On the Applicability of Machine Learning Techniques for Object Oriented Software Fault Prediction

Ruchika Malhotra[1], Yogesh Singh[2]
[1]*Department of Software Engineering, Delhi Technological University, Delhi, India*
ruchikamalhotra2004@yahoo.com

[2]*Maharaja Sayajirao University of Baroda, Baroda, India*
ys66rediffmail.com

Abstract— **Software testing is a critical and essential part of software development that consumes maximum resources and effort. The construction of models to predict faulty classes can help and guide the testing community in predicting faulty classes in early phases of software development. It is important to analyze and compare the predictive accuracy of machine learning classifiers. The aim of this paper is to find the relation of object oriented metrics and fault proneness of a class. We have used seven machine learning and one logistic regression method in order to predict faulty classes. The results of our work are based on data set obtained from open source software. The results show that the predictive accuracy of machine learning technique LogitBoost is highest with AUC of 0.806.**

Keywords— **object oriented metrics, software testing, machine learning techniques, Fault prediction, Receiver Operating Characteristics analysis, open source software.**

1. Introduction

Software testing consumes at least 50% of the resources and still does not guarantee the 100% correctness of the software. The obvious question is "why cannot we guarantee 100% correctness?" There are many reasons for not achieving the objective of "100% correctness of software" even with latest available technologies, tools and techniques. The ultimate target of developing defect free software opens many challenges, directions and scope for research which may ultimately improves the practices prevailing in the software testing community. New research challenges are emerging due to increasing size and complexity of the software. Many earlier concepts and techniques may not be relevant to meet the requirements of software testing community. The most important challenge is to make testing effective with reasonable consumption of resources. The challenge of effective testing opens research area of predicting faulty classes in the early phases of software development. These fault prediction models will help practitioners and testers in producing defect free software at reasonable cost using object oriented metrics for predicting faulty classes.

A number of machine learning techniques are available which may be used to predict faulty classes. These techniques have the following advantages:

- They can be used to model complex relationships.
- They can be used to adapt to changing environment as new knowledge is being discovered [6].

We may wonder: Which machine learning technique performs the best? How do machine learning techniques perform on different data sets? The previous studies in the literature may have reached to different conclusions as their results are based on different data sets [49]. Now a days data collected from public domain and open source software are available from data repositories such as NASA and PROMISE. This enables the researchers to carry out empirical studies in order to assess and compare various machine learning techniques. More empirical studies with different data set will highlight the criticalities, complexities and differences along with their strengths and weaknesses. This is the main motivation of reported work here.

The selection of subset of object oriented metrics is also important to obtain better machine learning models for fault prediction. Thus, the aims of this paper are 1) to find the relation between object oriented metrics and fault proneness of a class 2) to find subset of object oriented metrics that are effective predictors of fault proneness, and 3) assess and compare seven machine learning techniques: Artificial Neural Network (ANN), Random Forest (RF), two Boosting algorithms (LogitBoost (LB), AdaBoost (AB)), Naïve Bayes (NB), KStar, Bagging and one statistical technique: logistic regression (LR) in order to gain insight about their performance. In order to achieve above goals we validate our results on open source software Arc [54]. We use Area Under Curve (AUC) obtained from receiver operating characteristics curves (ROC) in order to evaluate the performance of the machine learning classification techniques. ROC analysis provides optimal cut off point that provides balance

between number of classes predicted as faulty and number of classes predicted as non faulty.

The developed machine learning models will enable the testers in the early stages of software development to identify which classes are more prone to faults and need extra attention. For example, testers will only concentrate on a subset of the classes which are predicted to be faulty by the developed models. This study will also guide the testing community on the predictive accuracy of machine learning predictors for predicting faulty classes.

The rest of the paper is organized as follows: Related work is summarized in section 2. Section 3 provides research methodology including the independent and dependent variables used in this work. Section 4 provides an overview of machine learning techniques and performance evaluation measures. Description of the results can be found in section 5. Section 6 provides threats to the validity of this work and a conclusion is presented in section 7.

2. RELATED WORK

Traditional logistic regression technique for predicting faulty classes are widely researched in the past [1, 2, 4, 5, 8, 13, 14, 16, 19, 28, 33, 41, 42, 48, 51, 52, 55, 57, 58, 60, 61]. Several machine learning techniques have been studied to find a relation between static code metrics and fault proneness. The use of neural network for predicting software quality was introduced by Khoshgaftaar et al. [39]. They classified modules as faulty and non faulty using large telecommunication system. They also compared their results with another model obtained from discriminant method. Guo et al. [27] used RF method for predicting faulty modules using NASA data sets. Khoshgaftaar et al. [40] proposed the use of regression trees in order to classify modules as faulty and non faulty using large telecommunications system. Fenton et al. [24] introduced the use of Bayesian belief networks (BBN) for the prediction of faulty software modules. Elish et al. [23] proposed the use of support vector machines for predicting faulty modules using NASA data sets. Porter and Selly proposed the use of decision trees [53].

Although there have many studies in past that evaluate the relation between static code metrics and fault proneness, but their have been very few studies that find the effect of object oriented metrics on fault proneness using machine learning methods. Table 1 summarizes studies that have used machine learning methods: ANN, DT, BBN, RF, NNage for finding relation between object oriented metrics and fault proneness.

It is worth noting that only a few machine learning techniques have been used for predicting faulty classes using object oriented metrics in the literature. A number of research groups have used NASA data sets for validation of their results. It can be also seen that Chidamber and Kemerer metrics have been widely used for predicting faulty classes. In this work we use seven machine learning techniques: ANN, RF, LB, AB, NB, KStar, Bagging and one statistical technique: LR. We use object oriented metrics [17] and validate our results on open source software Arc [54].

