Empirical Evaluation of Machine Learning Algorithms for Fault Prediction

Arvinder Kaur and Inderpreet Kaur

***Abstract*—Producing quality software is a very challenging task looking at the size and complexity of software developed these days. Predicting software quality early helps in using testing resources optimally. So, many statistical and machine learning techniques are used to predict quality classes in software. In this work, six machine learning classifiers have been used to estimate the fault proneness of 5885 classes used in five open source software on the basis of object-oriented metrics calculated on these classes. Bagging and J48 classifiers turn out to be the best one amongst the classifiers used.**

***Index Terms*—Classifiers, fault proneness, object-oreiented software metrics, quality metrics.**

I. Introduction

Predicting software defects with reliability is difficult part in software engineering. Researchers have been trying hard by implementing various defect/bug prediction techniques. Tracking the defect as early as possible in a software life cycle will not only improve the effective cost but will also help to achieve the customers' satisfaction and reliability of the software developed.

The intensification of software complexity and the constraints, under which it is developed, increases the chances of faults in software. The faulty classes may increase the development and maintenance costs due to software failures and hence reduce reliability of the software [1].

Studies of fault proneness have been conducted earlier [2]-[18].These studies used software metrics [19]-[29] to develop mathematical models for prediction of fault proneness. Empirical validation of machine learning methods is needed to verify the potentiality of machine learning algorithms. Clues collected from these empirical studies are considered to be powerful support for testing any given hypothesis.

In this paper software quality estimation using 6 machine learning classifiers is performed. Datasets provided by Marco's website have been used [30]. These datasets allow the researchers to compare the various defect prediction techniques and evaluate if there is improvement made or not. Open source softwares are preferred for these studies as results of these can be compared and repetition of validation can be performed.

The paper is organized as follows: Section II gives the related work done. Section III provides the metrics studied with the research methodology followed in this study. Results of research are summarized in Section IV. Section V describes the conclusions made from the study.

Manuscript received August 15, 2013; revised October 23, 2013.

Arvinder Kaur is with University School of Information and Communication Technology , Department of Information and Technology, Guru Gobind Singh Indraprastha University, Delhi, India (e-mail: arvinder70@gmail.com).

Inderpreet Kaur is with Department of Information Technology, Guru Gobind Singh Indraprastha University, Delhi, India (e-mail: inderpreet_2772@yahoo.co.in).

II. Related Work

Briand *et al.* extracted 49 metrics to identify model for predicting fault proneness of classes [7]. System that was investigated was medium sized c++ software system developed by undergraduate / graduate students. The eight systems under study had total 180 classes. Univariate and multivariate analysis were done to find individual and combined impact of OO metrics and fault proneness. Result of the study showed that all metrics except NOC to be significant predictor of fault proneness.

Nagappan *et al.* [31] used catalog of source code metrics to predict post release defects at module level on five Microsoft systems and found it was possible to build predictors for one individual project, but that no predictor would perform well on all projects.

The study described by Arvinder *et al.* [32] is a replication of an analogous study conducted by Briand *et al.* [7]. The study provided empirical evidence to draw the strong conclusions across studies. Results of their study show that many metrics capture the same dimensions in the metric set hence are based on comparable ideas and provide redundant information. It is shown that by using a subset of metrics, prediction models can be built to identify faulty classes. The model predicts faulty classes with more than 90% accuracy. The predicted model shows that import coupling and size metrics are strongly related to fault proneness, confirming the results from previous studies. However, there are also differences reported in this study with respect to previous studies such as inheritance metric which counts methods inherited in a class is also included in the predicted model.

Menzies *et al.* [33] disagreed to the belief that the exact static source code metric used is not as important as which learning algorithm is used. Based on data from NASA Metrics Data Program (MDP), authors compared the impact of using LOC, Alstead and McCabe metrics, versus the impact of using the Na ïve Bayes, OneR and J48 algorithms.

Arvinder *et al.* [34] compared the LR and ANN approaches in their study of predicting fault proneness. NASA datasets were used for fault proneness prediction using regression and decision tree methods. The study proves that LR and ANN provide good models for prediction

Arvinder *et al.* [35] examined the effect of logistic regression and six machine learning methods (Artificial neural network, decision tree, support vector machine, cascade correlation network, group method of data handling polynomial method, gene expression programming). The

DOI: 10.7763/LNSE.2014.V2.118

performance of the methods is compared by computing the area under the curve using Receiver Operating Characteristic (ROC) analysis. The results show that the area under the curve (measured from the ROC analysis) of model predicted using decision tree modeling is 0.865 and is a better model than the model predicted using regression and other machine learning methods.

