. 2007; Pai 2007; Singh et al. 2010). There is a need to empirically validate the relationship between Object-Oriented (OO) metrics and fault proneness at different severity levels. Therefore, more data-based empirical studies that are capable of being verified by observation or experiment 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) building the fault proneness models with regard to severity of faults. 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 methods feasible and adaptable?
- How accurately and precisely do the OO metrics predict high severity faults?
- Which OO metrics are related to fault proneness of class with regard to high severity faults?
- How accurately and precisely do the OO metrics predict medium severity faults?
- Which OO metrics are related to fault proneness of class with regard to medium severity faults?
- How accurately and precisely do the OO metrics predict low severity faulty?
- Which OO metrics are related to fault proneness of class with regard to low severity faults?

Figure 1 presents the framework for software quality assessment. The inputs to the SVM method are process or product metrics and the outputs are the various observables that can be used to provide information about the software quality. In this work, we primarily investigate the use of SVM method in assessing software quality (shown in dotted in Fig. 1). The performance of machine-learning method SVM was evaluated in the study for predicting the fault proneness in the classes. The validation of the method is carried out using ROC analysis. To obtain a balance between the number of classes predicted as fault prone and number of classes predicted as not fault prone we use ROC curves. We also obtain the accuracy of predicted by calculating area under the curve from an ROC curve (El Emam et al. 1999). We investigate the accuracy of the fault proneness predictions using OO design metrics suite given by (Chidamber and Kamerer 1994), with particular focus on how accurately these metrics predict fault proneness when severity is taken into account. We categorized faults into three levels: faults are either of high severity, medium severity or of low severity. In order to perform the analysis we validate the performance of the SVM method using public domain KC1 NASA data set (NASA Data Repository). The 145 classes in this data were developed using C ++ language.

The paper is organized as follows: related work is summarized in Sect. 2. Section 3 summarizes the metrics studied, gives hypothesis to be tested in the study, and describes sources from which data is collected. Section 4 presents the research methodology followed in this study. The results of the study are given in Sect. 5. The model is evaluated in Sect. 6 and the applications of the model are given in Sect. 7. Section 8 presents threats to validity of the models. Conclusions of the research are presented in Sect. 9.

2 Related work

Based on a study of eight medium-sized systems, developed by students Basili et al. (1996) found that several of the (Chidamber and Kamerer 1994) metrics were associated with fault proneness.

Tang et al. (1999) analyzed (Chidamber and Kamerer 1994) OO metrics suite on three industrial applications developed in C ++. They found none of the metrics examined to be significant except RFC and WMC. El Emam et al. (2001) examined a large telecommunication application developed in C ++ and found that class size i.e. SLOC has confounding effect of most OO metrics on faults.

Briand et al. (2000) has extracted 49 metrics to identify a suitable model for predicting fault proneness of classes. The system under investigation was medium sized C ++ software system developed by undergraduate/graduate students. The eight systems under study consisted of a

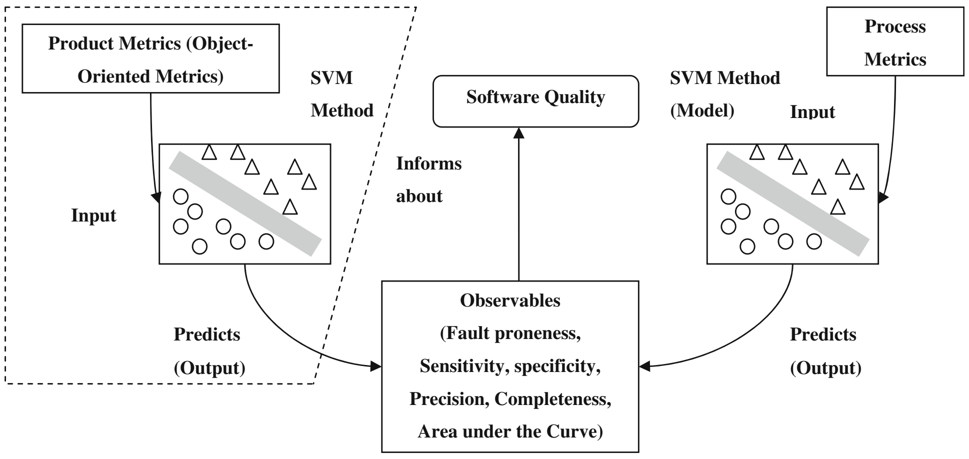
Product Metrics (Object-Oriented Metrics)
SVM Method
Input
Predicts (Output)
Software Quality
Informs about
Observables (Fault proneness, Sensitivity, specificity, Precision, Completeness, Area under the Curve)
SVM Method (Model)
Process Metrics
Input
Predicts (Output)

Fig. 1 Software quality assessment framework

total of 180 classes. They used univariate and multivariate analysis to find individual and combined impact of OO metrics and fault proneness. The results showed all metrics except NOC (which was found related to fault proneness in an inverse manner) to be significant predictor of fault proneness. Another study by (Briand et al. 2001) used a commercial system consisting of 83 classes. It found DIT metric related to fault proneness in inverse manner and NOC metric to be insignificant predictor of fault proneness.

Yu et al. 2002 examined the relationship between OO metrics and the fault-proneness. Gyimothy et al. (2005) empirically validated (Chidamber and Kamerer 1994) metrics on open source software for fault prediction. They employed regression (linear and logistic regression) and machine learning methods (neural network and decision tree) for the model prediction. The results indicated that NOC was not significant predictor of fault proneness but all the other metrics were found significant in LR analysis.

Olague et al. (2007) validated OO metrics on versions of an open source agile software. They found WMC, CBO, RFC and LCOM metrics to be very significant, while DIT was found insignificant in two versions of system. NOC metric was found to be insignificant in one version while less significant in other two versions.