TABLE 1

EMPIRICAL STUDIES FOR PREDICTING FAULT PRONENESS USING MACHINE LEARNING TECHNIQUES

S. No	Source	Data type	Data set	Metrics	Technique used in model prediction
1	[28]	Open source	Mozilla	CK metric suite: WMC, DIT, RFC, NOC, CBO, LCOM, LCOMN, LOC	logistic regression, linear regression, decision tree, neural network
2	[60]	Public domain	NASA data set KC1 Implemented in C++. Consists of 145 classes, 2107 methods, 40,000 LOC	CK metric suite: WMC, DIT, RFC, NOC, CBO, LCOM, SLOC	logistic regression, machine learning methods (naïve Bayes network, random forest, NNge)
3	[42]	Students	System developed in C++. Consists of 1185 classes.	Total 64 metrics are used: 10 cohesion, 18 inheritance, 29 coupling and 7 size	logistic regression, back propagation neural network, probabilistic neural network
4	[52]	Public domain	NASA data set KC1 Implemented in C++. Consists of 145 classes, 43KLOC.	CK metric suite: WMC, RFC, NOC, CBO, LCOM, SLOC	multiple regression: ordinary least squares, bayesian linear regression, bayesian poisson regression
5	[55]	Public domain	NASA data set KC1 Implemented in C++. Consists of 145 classes, 43KLOC.	CK metric suite: CBO, RFC, LCOM, NOC, DIT, WMC, SLOC	logistic regression, machine learning (Artificial neural network, Decision Tree)
6	[61]	Open source	Data collected from 3 releases 2.0, 2.1 and 3.0 of Eclipse. Consists of 796, 988,1306 KLOC resp.	WMC, SDMC, AMC, CCMax, NIM, NCM, NTM, NLM, aVGloc, LOC	logistic regression

3. Research Background

In this section we define independent and dependent variables and also summarize the empirical data collection.

3.1 Independent and Dependent Variables

The independent variables in this paper are object oriented metrics and dependent variable is fault proneness. There have been several object oriented metrics proposed in the literature [4, 11, 12, 17, 18, 35, 36, 44-46, 48, 58]. We have used Chidamber and Kemerer [17, 18] and QMOOD metrics as independent variables, shown below in Table 2. Fault proneness is defined as probability of fault detection in a class [13, 55]. We have used seven machine learning techniques and logistic regression technique in order to predict probability of a class being faulty.

TABLE 2
OO METRICS – INDEPENDENT VARIABLES

Metric	**Definition**
Coupling Between Objects (CBO)	CBO for a class is a count of the number of other classes to which it is coupled and vice versa.
Lack of Cohesion Methods (LCOH)	Measures the dissimilarity of methods in a class by looking at the instance variable or attributes used by methods.
Number of Children (NOC)	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 sum of complexities of all methods in a class.
Response For a Class (RFC)	A set of methods that can be potentially executed in response to a message received by an object of that class.
Number of Public Methods (NPM)	The count of number of public methods in a class.
Afferent Couplings (Ca)	It counts how many other classes use a given class.
Lack of cohesion in methods (LCOM1).	$$LCOM1 = \frac{\frac{1}{N}\sum_{i=1}^{n}\mu(D_i) - m}{1-m}$$
Data Access Metric (DAM)	It is defined as number of private methods divided by total number of methods.
Measure of Aggregation (MOA)	It counts abstract data types in a class.
Measure of Functional Abstraction (MFA)	It is defined as number of inherited methods divided by total number of methods accessible by its member functions.
Cohesion Among Methods of Class (CAM)	It is based upon parameters list of a method.
Inheritance Coupling (IC)	It is based upon inheritance based coupling.
Coupling Between Methods (CBM)	It counts the newly added functions with which inherited based methods are coupled.
Average Method Complexity (AMC)	It counts average size of method in a class.
Cyclomatic Complexity (CC)	CC= e-n+P, where e= number of edges in a flow graph, n=number of nodes in a flow graph, P= connected components
Lines of Code (LOC)	The count of lines in the text of the source code excluding comment lines

3.2 Empirical Data Collection

In this work we validate the relation between object oriented metrics and fault proneness using open source data set Arc. The data set is available at [54]. The data set consist of 234 classes out of which 24 are faulty classes and rest are non faulty classes. The number of faults in the data set are 33. The metrics are collected using ckjm tool http://gromit.iiar.pwr.wroc.pl/p_inf/ckjm/.

4. RESEARCH TECHNIQUES

In this section, we summarize research techniques including logistic regression and machine learning techniques. The details on statistical and machine learning techniques can be obtained from [21, 22, 43].

4.1 Descriptive Statistics, Outlier Analysis and Correlation Analysis

The statistics measures such as mean, median, standard deviation collected from the given data set can be used to obtain further information about the data that will help the researchers in data filtering. Descriptive statistics provide summary about the data. The measures having less than ten data points are removed from the analysis in this work.

Outliers are the data points located in an empty space of the total data space [55]. The outliers numerically lie far away from rest of the data points [31]. Data can be summarized by using measures of central tendency (mean, median and mode) and measures of dispersion (standard deviation, variance, quartile).

There are two types of outliers: univariate and multivariate. Univariate outliers are those exceptional values that occur within a single variable and multivariate outliers are present within the combination of more than two variables. Once the outliers are identified the decision about the inclusion or exclusion of the outlier must be made. The decision depends upon the reason why the case is identified as outlier. In this paper univariate outliers are determined by calculating the Z-score value of the data points of the variable. The data values exceeding ±2.5 are considered to be outliers [29]. Multivariate outliers are identified using Mahalanobis Jackknife [29] distance. Details on outlier analysis can be obtained from [7].

Correlation analysis is used to provide information about dependency amongst the variables. In this work we find correlation between OO metrics to gather information about redundancy amongst object oriented metrics. We use Pearson's measure of correlation to test the correlation amongst object oriented metrics.

4.2 LR Analysis

LR is the most widely used statistical technique used in empirical studies in literature (a detailed description on LR analysis can be obtained from Hosmer et al. [37] and Basili et al. [8]). We can carry out univariate and multivariate LR analysis to find the relation between independent variables (object oriented metrics in our case) and dependent variable (fault proneness). We use binary LR analysis in this work as our dependent variable fault proneness is binary. Univariate analysis is used to find the individual effect of each independent variable on the dependent variable. The univariate LR formula is given below:

$$prob(X_1, X_2,.....X_n) = \frac{e^{(A_o + A_1 X)}}{1 + e^{(A_o + A_1 X)}} \quad (1)$$

Multivariate LR analysis is used to find the combined effect of independent variables on the dependent variable. Thus LR analysis is used to construct fault prediction model in this paper. Multivariate analysis generalized case of univariate analysis and the equation is given below:

$$prob(X_1, X_2,.....X_n) = \frac{e^{(A_o + A_1 X_1 + ... + A_n X_n)}}{1 + e^{(A_o + A_1 X_1 + ... + A_n X_n)}} \quad (2)$$

where $X_i, i = 1,2,....., n$, are the independent variables. Prob is the probability of detecting faults in a class. Two stepwise selection methods: forward selection and backward elimination can be used to predict dependent variable using LR analysis [2, 55]. In forward selection method each variable is entered stepwise and insignificant variables are removed at each step. In backward elimination method all the variables are entered together and variables are removed one at a time from the predicted model. We have used forward selection method using metrics selected in univariate analysis. We also attempted to use a backward elimination method. The results of both the methods were same. Hence, we used forward selection method for model prediction in this paper.