III. Research Background

In this section, summary of metrics selected for this paper, machine learning methods used, research hypothesis and measures used for the model are presented.

A. Metrics and Fault Proneness

Fault proneness is defined as probability of presence of fault in the module. In our research fault proneness is binary dependent variable while other metrics form the independent variables. We aim at exploring the effect of metrics on the fault proneness of a class using machine learning methods

B. Research Hypothesis

We tested the hypothesis given below to test the machine learning methods used.

H_o: There is no difference in accuracy of six machine learning classifiers.

H_A: At least one of the classifiers is more accurate than other.

C. Data Collection

This paper uses the datasets provided by Marco *et al.* [30]. The dataset is a collection of models and metrics of five software systems and their histories. These datasets are collection of models and metrics of five software systems and their histories.

The goal of such a dataset is to allow researchers to compare different defect prediction approaches and to evaluate whether a new technique is an improvement over existing ones. The datasets were designed to perform defect prediction at the class level. Datasets [30] used are given in Table I.

TABLE I: Datasets Description

System url	Prediction release	Time period	#Classes	#Versions
Eclipse JDT Core www.eclipse.org/jdt/core/	3.4	1.01.2005 6.17.2008	997	91
Eclipse PDE UI www.eclipse.org/pde/pde-ui/	3.4.1	1.01.2005 9.11.2008	1562	97
Equinox framework www.eclipse.org/equinox/	3.4	1.01.2005 6.25.2008	439	91
Mylyn www.eclipse.org/mylyn/	3.1	1.17.2005 3.17.2009	2196	98
Apache Lucene lucene.apache.org	2.4.0	1.01.2005 10.08.2008	691	99

Metrics used for prediction are provided in Table II.

D. Machine Learning and Model Prediction

Machine Learning is a branch of Artificial Intelligence which is concerned with design and development of algorithms that describes behaviors based on empirical data [36]. Researchers have started using machine learning as it provides better results than regression and can incorporate complex nature of data.

TABLE II: Metrics Description

Metrics	Description	Source
WMC	Weighted methods per class (NOM: Number of Methods in the QMOOD metric suite)	C&K
DIT	Depth of Inheritance Tree	C&K
NOC	Number of Children	C&K
RFC	Response for a Class	C&K
CBO	Coupling between object classes	C&K
LCOM	Lack of cohesion in methods	C&K
Ca	Afferent coupling	Martin's metrics
Ce	Efferent coupling	Martin's metrics
NOA	numberOfAttributes	Size metrics
NOAI	numberOfAttributesInherited	Size metrics
NOM	numberOfMethods	Size metrics
NPM	Number of Public Methods for a class(Also called as CIS: Class Interface Size)	QMOOD
NOMI	numberOfMethodsInherited	Size metrics
NOPRA	numberOfPrivateAttributes	Size metrics
NOPRM	numberOfPrivateMethods	Size metrics
NOPA	numberOfPublicAttributes	Size metrics
NTB	nonTrivialBugs	Bug Metrics
MB	majorBugs	Bug Metrics
CB	criticalBugs	Bug Metrics
HPB	highPriorityBugs	Bug Metrics
CC	Cyclomatic complexity	McCabe's
LOC	Lines of Code	McCabe's

Classification is data mining technique used to predict group membership for data instances. Various classifiers used in this paper are summarized in Table III.

E. Measures for Model Validation

Specificity and Sensitivity validate the model's correctness. While specificity means proportion of classes predicted to be fault prone, sensitivity states the classes correctly predicted to be fault prone.

Precision is the proportion of classes predicted correctly and F- measure is the harmonic mean on recall (sensitivity) and precision.

Receiver operating characteristic (ROC) is plot of sensitivity on Y-coordinate and its specificity on x-coordinate. ROC is used effectively to evaluate the performance of prediction models [41]. Area under ROC Curve (AUC) is combined measure of sensitivity and specificity. To compute the accuracy of model being predicted, AUC is used [41].