Yuming and Hareton (2006) validated the same data set as ours to predict the fault proneness models with respect to two categories of faults: high and low. Pai (2007) also used the same data set using a Bayesian approach to predict the fault proneness models. In our previous work we validated this data set using statistical, artificial neural network and decision tree methods (Singh et al. 2010). Our approach in this work differs as we use SVM method for predicting fault proneness models and evaluate the performance of the predicted models using ROC analysis. We also use three categories of faults: high, medium, and low severity of faults.

3 Research background

In this section, we present the summary of metrics studied in this paper (Sect. 3.1), and hypotheses to be tested in our work (Sect. 3.2) and the data collected for this study (Sect. 3.3).

3.1 Dependent and independent variables

The binary dependent variable in our study is fault proneness. There have been lots of metrics proposed in the literature (Aggarwal et al. 2005; Briand et al. 1998; 1999; Bieman and Kang 1995; Cartwright and Shepperd 1999; Chidamber and Kamerer 1991; Chidamber and Kamerer 1994; Harrison et al. 1998; Henderson-Sellers 1996; Hitz and Montazeri 1995; Lake and Cook 1994; Li and Henry 1993; Lee et al. 1995; Lorenz and Kidd 1994; Tegarden et al. 1995). The goal of our study is to explore empirically the relationship between OO metrics and fault proneness at the class level. We use machine learning method SVM 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 (Chidamber and Kamerer 1994) are summarized in Table 1. These metrics are explained with practical applications in (Aggarwal et al. 2005).

3.2 Hypotheses

In this section, research hypotheses are presented. We tested the hypotheses given below to find the individual effect of each OO metric on fault proneness.

RFC Hypothesis: A class with high number of external and internal methods is more likely to be fault-prone than a class with low number of external and internal methods. (Null hypothesis: A class with high number of external and internal methods is less likely to be fault-prone than a class with low number of external and internal methods).

Table 1 Metrics studied (Chidamber and Kamerer 1994)

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.

CBO Hypothesis: A class with high import or export coupling is more likely to be fault-prone than a class with less import or export coupling. (Null hypothesis: A class with high import or export coupling is less likely to be fault-prone than a class with less import or export coupling).

LCOM Hypothesis: A class with less cohesion is more likely to be fault-prone than a class with high cohesion. (Null hypothesis: A class with less cohesion is less likely to be fault-prone than a class with high cohesion).

DIT Hypothesis: A class with more depth in inheritance tree is more likely to be fault-prone than a class with less depth in inheritance tree. (Null hypothesis: A class with more depth in inheritance tree is less likely to be fault-prone than a class with less depth in inheritance tree).

NOC Hypothesis: A class with a greater number of descendants is more likely to be fault-prone than a class with lesser number of descendants. (Null hypothesis: A class with more number of descendants is less likely to be fault-prone than a class with lesser number of descendants).

SLOC Hypothesis: A class with a larger size i.e., more information is more likely to be fault-prone than a class with a smaller size. (Null hypothesis: A class with a larger size i.e., more information is less likely to be fault-prone than a class with a smaller size).

WMC Hypothesis: A class with more number of methods weighted by complexities is more likely to be fault-prone than a class with less number of methods weighted by complexities. (Null hypothesis: A class with more number of methods weighted by complexities is less likely to be fault-prone than a class with less number of methods weighted by complexities).

3.3 Empirical data collection

This study makes use of the public domain data set KC1 from the NASA Metrics Data Program (NASA 2004). 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 40 K 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 7 metrics given by (Chidamber and Kamerer 1994). These 7 OO metrics are taken in our study (see Sect. 3.1) for analyses. In KC1 four files are of particular interest, one representing the association between classes and methods, representing the association between classes and defects, representing the association between defects and severity of faults and other representing the association between defects and specific reason for closure of error report.

We first associated defects with each class according to their severities. The value of severity quantifies the impact of the defect on the overall environment with 1 being most severe to 5 being least severe. An error could be either from source code, COTS/OS, design or is actually not a bug. We take into account defects produced from source code, COTS/OS and design. We further processed the data by removing all the faults that were not bugs. This reduced the number of faults from 669 to 642. Out of 145 classes, 59 were faulty classes and the rest were non-faulty.

In this study we categorized the faults as high, medium or low severity. Faults with severity rating 1 were classified as high severity faults. Faults with severity rating 2 were classified as medium severity faults and faults with severity rating 3, 4, 5 as low severity faults. Faults at severity rating 1 require immediate correction in order for the system to continue to operate properly (Yuming and Hareton 2006). This gave us 42 high severity faults (6.452 percent), 449 medium severity faults (69.93 percent) and 151 low severity

Table 2 Distribution of Faults

Level of severity	Number of classes	Number of faults associated	% of faults covered
1	11	1	1.71
	8	2	2.49
	4	3–5	2.34
2	6	1	0.93
	13	2	4.04
	9	3–4	4.67
	16	5–10	19
	12	11–21	24.92
	1	28	4.36
	1	77	12
3	8	1	1.25
	11	2	3.43
	11	3–5	6.23
	9	7–12	12.61

faults (23.52 percent). Table 2 shows the distribution of faults at each severity rating in the KC1 NASA data set after preprocessing of faults in the data set. High severity faults were distributed in 23 classes (38.98 percent).

In Fig. 2, column 1 of the bar graph shows the percentage of faulty classes at each severity rating and column 2 shows the number of faults covered by these classes. As shown in Fig. 2, majority of the classes are faulty at severity rating 2 (58 out of 59 faulty classes). In addition, maximum numbers of faults 449 out of 669 are covered at severity rating 2. At severity rating four number class was found to be faulty and at severity rating five only one class is faulty. Table 2 summarizes the distribution of faults at each severity level defined in Table 2.