The following statistics are summarized for each object oriented metric found significant using LR analysis [2, 55]:

- **Odds Ratio**: It is defined as the probability of occurrence of an event divided by the probability of occurrence of non-event. The event in this paper is faulty class and nonevent is probability of not faulty class. An Odds Ratio with value 4

implies that dependent variable (fault proneness) is multiplied by 4 when the independent variable increases by 1 unit.

- **Coefficients (Ai's) and Maximum Likelihood Estimation (MLE):** coefficients in LR analysis are estimated by using MLE. The likelihood function (L) maximizes $Log(L)$ in order to compute the coefficients. Large value of coefficients implies that independent variables (object oriented metrics) have stronger relationship with the dependent variable (fault proneness).

- **Significance (sig):** the statistical significance implies that there is a significant relationship between independent variables and the dependent variable. In this paper we use 0.05 level as the significance threshold.

- **The R^2 Statistic:** the R^2 statistic is defined as:

$$R^2 = 1 - \frac{LL(a, B)}{LL(a)} \tag{3}$$

where *(LL(a, B)* is the maximum log-likelihood function that includes significant independent variables in the model. *LL(a)* is the log-likelihood function of the model with only constant included in it. The higher the value of this statistic more is the accuracy of the predicted model. However, unlike linear regression, in LR analysis this value is low for predicted model [60].

4.3 Machine Learning Techniques

In this section the machine learning techniques used for feature selection and model prediction are described.

Correlation Based Feature Selection

In order to predict models using machine learning techniques, it is important to identify relevant and important features. A relevant feature is one that is correlated to the class and is less related to other features [30]. In this paper we use Correlation based Feature Selection (CFS) method to determine relevant object oriented metrics and remove unwanted and noisy metrics. M.A Hall proved in his research paper that "classification accuracy using reduced feature set is equal or better than accuracy using complete feature set" [30]. CFS is based on correlation method and its advantages include recurred dimensionality, improved predictive accuracy and reduced execution time [30, 50].

Bagging

Bagging or Bootstrap Aggregating improves the performance of classification models by creating various sets of the training sets. Leo Breiman proposed this technique in 1996 [15]. The aim is to create numerous similar training sets and train a new function for each of these sets. In the case of class prediction the result of majority voting is considered. In order to construct multiple versions we create bootstrap duplicates of the learning set. These sets are then used as the new learning set. Bagging has the following advantages [10, 15]:

1. It can improve classification accuracy over other classification models.
2. It reduces variance.
3. It avoids overfitting.

If we are given a training set R_n, with N samples from a population P, a bootstrap set B_i would also contain N samples. For each sample of B_i we can draw any sample from R independently and with replacement from R. Hence we would obtain some samples being repeated in B_i and some samples being present in R but not in B_i. Hence the dataset B_i is as plausible as R_n but is drawn from R rather than P. The bootstrap datasets are combined by taking the average output and hence we get an aggregated prediction result [10, 15]. In this work we use Bag Size Percent of 100, 10 iterations and REP tree as classifier. These are the default setting provided by the WEKA tool.

Random Forests

RF was proposed by Breiman [24] and constructs a forest of multiple trees and each tree depends on the value of a random vector. For each of the tree in the forest, this random vector is sampled with the same distribution and independently. Hence, random forest is a classifier that consists of a number of decision trees. The resultant output class is the mode of classes output by the individual trees [10].

The algorithm for constructing the tree is as follows: For M training sets, N variables in the classifier and the variable n ($n << N$) where n indicates the number of independent variables that determine the decision at the terminal node of the tree. A bootstrap training sample is selected. The best split is based on these n variables in the training set. Each tree is allow to grow fully and is not pruned. RFs have the following benefits [10, 15, 47]:

1. RFs are simple.
2. RFs are comparatively robust to outliers and noise.

3. RFs provide give useful internal estimates of error, strength , correlation and variable importance.
4. RFs may produce a highly accurate classifier for various data sets.
5. RFs provide fast learning.

In this paper, the number of decision trees in RF are taken as 10. These were the default settings for WEKA tool.

Artificial Neural Networks

ANN uses Multilayer Perceptron (MLP) that is based on biological neurons, for model construction. ANN is used to model complex relationships between inputs and outputs and is used to search patterns in data set [34]. Multilayer feedforward network consists of one input layer, one or more hidden layers and one output layer [34].

Each layer consists of nodes that are connected to their immediate preceding layers for input and immediate succeeding layers for output. Back-propagation is the most commonly used learning algorithm in order to train multilayer feed forward networks and consists of two passes (forward and backward) through the different layers of network. In the forward pass a training input data set is applied and a set of outputs are produced as the actual response. In this pass weights of the network are fixed and the effect is propagated through the network layer by layer [34]. In the backward pass an error is calculated which is the difference between the networks actual and desired output. The error calculated is propagated backward through the network and the weights are readjusted in order to make the actual output closer to the desired response.
The benefits of ANN are its non-linearity, adaptivity, parallel architecture and fault tolerance.

Boosting Techniques

Boosting uses decision tree algorithm for creating new models. Boosting assigns weights to models based on their performance. There are many variants of boosting algorithms available in the literature. In this paper we evaluate the performance of two such variants AB [25] and LB [26] designed for classification purpose. The boosting algorithm works as follows [59]:

1. Assign equal weights to all the instances in the data set.
2. For each iteration m do the following:
   a. Apply learning algorithm to weighted data and store results of the modes.
   b. Compute error err for each model.
   c. If value of err is zero or greater or equal to 0.5 stop model prediction.
   d. For each instance in the data set do the following:
      i. If instance is correctly predicted by the model multiply instance by weight err/(1-err)
   e. Normalize respective weights for each instance

Classification

1. Assign zero weight to each class.
2. For each m models:
   a. Add –log(err/(1-err)) to previous weight of class.
3. Return class having highest weight.

LB uses additive LR for generating models and details can be obtained from [26].

Naïve Bayes and KStar

NB is a simple ad efficient algorithm for model prediction [38]. NB is based on probability based theorem and uses bayes theorem for classification purpose. KStar is an instance based learning algorithm that uses entropy distance measure [20]. It is based on distance between instances. The details on both the techniques can be obtained from [20, 38].