TABLE III: CLASSIFIERS DESCRIPTION

Classifier	Description
Naïve Bayes	A naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions. A more descriptive term for the underlying probability model would be "independent feature model"[37].
Logistic Regression	Logistic regression is a type of regression analysis used for predicting the outcome of a categorical (a variable that can take on a limited number of categories) criterion variable based on one or more predictor variables. Logistic regression can be bi- or multinomial [38].
Instance Based(IB1)	It is Nearest-neighbor classifier and uses normalized Euclidean distance to find the training instance closest to the given test instance, and predicts the same class as this training instance. If multiple instances have the same (smallest) distance to the test instance, the first one found is used [36].
Bagging	Bagging is a ``bootstrap'' ensemble method that creates individuals for its ensemble by training each classifier on a random redistribution of the training set. Each classifier's training set is generated by randomly drawing, with replacement, N examples - where N is the size of the original training set; many of the original examples may be repeated in the resulting training set while others may be left out. Each individual classifier in the ensemble is generated with a different random sampling of the training set [39].
J48 Decision Tree	A decision tree is a predictive machine-learning model that decides the target value (dependent variable) of a new sample based on various attribute values of the available data. The internal nodes of a decision tree denote the different attributes; the branches between the nodes tell us the possible values that these attributes can have in the observed samples, while the terminal nodes tell us the final value (classification) of the dependent variable [36].
Random Forest	It is a learning ensemble consisting of a bagging of un-pruned decision tree learners with a randomized selection of features at each split [40].

IV. EXPERIMENTS AND RESULTS

In this section we present the results of multivariate analysis of five datasets specified. In this paper, six well known classification algorithms have been used. Classifiers selected are Random Forest, Naive Bayes, Bagging, J48, Logistic regression and IB1 These six classifiers have been chosen for the current study as previous studies indicate that these classifiers provide better than average performance in software fault prediction [42].

TABLE IV: CLASSIFIERS DESCRIPTION

Training Project	Modeling Technique	Accuracy	Precision	Recall	AUC	Specificity	PF
Eclipse							
	Naive Bayes	0.87	0.77	0.54	0.59	0.96	0.04
	Logistic	0.99	0.99	0.98	0.59	1.00	0.00
	IB1	0.94	0.89	0.82	0.59	0.97	0.03
	Bagging	1.00	1.00	1.00	0.59	1.00	0.00
	J48	1.00	1.00	1.00	0.59	1.00	0.00
	Random Forest	1.00	1.00	1.00	0.59	1.00	0.00
Equinox	**Naive Bayes**	0.76	0.80	0.53	0.59	0.91	0.67
	Logistic	0.99	1.00	0.98	0.59	1.00	0.98
	IB1	0.87	0.86	0.80	0.59	0.91	0.84
	Bagging	1.00	1.00	1.00	0.59	1.00	1.00
	J48	1.00	1.00	1.00	0.59	1.00	1.00
	Random Forest	1.00	0.99	1.00	0.59	0.99	0.99
Lucene	**Naive Bayes**	0.88	0.34	0.36	0.35	0.59	0.07
	Logistic	1.00	1.00	0.97	0.98	0.59	0.00
	IB1	0.94	0.77	0.53	0.63	0.59	0.02
	Bagging	1.00	1.00	1.00	1.00	0.59	0.00
	J48	1.00	1.00	1.00	1.00	0.59	0.00
	Random Forest	1.00	1.00	0.98	0.99	0.59	0.00
Mylyn	**Naive Bayes**	0.88	0.55	0.51	0.59	0.94	0.06
	Logistic	1.00	1.00	1.00	0.59	1.00	0.00
	IB1	0.97	0.94	0.85	0.59	0.99	0.01
	Bagging	1.00	1.00	1.00	0.59	1.00	0.00
	J48	1.00	1.00	1.00	0.59	1.00	0.00
	Random Forest	1.00	1.00	1.00	0.59	1.00	0.00
PDE	**Naive Bayes**	0.85	0.43	0.32	0.59	0.93	0.07
	Logistic	1.00	1.00	0.99	0.59	1.00	0.00
	IB1	0.91	0.74	0.58	0.59	0.97	0.03
	Bagging	1.00	1.00	1.00	0.59	1.00	0.00
	J48	1.00	1.00	1.00	0.59	1.00	0.00
	Random Forest	1.00	1.00	1.00	0.59	1.00	0.00

A. Multivariate Analysis

Cross validation with 10 fold was performed. Table IV shows the multivariate analysis of the system under study where Random Forest with optimum AUC value provides a good model for study of prediction of fault proneness.

Rejection of null hypothesis can be preceded with post hoc test. Demsar [43] recommends Friedman test followed by corresponding post – hoc Nemenyi test if more than two classifiers are taken over multiple datasets. Demsar recommends these tests as they are less strict on the data. But the non parametric tests do not utilize all the facts available, since the actual data values are not used in it. So, parametric tests would be recommended with respect to non parametric tests, when there is a good reason for them to use.