4 Research methodology

In this section the steps taken to analyze coupling, cohesion, inheritance and size metrics for classes taken for analysis are described. The procedure used to analyze the data collected for each measure is described in following stages (i) data statistics and outlier analysis (ii) Support Vector Machine (SVM) Method (iii) model evaluation.

4.1 Descriptive statistics and outlier analysis

The role of statistics is to function as a tool in analyzing research data and drawing conclusions from it. The research data must be suitably reduced so that the same can be read easily and can be used for further analysis. Descriptive statistics concern development of certain indices or measures to summarize data. The important statistics measures used for comparing different case studies include mean, median, and standard deviation. Data points, which are located in an empty part of the sample space, are called outliers. Outlier analysis is done to find data points that are over influential and removing them is essential. Univariate and multivariate outliers are found in our study. To identify the multivariate outliers, we calculate for each data point the Mahalanobis Jackknife distance. Mahalanobis Jackknife is a measure of the distance in multidimensional space of each observation from the mean center of the observations (Aggarwal et al. 2009; Hair et al. 2006).

The influence of univariate and multivariate outliers was tested. If by removing a univariate outlier the significance of metric changes i.e., the effect of that metric on fault proneness changes then the outlier is to be removed. Similarly, if the significance of one or more independent variables in the model depends on the presence or absence of the outlier, then that outlier is to be removed. Details on outlier analysis can be found in (Belsley et al. 1980; Barnett and Price 1995).

4.2 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 (Wang et al. 2007), pattern recognition (Burges 1998), chinese character classification (Zhao and Takagi 2007), and identification of organisms (Morris et al. 2001). SVM constructs an N-dimensional hyperplane that optimally separates the data set into two categories. The purpose of the 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 (Sherrod 2003). The support vectors are the vectors near the hyperplane. The SVM modeling finds

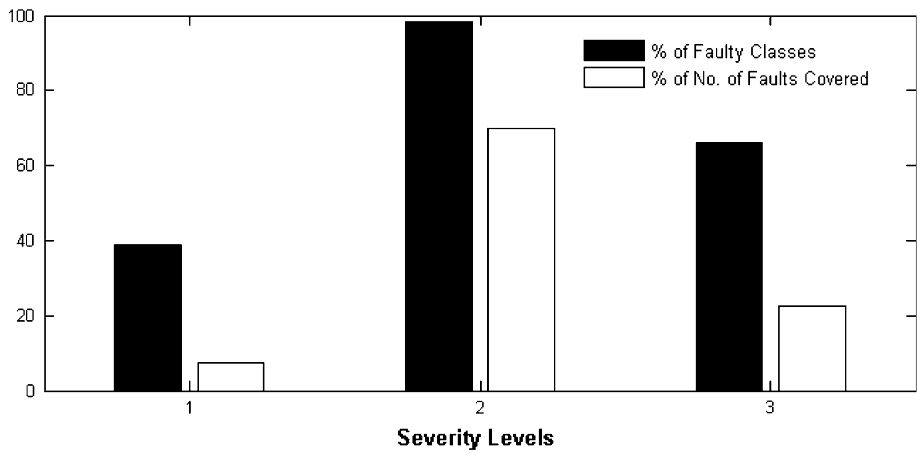

Fig. 2 Fault distribution at various severity levels

the hyperplane 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 in order 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 (Cristianini and Shawe-Taylor 2000; Cortes and Vapnik 1995).

The recommended kernel function is the Radial basis Function (RBF) (Sherrod 2003). Thus, we used RBF function in the 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 3 shows the RBF kernel. One category of the dependent variable is shown as rectangles and the other as circles. The shaded circles and rectangles are support vectors.

Given a set of (xi, yi),……., (xm, ym) and $yi \in \{-1, +1\}$training samples. $\alpha i = (1,\ldots.., \mathrm{m})$ 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 (Zhao and Takagi 2007):

$$D(x) = \sum_{i=1}^{m} yi\alpha iK(xi, x) + b \tag{1}$$

Then an input pattern x is classified as (Zhao and Takagi 2007):

$$x = \left\{ \begin{array}{c} +1 \text{ if } D(x) > 0 \\ -1 \text{ if } D(x) < 0 \end{array} \right\} \tag{2}$$

4.3 Evaluating the performance of the model

The common measures to assess the quality of the predicted model in our study are:

- The sensitivity and specificity of the model is calculated to predict the correctness of the model. Sensitivity of the model is defined as the ratio of classes predicted to be fault prone and total number of classes. The percentage of non-occurrences correctly predicted i.e. classes predicted not to be fault prone is called specificity of the model. Ideally, both the sensitivity and specificity should be high.
- *Completeness:* Completeness is defined as the number of faults in classes classified fault-prone, divided by the total number of faults in the system.
- *Precision*: Precision is defined as the number of classes that are predicted correctly, divided by the total number of classes.
- *Receiver operating characteristic (ROC) analysis*: The SVM model outputs were evaluated for performance using ROC analysis. 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 predicted models (Hanley and McNeil 1982). While constructing ROC curves, one selects many cutoff points between 0 and 1 in our case, and calculates sensitivity and specificity at each cut off point. The optimal choice of cutoff point (that maximizes both sensitivity and specificity) can be selected from the ROC curve (El Emam et al. 1999). Hence, by using ROC curve one can

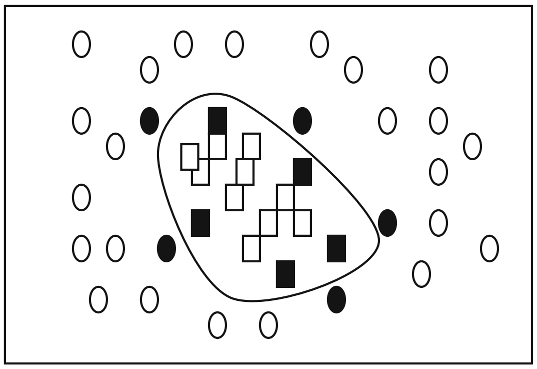

Fig. 3 Radial basis function

easily determine optimal cutoff point for an predicted model.