4.4 Performance Evaluation Methods

In order to validate the predicted models we use the following performance measures:

1. Sensitivity: It is defined as the ratio of classes predicted as faulty to the total number of classes actually faulty.
2. Specificity: It is defined as the ratio of classes predicted as non faulty to the total number of classes actually non faulty.
3. Precision: It is defined as the ratio of classes predicted correctly as faulty and non faulty to the total number of classes.
4. ROC analysis: The output of the predicted models can be analyzed using ROC analysis. ROC curve is a plot of sensitivity (on the y-axis) and 1-specificity (on the x-axis). Many cut off points are selected between 0 and 1 while the construction of ROC curves [32]. AUC is a measure obtained using ROC analysis. This gives optimal cut off point that maximizes both sensitivity and specificity. This measure is very effective in measuring the quality of the predicted models and is popularly being used in machine learning research [61]. The following rules can be used to categorize AUC:

- If ROC = 0.5, then No discrimination
- If 0.7 <= ROC < 0.8, then Acceptable discrimination
- If 0.8 <= ROC < 0.9, then Excellent discrimination
- If ROC <= 0.9, then Outstanding discrimination

In this paper, we use AUC as a measure to evaluate and assess the models predicted using machine learning techniques. We also use ROC analysis to obtain optimal cutoff point that provides balance between classes predicted as faulty and non faulty.

4.4 Cross Validation Method

We use K-cross validation method in order to validate the predicted models. In K-fold cross-validation, the original data set is partitioned into K subparts [56]. The K-1 subparts are used as training data and the single subpart is used as validation data. The cross-validation process is then repeated K times, with each of the K subparts used exactly once as the validation data. The result obtained from K folds results into a single estimation. This is popular method used in model validation. To obtain a realistic estimate we use K=10 to validate the model.

5. RESEARCH RESULTS

5.1 Descriptive Statistics and Correlation Analysis

The maximum (max.), minimum (min.), mean, median standard deviation (Std. Dev.), percentile values for each object oriented metric are shown in table 3.

TABLE 3
DESCRIPTIVE STATISTICS OF OBJECT ORIENTED METRICS

Metrics	Mean	Median	Std. Dev	Min	Max	Percentiles	
						25	75
WMC	9.226	6	9.156	0	55	3	13
DIT	1.568	1	1.102	1	5	1	1.25
NOC	0.162	0	1.596	0	23	0	0
CBO	7.559	5	8.860	0	74	3	8
RFC	21.303	14	26.42	0	166	5	25.25
LCOM	56.158	6	138.67	0	1417	1	54.25
CA	2.893	1	6.628	0	71	0	3
CE	4.705	2	6.490	0	36	1	5
NPM	8.572	5	8.550	0	55	3	12
LCOM3	1.328	0.955	0.614	0	2	0.833	2
LOC	133.94	56.5	248.02	0	2090	9	135
DAM	0.543	1	0.491	0	1	0	1
MOA	0.8931	0	1.806	0	10	0	1
MFA	0.216	0	0.387	0	1	0	0.058
CAM	0.483	0.437	0.277	0	1	0.285	0.666
IC	0.2135	0	0.431	0	2	0	0
CBM	0.230	0	0.488	0	3	0	0
AMC	9.746	6.571	10.01	0	49.46	3	15
MAX_CC	2.782	1	3.556	0	34	1	4
AVG_CC	1.031	1	0.724	0	6	0.702	1.268

We perform correlation analysis to gain insight about the dependencies amongst object oriented metrics. The value of coefficient between 0.9-1.0 is considered perfect, value between 0.7-0.9 is considered to be very large, and value between 0.5-0.7 is considered to be large. Hence, in table 4 the object oriented metrics with correlation coefficient greater than 0.5 and significant at 99% are shown in bold. There are many correlations amongst metrics hence, the metrics are not independent and represent redundant information.

TABLE 4

CORRELATION ANALYSIS AMONGST OBJECT ORIENTED METRICS

Metrics	WMC	DIT	NOC	CBO	RFC	LCOM	CA	CE	NPM	LCOM3	LOC	DAM	MOA	MFA	CAM	IC	CBM	AMC	MAX_CC	AVG_CC
WMC	1																			
DIT	-0.23	1																		
NOC	0.23	0.09	1																	
CBO	**0.69**	-0.01	0.40	1																
RFC	**0.80**	-0.08	0.25	**0.68**	1															
LCOM	**0.84**	-0.12	0.29	**0.72**	**0.59**	1														
CA	0.36	-0.06	0.27	**0.68**	0.06	0.58	1													
CE	**0.57**	0.04	0.27	**0.66**	**0.87**	0.39	-0.08	1												
NPM	**0.98**	-0.24	0.21	**0.67**	**0.73**	0.85	0.40	0.50	1											
LCOM3	-0.41	0.29	-0.09	-0.20	-0.41	-0.18	-0.06	-0.22	-0.39	1										
LOC	**0.72**	-0.07	0.18	**0.59**	**0.94**	0.52	-0.00	**0.81**	**0.66**	-0.35	1									
DAM	0.48	-0.32	0.07	0.22	0.46	0.25	0.05	0.26	0.45	-0.94	0.39	1								
MOA	**0.64**	-0.21	-0.02	0.50	**0.69**	0.50	0.05	0.63	**0.61**	-0.36	**0.67**	0.40	1							
MFA	-0.29	**0.92**	0.01	-0.10	-0.14	-0.17	-0.10	-0.03	-0.31	0.25	-0.12	-0.30	-0.21	1						
CAM	-0.56	0.38	-0.08	-0.37	-0.50	-0.36	-0.15	-0.36	-0.55	0.47	-0.41	-0.50	-0.44	0.40	1					
IC	0.14	0.03	-0.05	0.00	0.00	0.12	0.17	-0.16	0.16	-0.33	-0.01	0.30	0.10	0.17	-0.21	1				
CBM	0.11	0.09	-0.04	0.00	0.01	0.09	0.13	-0.12	0.12	-0.32	-0.01	0.30	0.08	0.21	-0.19	0.94	1			
AMC	0.34	0.12	0.06	0.35	**0.70**	0.14	-0.16	**0.65**	0.27	-0.45	**0.69**	0.43	0.44	0.13	-0.32	0.10	0.158	1		
MAX_CC	0.31	-0.06	0.02	0.29	0.41	0.19	0.06	0.33	0.30	-0.33	0.34	0.35	0.23	-0.09	-0.33	0.33	0.283	0.50	1	
AVG_CC	0.27	-0.23	0.02	0.25	0.37	0.12	0.02	0.32	0.26	-0.28	0.29	0.30	0.14	-0.24	-0.26	0.18	0.151	0.52	0.82	1

5.2 Model Prediction using LR analysis

In this section the results of univariate and multivariate LR analysis is presented. Table 5 summarizes the result of univariate LR analysis. The classes were treated as faulty if they consisted of one fault. Table 5 summarizes the values of coefficient (B), standard error (SE), significance (sig.) and Odds Ratio for each metric. The metrics with significance threshold above 0.01 significance level are shown in bold. As can be seen from Table 5, WMC, CBO, RFC, NPM, LOC, MOA, CAM, AMC metrics are found to be predictor of fault proneness. Rest of the metrics are not found to be a predictor of fault proneness. Hence, out of 19 metrics 8 metrics were found to be significant for predicting faulty classes. CAM metric has negative coefficient that indicates that this metric is inversely related with fault proneness.