Procedure provided by Demsar is followed in this work. Mean AUC as measure of interest is taken, 10 by 10 cross validation is performed with 95% confidence level (p=0.95) as threshold to check significance (p<0.95). We have taken 6 classifiers over 5 data sets for the Friedman test. Friedman test will show whether there is statistical difference between the classifiers used or not and then Nemenyi test would be performed to decide which classifier gives best result for the 5 datasets used.

Ken Black's Book on Business Statistics [44] gave the following implementation for Friedman test.:

$$x^2 = \frac{12}{bc(c+1)} \sum_j R_i^2 - 3b(c+1)$$

where b is number of blocks (rows) and c is number of treatment levels (columns). Here k is the number of classifiers used, N is number of datasets, R_j is average rank of classifier j taken over the given datasets. $R_j = \frac{1}{N} \sum_j r_i^j$, where r_i^j is the rank of j^{th} classifier over i^{th} data. F_f is F-distribution with K-1 and (k-1) (N-1) degrees of freedom and the critical values available.

Nemenyi test is post hoc test applied when the null hypothesis is rejected. It compares the classifiers with each other.

Critical difference in this test is evaluated as:

$$CD = q_a \sqrt{\frac{k(k+1)}{6N}},$$

where q_a is critical value in Nemenyi test [38]. If the difference of mean ranks between two classifiers is larger than value of CD, their performance difference is significant. Here we have N= 5 and k=6.

For c=6 and df = 6-1=5.The critical value of $\chi^2_{.05,5} = 11.0705$. The observed value of chi-square is higher than critical value, the decision is to reject the null hypothesis.

Post hoc Nemenyi Test is performed because null hypothesis has been rejected in Friedman test that means there is difference in performance of the six classifiers used. With six classifiers, the critical value q_a is 2.85. Hence the critical difference is $CD = 2.85\sqrt{\frac{6(6+1)}{6*5}} = 3.37$

Results show that difference in mean ranks of Random Forest, Bagging, J48, IB1, Naïve Bayes and Logistic Regression is less than CD which means it is statistically insignificant, so they perform equally well.

Bagging and J48 provide similar results on all the datasets taken in this work. The ranks of these classifiers were best compared to other classifiers.

TABLE V: RANK WISE ORDERING OF CLASSIFIERS

Classifier	Rank	Mean Rank
Bagging	5	1
J48	5	1
Random Forest	8	1.6
Logistic	10	2
IB1	16	3.2
Naïve Bayes	19	3.8

Roc plot for multivariate analysis using those six classifiers is given in Fig. 1 for the dataset Lucene.

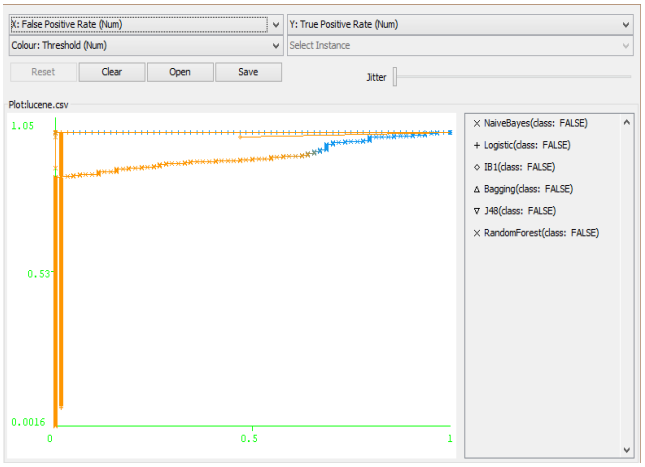

Fig. 1. ROC plot for Lucene.

V. CONCLUDING REMARKS AND FUTURE WORK

The goal of our research is to empirically study performance of various classifiers for fault prediction on the data sets provided by Marco D'Ambros [45].

Since the difference in mean ranks of Random Forest, Bagging, J48, IB1, Naïve Bayes and Logistic Regression is less than CD making it statistically insignificant, so they perform equally well.

Bagging and J48 provide the similar and best results for all the datasets.

While highest ROC(AUC) was 0.59 the Specificity, sensitivity, accuracy and precision were 1.0 in case of Bagging and J48 for all the datasets taken.

This study confirms that constructing machine learning algorithms (Bagging and J48) for fault prediction is feasible and can be adapted for OO systems providing usefulness in predicting fault proneness.