- Area Under the ROC Curve (AUC) is a combined measure of sensitivity and specificity. In order to compute the accuracy of the predicted models, we use the area under ROC curve.
- In order to predict the accuracy of the model it should be applied on different data sets. We therefore performed k-cross validation of the models (Stone 1974). The data set is randomly divided into *k* subsets. Each time one of the *k* subsets is used as the test set and the other *k*-1 subsets are used to form a training set. Therefore, we get the fault proneness for all the *k* classes.

5 Analysis results

This section presents the analysis results, following the procedure described in Sect. 4.1 and 4.3. Descriptive statistics and correlation to size (Sect. 5.1), univariate analysis and multivariate analysis (Sect. 5.2), results are presented.

5.1 Descriptive statistics and correlation analysis

Each table presented in the following subsections show “min”, “max”, “mean”, “std dev”, “25% quartile” and “75% quartile” for all metrics considered in this study.

The following observations are made from Table 3:

- The size of a class measured in terms of lines of source code ranges from 0 to 2313.
- The values of DIT and NOC are low in the system, which shows that inheritance is not much used in all the systems; similar results have also been shown by other (Chidamber et al. 1998; Briand et al. 2000; Cartwright and Shepperd 1999).
- The LCOM measure, which counts the number of classes with no attribute usage in common, has high values (up to 100) in KC1 data set.

Table 3 Descriptive Statistics for metrics

Metric	Min.	Max.	Mean	SD	Percentile (25%)	Percentile (75%)
CBO	0	24	8.32	8	3	14
LCOM	0	100	68.72	84	56.5	96
NOC	0	5	0.21	0	0	0
RFC	0	222	34.38	28	10	44.5
WMC	0	100	17.42	12	8	22
LOC	0	2313	211.25	108	8	235.5
DIT	0	6	1	1	0	1.5

We calculated the correlation among metrics, which is an important static quantity. as shown in Table 4. Gyimothy et al. (2005) and Basili et al. (1996) also calculated the correlation among metrics. In this data set, LOC, LCOM, DIT metrics are correlated with RFC metric. Similarly, WMC and CBO metrics are correlated with LOC metric. Therefore, it shows that these metrics are not totally independent and represents redundant information.

5.2 Univariate and multivariate analysis results

The results of univariate and multivariate analysis are presented with respect to high severity, medium severity, low severity and ungraded severity faults.

High Severity Fault (HSF) prediction model: We do not wish to select an arbitrary cut point. Thus in order to obtain a balance between the number of classes predicted as fault prone and not fault prone, the cutoff point of the model is computed using ROC analysis method (Hanley and McNeil 1982). Table 5 shows the sensitivity, specificity, precision, completeness and cutoff point of each metric and the model predicted. SLOC metric has the highest sensitivity (56.52 percent) but less completeness (45.8 percent) values. RFC metric have the next highest value of sensitivity (52.17 percent). NOC and DIT metrics predicted all classes to be non faulty.

The model was applied to 145 classes and Table 7(a) presents the results of accuracy of the fault proneness model predicted using HSF. Out of 23 classes, actually fault prone, 16 classes were predicted to be fault prone. As shown in Table 5, the cut off point for the model is 0.15. The sensitivity of the model is 69.5%. Similarly, 84 out of 122 classes were predicted not to be fault prone. Thus, specificity of the model is 70.49%. The completeness of the model is 68.75%. The model predicted has low sensitivity and completeness.

Medium severity fault (MSF) prediction model: Table 6 shows the sensitivity, specificity, precision, completeness and cutoff point of each metric and the model predicted. As shown in CBO metric has sensitivity 75.8 percent and completeness 83.74 percent. The completeness of SLOC

Table 4 Correlations among metrics

Metric	CBO	LCOM	NOC	RFC	WMC	LOC	DIT
CBO	1						
LCOM	0.256	1					
NOC	−0.03	−0.028	1				
RFC	0.386	0.334	−0.049	1			
WMC	0.245	0.318	0.035	0.628	1		
LOC	0.572	0.238	-0.039	0.508	0.624	1	
DIT	0.4692	0.256	-0.031	0.654	0.136	0.345	1

Table 5 Sensitivity, specificity, precision and completeness for HSF

Metric	Sensitivity	Specificity	Precision	Completeness	Cutoff
CBO	47.82	72.9	68.96	56.25	0.15
WMC	47.8	34.42	36.55	33.33	0.15
RFC	52.17	59.83	58.68	50	0.15
SLOC	56.52	53.27	53.79	45.8	0.15
LCOM	43.47	48.36	47.58	31.25	0.15
NOC	–	–	–	–	–
DIT	–	–	–	–	–
Model I	69.56	70.49	70.34	68.75	0.15

Table 6 Sensitivity, specificity, precision and completeness for MSF

Metric	Sensitivity	Specificity	Precision	Completeness	Cutoff
CBO	75.8	68.23	71.72	83.74	0.40
WMC	56.8	64.36	55.86	70.6	0.35
RFC	55.7	63.5	60.68	68.8	0.36
SLOC	63.79	75.86	55.86	75.94	0.44
LCOM	43.1	47.12	43.44	55.45	0.41
NOC	–	–	–	–	–
DIT	81.03	19.5	43.81	72.6	0.4
Model II	81.03	88.5	82.75	85.96	0.45

(75.94 percent) is also high. However, in case of NOC metric all the classes were predicted to be non faulty hence the results are not shown. The sensitivity of DIT metric is high but the precision (43.81 percent) is very low. LCOM metric also show less values of sensitivity (43.1 percent), specificity (47.12 percent), precision (43.44 percent) and completeness (55.45 percent).