TABLE 5

RESULTS OF UNIVARIATE LR ANALYSIS

Metric	B	SE	Sig.	Odds Ratio
WMC	0.056	0.018	**0.002**	1.058
DIT	-0.271	0.235	0.247	0.762
NOC	-15.048	3573.368	0.997	0.000
CBO	0.052	0.018	**0.005**	1.053
RFC	0.025	0.006	**0.000**	1.025
LCOM	0.002	0.001	0.158	1.002
CA	-0.098	0.082	0.232	0.907
NPM	0.050	0.020	**0.010**	1.052
LCOM3	-0.045	0.334	0.894	0.956
LOC	0.002	0.001	**0.001**	1.002
DAM	0.235	0.422	0.578	1.265
MOA	0.241	0.086	**0.005**	1.272
MFA	-1.183	0.719	0.100	0.306
CAM	-2.660	0.945	**0.005**	0.070
IC	-0.798	0.621	0.198	0.450
CBM	-0.792	0.598	0.185	0.453
AMC	0.050	0.018	**0.004**	1.051
MAX_CC	0.088	0.046	0.054	1.092
AVG_CC	0.439	0.233	0.060	1.551

Table 6 shows the model predicted using multivariate LR analysis. The model consists of CE metric. The R^2 of the model is 0.42. The cutoff point of the predicted model is obtained using ROC analysis as we did not want to select arbitrary cutoff point. The cutoff point for the model predicted is 0.08. The sensitivity of the model is 70.40% and specificity of the model is 69.6%.

TABLE 6

RESULTS OF MULTIVARIATE LR ANALYSIS

Metric	B	SE	Sig.	Odds Ratio
CE	0.119	0.025	0.000	1.126
Constant	-2.828	0.302	0.000	0.059

5.3 Subset Selection Using CFS

In machine learning, subset selection is a popular method for selection of important object oriented metrics. Table 7 shows the metrics identified to be important predictor of faulty classes using CFS method. These metrics can be used for model prediction using machine leaning techniques. Hence, CBO, RFC, CE, NPM and CAM are the best predictors of fault proneness and will be used for fault prediction model construction using machine learning techniques.

TABLE 7

METRICS SELECTED BY CFS METHOD

S. No	Metrics Selected
1	CBO
2	RFC
3	CE
4	NPM
5	CAM

5.4 Model Validation Results

Table 8 summarizes the results of 10-cross validation of LR, ANN, LB, AB, KStar, RF, NB, bagging techniques. The data set with 234 classes was divided into 10 approximately equal parts (9 parts of 23 classes and one part of 24 classes). The 9 parts is used for training purpose and one part is used for validation purpose. The results of models predicted using machine learning techniques were predicted using WEKA tool.

Table 8 shows the cutoff point, sensitivity, specificity, precision, AUC and SE measures for each of the predicted model. The optimal cutoff points have been selected using ROC analysis. Thus, this will give balance between faulty and non faulty classes. The following observations are made from Table 8:

- ***The models predicted using machine learning techniques are competitive with the model predicted using LR technique*:** The AUC of the entire models obtained using machine learning

methods is greater than the AUC of model obtained from LR technique.

- ***The models predicted using RF, AB, NB and KStar have acceptable discrimination in terms of AUC whereas model predicted using LB has excellent discrimination:*** The model predicted using LR, ANN and bagging techniques have AUC below 0.7 and thus their accuracy is not high. The model predicted using LB has AUC greater than 0.8 whereas all the other models have AUC above 0.7.

- ***Boosting algorithm LB gave the best results in terms of AUC as compared to all the other models predicted:*** Table 7 shows that the AUC of LB model is 0.806 which is greater than the AUC of models obtained from all the other techniques.

- ***In literature boosting techniques were not evaluated for predicting faulty classes using object oriented metrics:*** In previous studies predictive accuracy of boosting algorithm were not assessed and evaluated. The results show that boosting algorithm LB may be effective in predicting faulty classes.

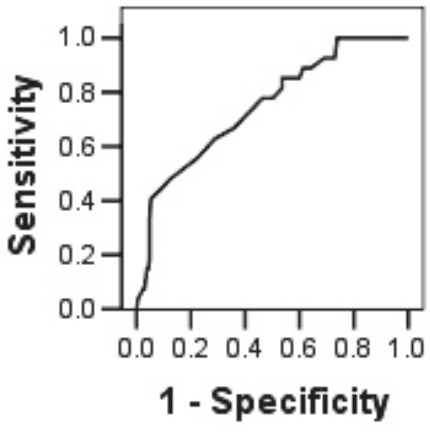
The results obtained suggest that the model predicted from the LB, RF, AB, NB and KStar have high discriminating power for predicting faulty and non faulty classes as compared to all the other predicted models. Zhou et al. [60] used three machine learning methods NB, RF and NNage and their predictive accuracy was found to be very low.