More studies similar to this research may be conducted on different datasets to provide the generalized results for different organizations. We plan to replicate this study on other machine learning algorithms and focus on cost/benefit analysis to determine whether model would be economically possible.

REFERENCES

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

[2] N. Ohlsson and H. Alberg, "Predicting fault-prone software modules in telephone switches," *IEEE Trans. Software Eng*., vol. 22, no. 12, 1996, pp. 886- 894.

[3] M. Alshayeb and W. Li, "An Empirical Validation of Object-Oriented Metrics in Two Different Iterative Software Processes," *IEEE transaction on software engineering*, 2003, vol. 12, no. 11, pp. 1043-1049.

[4] P. Bellini, I. Bruno, P. Nesiand, and D. Rogai, "Comparing fault-proneness estimation models," in *Proc. 10th IEEE International Conference on Engineering of Complex Computer Systems*, 2005, pp. 205–214.

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

[6] L. Briand, J. Daly, and J. Wust, "A Unified Framework for Coupling Measurement in Object-Oriented Systems," *IEEE Transactions on software Engineering*, 1999, vol. 25, pp. 91-121.

[7] L. Briand, J. Daly, V. Porter, and J. Wust, "Exploring the relationships between design measures and software quality," *Journal of Systems and Software*, 2000, vol. 5, pp. 245-273.

[8] L. Briand, W. L. Melo, and J. Wust, "Assessing the applicability of fault-proneness models across object oriented software projects," *IEEE Trans. on Software Engineering*, 2002, vol. 28, no. 7, pp. 706–720.

[9] S. Chidamber, D. Darcy, and C. Kemerer, "Managerial use of Metrics for Object-Oriented Software: An Exploratory Analysis," *IEEE Transactions on Software Engineering*, 1998, vol. 24, no. 8, pp. 629-639.

[10] K. Emam *et al*., "The Confounding Effect of Class Size on The Validity of Object-Oriented Metrics," *IEEE Transactions on Software Engineering*, 2001, vol. 27, no. 7, pp. 630-650.

[11] K. Emam, El, W. Melo, and J. Machado, "The Prediction of Faulty Classes Using Object-Oriented Design Metrics," *Journal of Systems and Software*, 2001, vol. 56, pp. 63-75.

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

[13] R. Kollmann, P. Selonen, and E. Stroulia, "A study on the current state of the art in tool-supported UML-based static reverse engineering," in *Pro. WCRE*, 2002, pp. 22-32.

[14] T. M. Khoshgoftaar and N. Seliya, "Comparative assessment of software quality classification techniques: An empirical study," *Empirical Software Engineering*, 2004, vol. 9, pp. 229–257.

[15] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," *IEEE Trans.on Software Engineering*, 2007, vol. 33, no. 1, pp. 2–13.

[16] H. Olague, L. Etzkorn, S. Gholston, and S. Quat-tlebaum, "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*, 2007, vol. 33, no. 8, pp. 402-419 .

[17] R. Subramanyam and M. S. Krishnan, "Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects," *IEEE transaction on software engineering*, 2003, vol. 29, no. 4, pp. 297-310.

[18] I. H. Witten and E. Frank, "Data mining: practical machine learning tools and techniques," *Morgan Kaufmann*, 2005.

[19] K. K. Aggarwal, Y. Singh, A. Kaur, and R. Malhotra, "Empirical Study of Object-Oriented Metrics," *Journal of Object Technology*, 2006, vol. 5, pp. 8.

[20] V. Basili, L. Briand, and W. L. Melo, "A Validation of Object-Oriented Design Metrics as Quality Indicators." *IEEE Transactions on Software Engineering*, 1996, vol. 22, no. 10, pp. 267-271.

[21] S. Chidamber and C. F. Kemerer, "A metrics Suite for Object-Oriented Design," *IEEE Transactions on Software Engineering*, 1994,vol SE-20, no. 6, pp. 476-493.

[22] S. R. Chidamber and C. F. Kemerer, "Towards a metrics suite for object oriented design," in *Proc. 6th ACM Conference on Object-Oriented Programming Systems Languages and Applications (OOPSLA)*, Phoenix, Arizona, 1991, pp. 197–211.

[23] B. Henderson-Sellers, *Object-Oriented Metrics, Measures of Complexity*, Prentice Hall, 1996, ISBN: 0-13-239872-9.

[24] M. Hitz and B. Montazeri, "Measuring Coupling and Cohesion in Object-Oriented Systems," in *Proc. Int. Symposium on Applied Corporate Computing, Monterrey*, Mexico, 1995.