In Table 6, the cut off point for the model build to predict fault proneness is 0.45. The model was applied to 145 classes and Table 7(b) presents the results of correctness of the fault proneness model predicted using MSF. Out of 58 classes, actually fault prone, 47 classes were predicted to be fault prone. The sensitivity of the model is 81.03 percent. Similarly, 77 out of 87 classes were predicted not to be fault prone. Thus, specificity of the model is 88.5 percent. The completeness of the model is 85.96 percent. Thus, the accuracy of the model is very high as compared to model predicted with regard to HSF.

Low severity fault (LSF) prediction model: Table 8 shows the sensitivity, specificity, precision, completeness and cutoff point of each metric and the model predicted. The sensitivity (74.35 percent) of CBO metric, specificity (74.35 percent), and completeness (79.86 percent) are high. The sensitivity (64.1 percent), precision (71.72 percent) and completeness (77.7 percent) values of SLOC metric are high. NOC and LCOM metric predicts all classes as non faulty. DIT metric show very poor sensitivity (15.4 percent), although the precision value is 72.8 percent. WMC metric also has poor specificity (31.13 percent), precision (37.25 percent), and completeness (37.5 percent).

Table 7 Predicted Correctness of Model **a** HSF, **b** MSF

Predicted		
(a)		
	Po $\leq$ 0.15	Po > 0.15
Observed	Not faulty	Faulty
Not faulty	86	36
Faulty	7 (15)	16 (33)
(b)		
	Po $\leq$ 0.45	Po > 0.45
Observed	Not faulty	Faulty
Not faulty	77	10
Faulty	11 (63)	47 (386)

Table 8 Sensitivity, specificity, precision and completeness for LSF

Metric	Sensitivity	Specificity	Precision	Completeness	Cutoff
CBO	74.35	70.75	71.7	79.86	0.21
WMC	53.84	31.13	37.24	37.5	0.26
RFC	53.84	51.88	52.41	62.5	0.26
SLOC	64.1	74.52	71.72	77.7	0.22
LCOM	–	–	–	–	–
NOC	–	–	–	–	–
DIT	15.4	94	72.8	34	0.68
Model III	74.35	77.35	76.55	77.08	0.25

The model was applied to 145 classes and Table 10(a) presents the results of correctness of the fault proneness model predicted with respect to LSF. As shown in Table 8, the cut off point for the model build to predict fault proneness is 0.25. Out of 39 classes, actually fault prone, 29 classes were predicted to be fault prone. The sensitivity of the model is 74.35 percent. Similarly, 82 out of 106 classes were predicted not to be fault prone. Thus, specificity of the model is 77.35 percent. The completeness of the model is 77.08 percent. Thus, the accuracy of the model is lower than the model predicted with respect to MSF but the accuracy is better than the fault proneness model predicted with respect to HSF.

Ungraded severity fault (USF) prediction model: Table 9 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 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.

Table 9 Sensitivity, specificity, precision and completeness for USF

Metric	Sensitivity	Specificity	Precision	Completeness	Cutoff
CBO	76.27	68.6	71.7	82.24	0.33
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	0.36
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	85.56	0.44

Table 10 Predicted Correctness of Model **a** LSF **b** USF

Predicted		
a		
	Po ≤ 0.25	Po > 0.25
Observed	Not faulty	Faulty
Not faulty	82	24
Faulty	10 (33)	29 (111)
b		
	Po ≤ 0.43	Po > 0.43
Observed	Not faulty	Faulty
Not faulty	70	16
Faulty	14(92)	45(550)

The model was applied to 145 classes and Table 10(b) presents the results of correctness of the fault proneness model predicted with respect to LSF. As shown in Table 9, 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, specificity of the model is 81.39 percent. The completeness of the model is 85.56 percent. Thus, the accuracy of the model is very high. The sensitivity of the model predicted with regard to HSF is less as compared to the model predicted with regard to medium, low and ungraded severity of faults. The model predicted with regard to MSF shows better results as compared to the models predicted using HSF, LSF, and USF.

5.3 Discussion and the validation of hypothesis

In this section, we validate our hypothesis stated in Sect. 3.3. At the same time, we also compare our results with those of previous studies until date.

RFC hypothesis: The sensitivity and specificity values of RFC metric are high for all severities of faults. Similar results were shown by (Basili et al. 1996; Briand et al. 2000; El Emam et al. 2001, Yuming and Hareton 2006; Olague et al. 2007 and Gyimothy et al. 2005). Tang et al. (1999) found it significant at 0.05. Yu et al. (2002) also found RFC metric significant predictor but their method of calculating the RFC metric was different.

Hence, we rejected the null hypothesis and accepted the alternative hypothesis for RFC metric.

CBO hypothesis: The sensitivity and specificity values are high for all severities of faults. It was also found significant predictor in all studies except Tang et al. (1999) and El Emam et al. (2001) (#2).

All of our models found CBO metric to be significant predictor of fault proneness; hence, we rejected the null hypothesis and accepted the alternative hypothesis.

LCOM hypothesis: The sensitivity and specificity are low in our SVM analysis (except for faults predicted with respect to ungraded severity), contradicting the results of (Basili et al. 1996), where LCOM was shown to be insignificant. Yuming and Hareton 2006), Olague et al. (2007) and Gyimothy et al. (2005) also found LCOM metric to be very significant predictor of fault proneness. Yu et al. (2002) calculated LCOM in a totally different way, hence, we could not compare our results with theirs.

Hence, we accepted the null hypothesis and rejected the alternative hypothesis.