TABLE 8

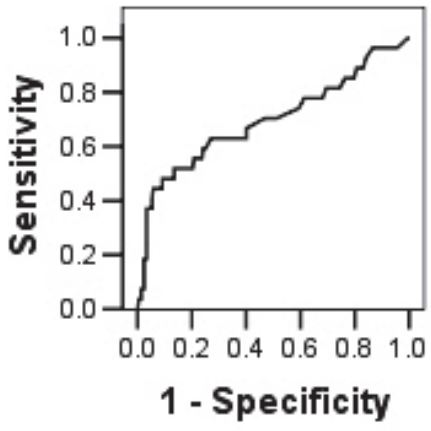
MODEL EVALUATION RESULTS

	Logistic Model	ANN Model	LB Model	RF Model	Bagging Model	AB Model	NB Model	KStar
Cut off point	0.08	0.07	0.12	0.09	0.1	0.10	0.09	0.03
Sensitivity	66.70	66.70	77.80	81.50	63.00	66.70	66.70	70.40
Specificity	64.70	61.80	78.70	64.30	72.90	64.30	64.30	67.10
Precision	64.90	62.29	77.45	66.25	71.83	64.54	64.54	67.48
AUC	0.689	0.696	0.806	0.785	0.696	0.747	0.747	0.76
SE	0.061	0.061	0.048	0.051	0.064	0.049	0.049	0.049

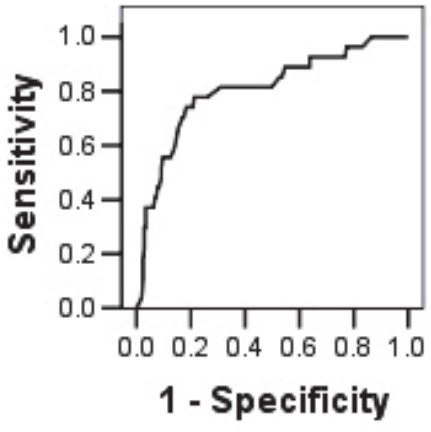
In Figure 1, the ROC curves obtained from LR, ANN, LB, AB, KStar, RF, NB, and Bagging techniques are shown.



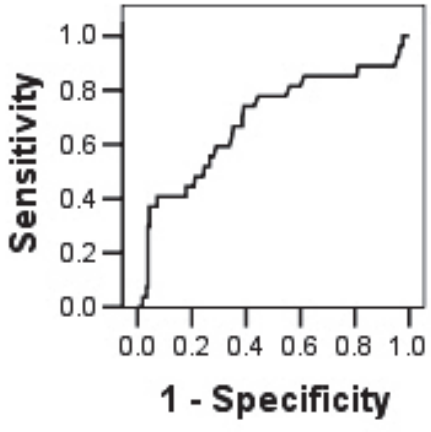
(a)



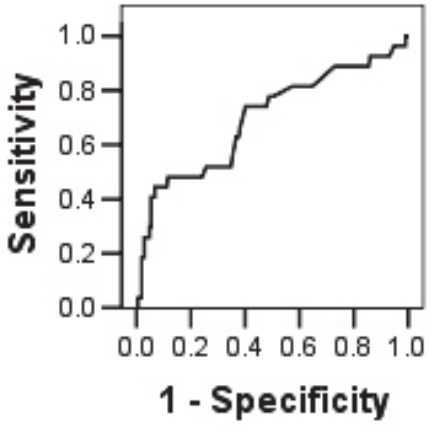
(b)



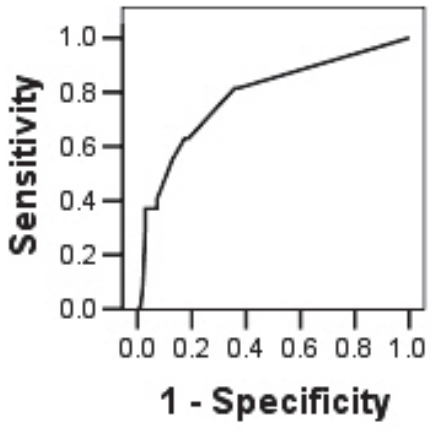
(c)



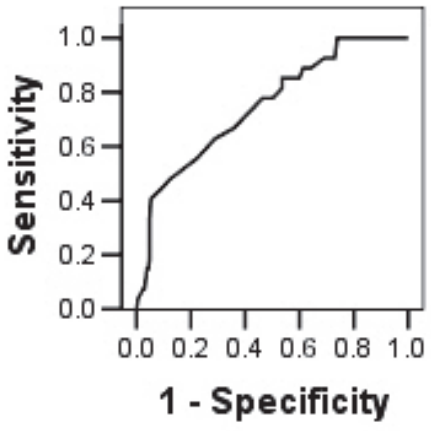
(d)



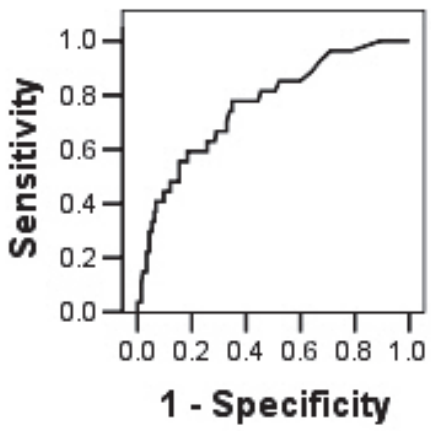
(e)



(f)



(g)



(h)

FIGURE 1: (A) AB (B) BAGGING (C) LB (D) LOGISTIC (E) ANN (F) RF (G) NB (H) KSTAR

5.5 Discussion of Results

The results predicted in this section can be used by the testing community to predict faulty classes in early phases of software development. The results are summarized as follows

- The results of univariate LR analysis show that WMC, CBO, RFC, NPM, LOC, MOA, AMC metrics are significant predictors of fault proneness. CAM metric has inverse relation with fault proneness of a class.

- The results of multivariate LR analysis show that CE can be used in predicting faulty classes in early phases of software development.

- CBO, RFC, CE, NPM and CAM are the best predictors of fault proneness using CFS method and can be used in predicting models using machine learning techniques.

- The fault prediction models developed using machine learning techniques have better performance as compared to the model predicted using LR technique.

- Amongst all the models predicted using machine learning techniques boosting algorithm LB has the best accuracy in terms of AUC.

6. THREATS TO VALIDITY

There are two types of threats to validity. One is threat to internal validity and the other is threat to external validity. The threats to external validity are more severe as compared to threats to internal validity. The threats to internal validity are present due to degree to which conclusions can be drawn between independent and dependent variables [61].

The threats to external validity are threats which are related to generalzability of the predicted models. The results in this paper are obtained from open source software and hence may not be applicable to other systems. The size of the data set is also not very large. These threats can be reduced by conducting more number of replicated studies across various systems.

7. CONCUSIONS AND FUTURE WORK

In this paper, we find the relation between nineteen object oriented metrics and fault proneness of a class. In order to validate this relation we have used LR techniques and seven machine learning techniques: ANN, RF, LB, AB, NB, KStar, Bagging. The results are based on data set obtained from open source software. The main contributions of this paper are 1) find the relation between object oriented metrics and faulty proneness 2) to obtain subset of object oriented metrics that are significant predictor of faulty classes and can be used in predicting models using machine learning techniques 3) to compare and assess machine learning techniques in order to gain insight about their predictive accuracy.