[25] W. Li and S. Henry, "Object Oriented Metrics that Predict Maintainability," *Journal of Systems and Software*, 1993, vol. 23, no. 2, pp. 111-122.

[26] M. Lorenz and J. Kidd, *Object-Oriented Software Metrics*, Prentice-Hall, 1994.

[27] McCabe & Associates, *McCabe Object Oriented Tool User's Instructions*, 1994.

[28] L. Rosenberg and L. Hyatt, "Software Quality Metrics for Object Oriented System Environments," NASA Technical Report, 1995.

[29] Schroeder, "A practical Guide to Object-Oriented Metrics," *IT Professional*, 1999, pp. 1-6, 30-36.

[30] Bug. [Online]. Available: http://bug.inf.usi.ch

[31] N. Nagappan, T. Ball, and A. Zeller, "Mining metrics to predict component failures," in *Proc. ICSE*, ACM, 2006, pp. 452-461.

[32] K. K. Aggarwal, Y. Singh, A. Kaur, and 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, 2009, vol. 14, issue 1, pp. 39–62.

[33] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," *IEEE Trans. Software Eng*., 2007, vol. 33, no. 1, pp. 2-13.

[34] Y. Singh, A. Kaur, and R. Malhotra, "Predicting Software Fault Proneness Model Using Neural Network," *Agile Processes in Software Engineering and Extreme Programming*, Springer, 2008, vol. 9, 2008, pp. 215-217.

[35] Y. Singh, A. Kaur, and R. Malhotra, "Prediction of Fault-Prone Software Modules Using Statistical and Machine Learning Methods," *International Journal of Computer Apllications*, 2010, vol. 1, no. 22, pp. 0975-8887.

[36] T. Mitchell. (1997). Machine Learning. McGraw Hill. [Online]. Available: http://www.cs.cmu.edu/~tom/mlbook-chapter-slides.html.

[37] K. Huang, "Discriminative Naive Bayesian Classifiers," Department of Computer Science and Engineering, the Chinese University of Hong Kong, 2003.

[38] D. W. Hosmer and S. Lemeshow, *Applied Logistic Regression*, ISBN: 9780471356325.

[39] L. Breiman, "Bagging Predictors," *Machine Learning*, 1996, vol. 26, pp. 123-140.

[40] L. Breiman, "Random Forests," *Machine Learning*, 2001, vol. 45, no. 1, pp. 5-32.

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

[42] B. Boehm and P. Papaccio, "Understanding and controlling software costs," *IEEE Transactions on Software Engineering*, 1988, vol. 14, pp. 1462–1477

[43] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach Learn Res.*, vol. 7, pp. 1–30, 2006.

[44] Y. Jiang, B. Cukic, and Y. Ma, "Techniques for evaluating fault prediction models," *Empir Software Eng*., vol. 13, pp. 561-595, 2008.

[45] M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating Defect Prediction Approaches: A Benchmark nd an Extensive Comparison," *Empir Software Eng.*, vol. 17, 2012, pp. 531-577, DOI: 10.1007/s10664-011-9173-9.

Arvinder Kaur is an associate professor in the University School of Information & Communication Technology. She obtained her Ph.D. from GGS Indraprastha University & M.E in Computer Science from Thapar Institute of Engg. & Tech. Prior to joining the school, she worked with Dr. B.R. Ambedkar Regional Engineering College, Jalandhar(1993-2000) and Thapar Institute of Engg. & Tech. (1990-1993) . She has also worked for Gabriel India Ltd., Parwanoo (H.P) as Engineer (R&D). Her research interests include Software Engineering, Object-Oriented Software Engineering, Software Metrics, Microprocessors, operating systems, artificial intelligence and computer networks. She is also a lifetime member of ISTE & CSI. She has published 80 research papers in International / National journals and conferences. Her paper titled " Analysis of object oriented Metrics", has been published as book chapter in a book titled Innovations in Software Measurement, Shaker - Verlag, Aachen 2005.Her seven research papers have been published as book chapters in various Springer publications. Her Arnetminer h index is 7.

Inderpreet Kaur is an assistant professor in Institute of Information Technology and Management, GGSIPU. She completed her M.Tech. from University school of Information and communication Technology, GGSIPU under the guidance of Dr. Arvinder Kaur and B. Tech. from Guru Tegh Bahadur Institute of Technology , GGSIPU. Her areas of interest include software engineering, artificial intelligence and data mining.