DIT hypothesis: The sensitivity and specificity are low in our SVM analysis. On the other hand (Briand et al. 2001) found it to be significant but in inverse manner. This finding is similar to those given by Yu et al. (2002). Basili et al. (1996) and Briand et al. (2000) found DIT metric to be significant predictor of fault proneness. Gyimothy et al. (2005) found DIT metric to be less significant predictor of fault proneness. Our DT results also showed very less sensitivity values for medium, low and ungraded severities of faults. For high severity of faults DT and ANN method predicted all classes as non faulty. ANN method showed low values of sensitivity for medium, low and ungraded severities of faults.

We thus accepted the null hypothesis for DIT metric and rejected the alternative hypothesis.

NOC hypothesis: The results of SVM predicted all classes to be non faulty for all severities of faults. Briand et al. (2001); Gyimothy et al. (2005) and Tang et al. (1999) found NOC not to be a significant predictor of fault proneness. Basili et al. (1996), Briand et al. (2000) and Yuming and Hareton (2006) found NOC metric to be significant but they found that the larger the value of NOC, the lower the probability of fault proneness. According to Yu et al. (2002), NOC metric was a significant predictor of fault proneness and they found that more the number of children in a class, the more fault prone it is.

We thus accepted the null hypothesis for NOC metric and rejected the alternative hypothesis.

Table 11 Summary of hypothesis

Metric	Hypothesis accepted			
Severity of faults	HSF	MSF	LSF	USF
RFC	√	√	√	√
CBO	√	√	√	√
LCOM	–	–	–	–
DIT	–	–	–	–
NOC	–	–	–	–
WMC	√	√	–	√
LOC	√	√	√	√

SLOC hypothesis: The sensitivity and specificity are high in our SVM analysis. It was also found significant in all the studies that examined them. Hence, we reject the null hypothesis and accept the alternative hypothesis.

WMC hypothesis: The sensitivity and specificity are high in our SVM analysis (except for low severity faults). On the other hand, Basili et al. (1996) found it less significant. In the study conducted by Yu et al. (2002), WMC metric was found to be a significant predictor of fault proneness. Similar to our regression and DT and ANN results, (Briand et al. 2000; Gyimothy et al. 2005; Olague et al. 2007; Yuming and Hareton 2006) also found WMC metric as one of the best predictors of fault proneness. Rest of the studies found it to be significant predictor but at 0.05 significance level.

All of our three models found WMC metric to be significant predictor of fault proneness; hence, we rejected the null hypothesis and accepted the alternative hypothesis.

There is an existing work in the literature, which empirically validates the same data set that we used in this paper, for assessing fault proneness (Yuming and Hareton 2006). We validate the metrics with SVM method and with different categories of faults. Thus, we did not repeat logistic regression analysis. Their study also finds WMC, CBO, RFC, LCOM, and SLOC to be very significant for fault proneness analysis, whereas DIT was not significant. In the case of NOC, however, their results were inconclusive.

Our results were similar as theirs with respect to high and low severity of faults. However, LCOM metric was not found significant at all severity levels. WMC metric also showed less value of specificity and completeness with regard to LSF.

In Table 11 we summarize the results of the hypothesis stated in Sect. 3.2 with respect of each severity of faults.

6 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 following the procedure given in Sect. 4.3. For the 10-cross validation, the classes were randomly divided into 10 parts of approximately equal (14 partitions of 10 data points each and 1 partition of 5 data points each). We summarized the results of cross validation of predicted models via the SVM approach in Table 12. In Table 13, Model I is predicted with respect to HSF, Model II is predicted with respect to MSF, Model III is predicted with respect to LSF and Model IV is predicted with respect to USF.

Table 13 summarizes the results of 10-cross validation of the models using SVM method. The AUC of the model predicted with regard to HSF is 0.728, which is, less than the AUC of the model predicted using MSF (0.88), LSF (0.84) and USF (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 models are good. Therefore, it appears that the model predicted using SVM might lead to development of optimum software quality models for predicting fault prone classes.

The results obtained have higher precision and completeness as compared to the results obtained from the model predicted using the logistic regression, random

Table 12 Predicted Correctness of Model **a** HSF **b** MSF **c** LSF **d** USF

Predicted		
a		
	$Po \leq 0.16$	$Po > 0.16$
Observed	Not faulty	Faulty
Not faulty	84	38
Faulty	10 (15)	13 (33)
b		
	$Po \leq 0.41$	$Po > 0.41$
Observed	Not faulty	Faulty
Not faulty	71	16
Faulty	17 (88)	41 (361)
c		
	$Po \leq 0.45$	$Po > 0.45$
Observed	Not faulty	Faulty
Not faulty	71	15
Faulty	18(131)	41(511)
d		
	$Po \leq 0.26$	$Po > 0.26$
Observed	Not faulty	Faulty
Not faulty	73	33
Faulty	10 (33)	29 (111)

Table 13 Results of 10-cross validation of models

Support vector machine

	Model I	Model III	Model III	Model IV
Cutoff point	0.16	0.45	0.26	0.45
Sensitivity	56.52	70.68	74.35	69.49
Specificity	68.85	81.6	68.86	82.55
Precision	66.89	77.30	70.34	77.24
Completeness	68.75	80.4	77.08	79.59
AUC	0.728	0.88	0.84	0.89

forest, NNage and Naïve bayes (Yuming and Hareton 2006). We have evaluated artificial neural network and decision tree in our previous work (Singh et al. 2010). The results obtained (AUC) by using SVM method are better as compared to artificial neural network method and comparable to decision tree method.

7 Model application

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 Figs. 4, 5, 6, and 7. The values for the predicted fault proneness were taken from the results of validation of the models. On X-axis, we plot 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 sensitivity and completeness of the predicted model, respectively. For example, in Fig. 4 at $X = 0$, the sensitivity of the model is 100% and completeness is 100%.