CBO, RFC, CE, NPM and CAM metrics are the best predictors of fault proneness using CFS method. The results show that predictive accuracy of LR model is low as the AUC is 0.689. The model predicted using LB technique yields the best AUC with value 0.806. Hence, software practitioners and researchers may use the machine learning techniques specially boosting algorithms for predicting faulty classes in early phases of software development.

The machine learning models predicted in this paper can help the testing community to focus the resources on the faulty parts of the software. The developers can also reconsider the software design and hence take necessary corrective actions. The fault prediction models can help the testers in planning and allocating resources in early phases of software development.

In future, we want to replicate our study with more object oriented metrics and different data sets so that generalized observations and conclusions can be made. We also intend to apply different machine learning algorithms such as slow learner genetic algorithm to further assess the accuracy of machine learning techniques.

REFRENCES

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

[2] K.K. Aggarwal, Y. Singh, A Kaur, R. Malhotra, "Empirical Analysis for Investigating the Effect of Object-Oriented Metrics on Fault Proneness: A Replicated Case Study," *Software Process Improvement and Practice*, Wiley, vol. 14, no. 1, 39-62, 2009.

[3] K.K. Aggarwal, Y. Singh, A Kaur, R. Malhotra, "Empirical study of object-oriented metrics," *Journal of Object Technology*, vol. 5, no. 8, 149-173, 2006.

[4] K.K. Aggarwal, Y. Singh, A Kaur, R. Malhotra, "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, 2005.

[5] K.K. Aggarwal, Y. Singh, A Kaur, R. Malhotra, "Investigating the Effect of Coupling Metrics on Fault Proneness in Object-Oriented Systems," *Software Quality Professional*, vol. 8, no. 4, 4-16, 2006.

[6] T. Ayodele, S. Zhou, R. Khusainov , "Machine Learning Email Prediction System (MLEPS)," in *International Journal for Infonomics (IJI)*, Vol. 3, no. 4, December 2010.

[7] V. Barnett, T. Price, "*Outliers in Statistical Data*," John Wiley & Sons, 1995.

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

[9] S. Bengio, "Statistical Machine Learning from Data Ensembles," IDIAP Research Institute, January 27, 2006.

[10] G. Biau, L. Devroye, G. Lugosi, "Consisting of Random Forests and other Averaging Classifiers," *Journal of Machine Learning Research*, 2008.

[11] L. Briand, W. Daly, and J. Wust, "Unified framework for cohesion measurement in object-oriented systems," *Empirical Software Engineering*, vol. 3, no.1, 65-117, 1998.

[12] L. Briand, W. Daly, and J. Wust, "A unified framework for coupling measurement in object-oriented systems," *IEEE Transactions on Software Engineering*, vol .25, no.1, 91-121, 1999.

[13] L. Briand, W. Daly, and J. Wust, "Exploring the relationships between design measures and software quality," *Journal of Systems and Software*, vol. 51, no. 3, 245-273, 2000.

[14] L. Briand, J. Wüst, H. Lounis, "Replicated Case Studies for Investigating Quality Factors in Object-Oriented Designs," *Empirical Software Engineering: An International Journal* , vol. 6, no. 1, 11-58, 2001.

[15] L. Breiman, "Random forests," *Machine Learning,* vol. 45, 5–32, 2001.

[16] M. Cartwright, and M. Shepperd, "An empirical investigation of an object-oriented software system," *IEEE Transactions of Software Engineering*, vol. 26, no. 8, 786-796, 1999.

[17] S. Chidamber, and C. Kamerer, "A metrics suite for object-oriented design," *IEEE Transactions on Software Engineering*, vol. 20, no. 6, 476-493, 1994.

[18] S. Chidamber, and C. Kamerer, "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, 1991.

[19] S. Chidamber, D. Darcy, and C. Kemerer, Managerial use of metrics for object-oriented software: An exploratory analysis. *IEEE Transactions on Software Engineering*, vol. 24, no. 8, 629-639, 1998.

[20] John G. Cleary, Leonard E. Trigg, "K*: An Instance-based Learner Using an Entropic Distance Measure," *In: 12th International Conference on Machine Learning*, 108-114, 1995.

[21] T. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Computation,* vol. 10, 1895–1924, 1998.

[22] R. Duda, P. Hart, D. Stork, *Pattern Classification*, second ed. John Wiley & Sons, New York, 2001.

[23] K. Elish, M. Elish, "Predicting defect-prone software modules using support vector machines," *Journal of System and Software*, vol. 81, 649-660.

[24] N. Fenton, N. Ohlsson, "Quantitative analysis of faults and failures in a complex software system," *IEEE Transactions on Software Engineering*, vol. 26, no. 8, 797–814, 2000.

[25] Y. Freund, R. Schapire, "Experiments with a new boosting algorithm," *In: Thirteenth International Conference on Machine Learning*, San Francisco, 148-156, 1996.

[26] J. Friedman, T. Hastie, R. Tibshirani, "Additive Logistic Regression: a Statistical View of Boosting", Stanford University.

[27] L. Guo, Y. Ma, B. Cukic, H. Singh, "Robust prediction of fault proneness by random forests," *In: Proceedings of the 15th International Symposium on Software Reliability Engineering (ISSRE'04)*, 417–428, 2004.

[28] T. Gyimothy, R. Ferenc, I, Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction," *IEEE Trans. Software Engineering*, vol. 31, no. 10, 897 – 910, 2005.

[29] J. Hair, R. Anderson, W. Tatham, "*Black Multivariate Data Analysis*," Pearson Education, 2006.

[30] M. Hall, "Correlation-based feature selection for discrete and numeric class machine learning," *In: Proceedings of the 17th International Conference on Machine Learning*, 359–366, 2000.

[31] J. Han, M. Kamber, "*Data Mining: Concepts and Techniques,*" Harchort India Private Limited, 2001.

[32] J. Hanley, BJ. McNeil, "The meaning and use of the area under a Receiver Operating Characteristic ROC curve,' *Radiology*, vol. 143, 29-36, 1982.

[33] R. Harrison, S.J Counsell, and R.V. Nithi, "An evaluation of MOOD set of object-oriented software metrics. IEEE Transactions on Software Engineering, vol. 4, no. 6, 491-496, 1998.

[34] S. Haykin, "*Neural Networks: A comprehensive Foundation*," Second Edition, Pearson Education 2004.

[35] B. Henderson-Sellers, "*Object-oriented metrics, measures of complexity*," Englewood Cliffs, N.J.: Prentice Hall, 1996.

[36] M. Hitz, and B. Montazeri, "Measuring coupling and cohesion in object-oriented systems," In Proceedings of the International Symposium on Applied Corporate Computing, Monterrey, Mexico, 1995.

[37] D. Hosmer, and S. Lemeshow, "*Applied logistic regression*," New York: John Wiley and Sons, 1989.

[38] G. John, P. Langley, "Estimating Continuous Distributions in Bayesian Classifiers," *In: Eleventh Conference on Uncertainty in Artificial Intelligence*, San Mateo, 338-345, 1995.

[39] T.M. Khoshgaftaar, E.D. Allen, J.P, Hudepohl, S.J. Aud, Application of neural networks to software quality modeling of a very large telecommunications system," *IEEE Transactions on Neural Networks*, vol. 8, no. 4, 902-909, 1997.

[40] T.M. Khoshgaftaar, E.D. Allen, J. Deng, Using regression trees to classify fault-prone software modules," *IEEE Transactions on Reliability*, vol. 51, no. 4, 455–462, 2002.

[41] A. Koru, H. Liu, "Building effective defect-prediction models in practice," *IEEE Software*, 23–29, 2005.

[42] S. Kanmani, V.R. Uthariaraj, V. Sankaranarayanan, and P. Thambidurai, "Object-oriented software fault prediction using neural networks," *Information and Software Technology*, vol. 49, 483–492, 2007.

[43] C. R. Kothari, "*Research Methodology. Methods and Techniques*," New Delhi**:** New Age International Limited, 2004.

[44] A. Lake, and C. Cook, "Use of factor analysis to develop OOP software complexity metrics," *In Proceedings of the 6th Annual Oregon Workshop on Software Metrics*, Silver Falls, Oregon, 1994.

[45] Y. Lee, B. Liang, S. Wu, and F. Wang, "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, 1995.

[46] W. Li, and S. Henry, "Object-oriented metrics that predict maintainability," *Journal of Systems and Software*, vol. 23, no. 2, 111-122, 1993.

[47] F. Livingston, "Implementation of Breiman's Random Forest Machine Learning algorithm," *Machine learning Journal*, 2008.

[48] M. Lorenz, and J. Kidd, *Object-oriented software metrics*," Englewood Cliffs, N.J.: Prentice-Hall, 1994.

[49] T. Menzies, J. Greenwald, and A. Frank, "Data Mining Static Code Attributes to Learn Defect Predictors," *IEEE Transactions On Software Engineering*, Vol. 32, No. 11, January 2007.

[50] Krzysztof Michalak, Halina Kwasnicka. Correlation-based Feature Selection Strategy in Neural Classification in Sixth International Conference on Intelligent Systems Design and Applications Volume 1, pp. 741-746.

[51] H. Olague, L. Etzkorn, S. Gholston, and S. Quattlebaum, "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*, vol. 3, no. 8, 402-419, 2007.

[52] G. Pai, , "Empirical analysis of Software Fault Content and Fault Proneness Using Bayesian Methods," *IEEE Transactions on software Engineering*, vol. 33, no. 10, 675-686, 2007.

[53] A. Porter, and R. Selly, "Empirically guided Software Development using Metric-Based Classification Trees," *IEEE Software*, vol. 7, no. 2, 46-54, 1990.

[54] Promise. http://promisedata.org/repository/.

[55] Y. Singh, A. Kaur, R. Malhotra., "Empirical Validation of Object-Oriented Metrics for Predicting Fault Proneness," *Software Quality Journal*, vol. 18, no. 1, 3-35, 2010.

[56] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *J. Royal Stat. Soc*., vol. 36, 111-147, 1974.

[57] M.H, Tang, M.H. Kaoand, and M.H. Chen, "An Empirical Study on Object-Oriented Metrics," *In Proceedings of Metrics*, 242-249, 1999.

[58] D. Tegarden, S, Sheetz, and D. Monarchi, 'A software complexity model of object-oriented systems." *Decision Support Systems*, vol. 13 , no. 3-4, 241-262, 1995.

[59] I. Witten, E. Frank, "*Data Mining: Practical Machine Learning Tools and Techniques*," second ed. Morgan Kaufmann, San Francisco, 2005.

[60] Y. Zhou , B. Xu, and H. Leung, " Empirical analysis of Object-Oriented Design Metrics for predicting high severity faults," *IEEE Transactions on Software Engineering*, vol. 32, no. 10, 771-784, 2006.

[61] Y. Zhou, B. Xu, and H. Leung, "On the ability of complexity metrics to predict fault-prone classes in object-oriented systems," *The Journal of Systems and Software*, vol. 83,:660–674, 2010.

ABOUT THE AUTHORS

Ruchika Malhotra. She is an assistant professor at the Department of Software Engineering, Delhi Technological University (formerly Delhi College of Engineering), Delhi, India, She was an assistant professor at the University School of Information Technology, Guru Gobind Singh Indraprastha University, Delhi, India. Prior to joining the school, she worked as full-time research scholar and received a doctoral research fellowship from the University School of Information Technology, Guru Gobind Singh Indraprastha Delhi, India. She received her master's and doctorate degree in software engineering from the University School of Information Technology, Guru Gobind Singh Indraprastha University, Delhi, India. Her research interests are in software testing, improving software quality, statistical and adaptive prediction models, software metrics, neural nets modeling, and the definition and validation of software metrics. She has published more for than 50 research papers in international journals and conferences. Malhotra can be contacted by e-mail at ruchikamalhotra2004@yahoo.com.

Yogesh Singh. He is Vice Chancellor at M. S. University of Baroda, Baroda, India. He was a professor at the University School of Information Technology, Guru Gobind Singh Indraprastha University, Delhi, India. He also was Controller of Examinations at the Guru Gobind Singh Indraprastha University, Delhi, India. He was founder Head (1999-2001) and Dean (2001-2006) of the University School of Information Technology, Guru Gobind Singh Indraprastha University, Delhi, India. He received his master's degree and doctorate from the National Institute of Technology, Kurukshetra, India. His research interests include software engineering focusing on planning, testing, metrics, and neural networks. He is coauthor of a book on software engineering, and is a Fellow of IETE and member of IEEE. He has more than 200 publications in international and national journals and conferences. Singh can be contacted by e-mail at ys66@rediffmail.com.