While planning testing during the software development process, we would like to cover maximum faults with the planned number of resources available to assure quality software. To optimize the use of available resources, we can use the chart in Figs. 4, 5, 6, and 7 to determine the classes that should be tested on high priority basis to cover maximum faults in the code. For example, if we have resources available to inspect 33% of the code, from Fig. 7 we can tell that 19.4% of classes with the highest predicted fault proneness (at $X = 1$) only need to be tested. If we select these classes for testing we can expect maximum faults to be covered.

8 Threats to validity

In the KC1 data set the fault severity rating are subjective and may be inaccurate. Thus, the generalizability of the results is possibly limited.

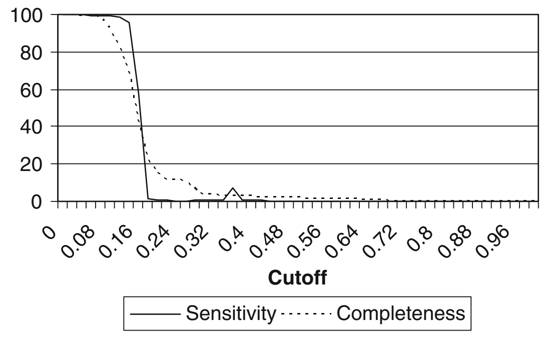

Fig. 4 Sensitivity and completeness of model I

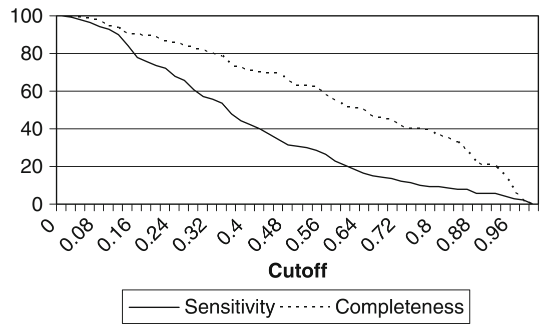

Fig. 5 Sensitivity and completeness of Model II

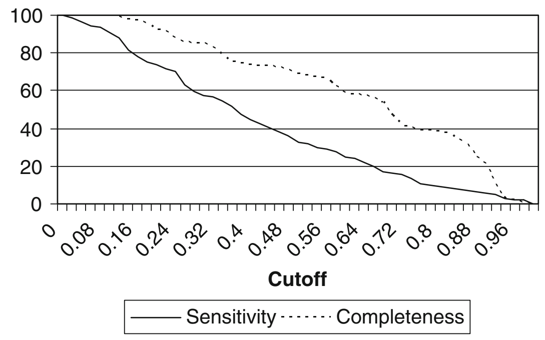

Fig. 6 Sensitivity and completeness of model III

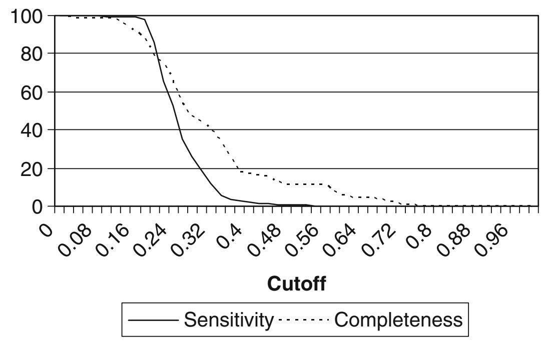

Fig. 7 Sensitivity and completeness of model IV

The usefulness of OO metrics for predicting the fault proneness models also depend on programming language (e.g. Java) being used. Thus, similar studies with different data sets are required to be carried out in order to establish the acceptability of the model.

Our conclusions are pertinent to only dependent variable fault proneness, as it seems to be most popular dependent variable in empirical studies. We do not claim about the validity of the chosen OO metrics in this study when the dependent variable changes like maintainability or effort (Singh et al. 2010).

While these results provide guidance for future research on the impact of metrics on fault proneness at different severity levels, further validations are needed with different systems to draw stronger conclusions.

9 Conclusions and future work

The goal of our research is to empirically analyze the performance of SVM method. We find the individual and combined effect of each metric on fault proneness of classes taking into account severity of faults. The faults were categorized into three categories: high, medium and low severity faults.

Based on public domain NASA data set KC1, we empirically analyzed the performance of SVM method using ROC analysis. The contributions of this paper are summarized as follows: First, we performed the analysis of public domain NASA data set (NASA Data Repository), 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. Third, we take severity of faults into account while predicting fault proneness of the classes.

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 across all severity of faults, LCOM metric less sensitivity and completeness with respect to all severities except USF. NOC and DIT metric was not found to be significantly related to fault proneness across any severity of faults. Thus, the results of machine learning method (SVM) show that usefulness of NOC and DIT metric is poor.
- The model predicted with respect to high severity faults has lower accuracy than other models predicted at medium and low severities. The fault proneness models predicted with respect to medium severity faults showed the best result using all the methods.
- 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 SVM method 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.”

More studies that are similar must be carried out with different data sets to give generalized results across different organizations. 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 the models that will help to determine whether a given the fault proneness model would be economically viable.

References

Aggarwal KK, Singh Y, Kaur A, Malhotra R (2005) Software reuse metrics for object-oriented systems. In: Proceedings of the 3rd ACIS international conference on software engineering research, management and applications (SERA ‘05), Mt. Pleasant, pp 48–55

Aggarwal KK, Singh Y, Kaur A, Malhotra R (2006a) Empirical study of object-oriented metrics. J Object Technol 5(8):149–173

Aggarwal KK, Singh Y, Kaur A, Malhotra R (2006b) Investigating the effect of coupling metrics on fault proneness in object-oriented systems. Software Qual Prof 8(4):4–16

Aggarwal KK, Singh Y, Kaur A, Malhotra R (2009) Empirical analysis for investigating the effect of object-oriented metrics on fault proneness: a replicated case study. Softw Process Improv Pract 14(1):39–62

Barnett V, Price T (1995) Outliers in statistical data. Wiley, NewYork

Basili V, Briand L, Melo W (1996) A validation of object-oriented design metrics as quality indicators. IEEE Trans Softw Eng 22(10):751–761

Belsley D, Kuh E, Welsch R (1980) Regression diagnostics: identifying influential data and sources of collinearity. Wiley, New York

Bieman J. Kang B. (1995). Cohesion and reuse in an object-oriented system. Proc ACM Symp Softw Reusability (SSR’94) 259–262

Binkley A, Schach S (1998). Validation of the coupling dependency metric as a risk predictor. In: Proceedings of the international conference on software engineering, Kyoto, pp 452–455

Briand L, Daly W, Wust J (1998) Unified framework for cohesion measurement in object-oriented systems. Empir Softw Eng 3(1):65–117

Briand L, Daly W, Wust J (1999) A unified framework for coupling measurement in object-oriented systems. IEEE Trans Softw Eng 25(1):91–121

Briand L, Daly W, Wust J (2000) Exploring the relationships between design measures and software quality. J Syst Softw 51(3):245–273
Briand L, Wüst J, Lounis H (2001) Replicated case studies for investigating quality factors in object-oriented designs. Empir Softw Eng 6(1):11–58
Burges C (1998) A tutorial on support vector machines for pattern recognition. Data Min Knowl Disc 2:121–167
Cartwright M, Shepperd M (1999) An empirical investigation of an object-oriented software system. IEEE Trans Softw Eng 26(8): 786–796
Chidamber S, Kamerer C (1991) Towards a metrics suite for object oriented design. In: Proceedings of the conference on object-oriented programming: systems, languages and applications (OOPSLA'91). SIGPLAN Notices 26(11):197–211
Chidamber S, Kamerer C (1994) A metrics suite for object-oriented design. IEEE Trans Softw Eng 20(6):476–493
Chidamber S, Darcy D, Kemerer C (1998) Managerial use of metrics for object-oriented software: an exploratory analysis. IEEE Trans Softw Eng 24(8):629–639
Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20:273–297
Cristianini N, Shawe-Taylor J (2000) An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, Cambridge
El Emam K, Benlarbi S, Goel N, Rai S (1999) A validation of object-oriented metrics. Tech Rep. ERB-1063, NRC
El Emam K, Benlarbi S, Goel N, Rai S (2001) The confounding effect of class size on the validity of object-oriented metrics. IEEE Trans Softw Eng 27(7):630–650
Gyimothy T, Ferenc R, Siket I (2005) Empirical validation of object-oriented metrics on open source software for fault prediction. IEEE Trans Softw Eng 31(10):897–910
Hair J, Anderson R, Tatham W (2006) Black multivariate data analysis. Pearson Education, Upper Saddle River
Hanley J, McNeil BJ (1982) The meaning and use of the area under a receiver operating characteristic ROC curve. Radiology 143: 29–36
Harrison R, Counsell SJ, Nithi RV (1998) An evaluation of MOOD set of object-oriented software metrics. IEEE Trans Softw Eng 24(6):491–496
Henderson-Sellers B (1996) Object-oriented metrics, measures of complexity. Prentice Hall, Englewood Cliffs
Hitz M, Montazeri B (1995) Measuring coupling and cohesion in object-oriented systems. In: Proceedings of the international symposium on applied corporate computing, Monterrey, pp 25–27
Lake A, 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
Lee Y, Liang B, Wu S, 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
Li W, Henry S (1993) Object-oriented metrics that predict maintainability. J Syst Softw 23(2):111–122
Lorenz M, Kidd J (1994) Object-oriented software metrics. Prentice-Hall, Englewood Cliffs
Morris C, Autret A, Boddy L (2001) Support vector machines for identifying organisms-a comparison with strongly partitioned radial basis function networks. Ecol Model 146:57–67
Olague H, Etzkorn L, Gholston S, 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 Trans Softw Eng 33(8):402–419
Pai G (2007) Empirical analysis of software fault content and fault proneness using Bayesian methods. IEEE Trans Softw Eng 33(10):675–686
Sherrod P. (2003) DTreg predictive modeling software
Singh Y, Kaur A, Malhotra R (2010) Empirical validation of object-oriented metrics for predicting fault proneness models. Softw Qual J 18(1):3–35
Stone M (1974) Cross-validatory choice and assessment of statistical predictions. J Royal Stat Soc 36:111–147
Tang MH, Kao MH, Chen MH (1999) An empirical study on object-oriented metrics. In: Proceedings of 6th international software metrics symposium, pp 242–249
Tegarden D, Sheetz S, Monarchi D (1995) A software complexity model of object-oriented systems. Decision Support Syst 13(3–4):241–262
Wang X, Bi D, Wang S (2007) Fault recognition with labelled multi-category', Third conference on Natural Computation. Haikou, China
NASA metrics data repository (2004). Available at www.mdp.ivv.nasa.gov
Yu P, Systa T, Muller H (2002) Predicting fault-proneness using OO metrics: an industrial case study. In: Proceedings of sixth European conference on software maintenance and reengineering, Budapest, Hungary, pp 99–107
Yuming Z, Hareton L (2006) Empirical analysis of Object-Oriented Design Metrics for predicting high severity faults. IEEE Transactions on Software Engineering 32(10):771–784
Zhao L, Takagi N (2007) An application of Support vector machines to Chinese character classification problem. In: IEEE international conference on systems, Man and Cybernetics, Montreal