Performance Evaluation of Ensemble Methods For Software Fault Prediction: An Experiment

Shahid Hussain, Jacky Keung, Arif Ali Khan, Kwabena Ebo BENNIN
Department of Computer Science
City University of Hong Kong
Shussain7-c @my.cityu.edu.hk, jwkeung@cityu.edu.hk, [aliakhan2-c,kebennin2-c]@ my.cityu.edu.hk

ABSTRACT
In object-oriented software development, a plethora of studies have been carried out to present the application of machine learning algorithms for fault prediction. Furthermore, it has been empirically validated that an ensemble method can improve classification performance as compared to a single classifier. But, due to the inherent differences among machine learning and data mining approaches, the classification performance of ensemble methods will be varied. In this study, we investigated and evaluated the performance of different ensemble methods with itself and base-level classifiers, in predicting the faults proneness classes. Subsequently, we used three ensemble methods AdaboostM1, Vote and StackingC with five base-level classifiers namely Naivebayes, Logistic, J48, VotedPerceptron and SMO in Weka tool. In order to evaluate the performance of ensemble methods, we retrieved twelve datasets of open source projects from PROMISE repository. In this experiment, we used k-fold (k=10) cross-validation and ROC analysis for validation. Besides, we used recall, precision, accuracy, F-value measures to evaluate the performance of ensemble methods and base-level Classifiers. Finally, we observed significant performance improvement of applying ensemble methods as compared to its base-level classifier, and among ensemble methods we observed StackingC outperformed other selected ensemble methods for software fault prediction.

Categories and Subject Descriptors
D.2.10 [**Software Engineering**]: Metrics – *Performance Measures and Product Metrics*

General Terms
Algorithms, Experimentation, Performance, Design

Keywords
Ensemble Methods, Chidamber and Kemerer (CK) Metrics, Classifiers, Performance, Measures, Weka

1. INTRODUCTION
The complexity of software is related to its development constraints, so it is difficult to develop software without faults. This problem can be overcome by estimating and predicting the quality features of a software such as effort, fault-proneness, reliability, productivity and maintenance efforts during the early phase of its development. Usually, software metrics have been used [4,5] in estimating and predicting the quality features of a software. A lot of metrics have been proposed and empirically validated to estimate or predict the quality attributes, such as Chidamber and Kemerer (CK) design metric suite [1], MOOD metric suite[2], and QMOOD metric suite [3]. In our study, we used Chidamber and Kemerer (CK) metrics, which have been introduced as a measurement suite, to measure the internal properties of object-oriented system.

There are many studies which have found association between CK metrics and fault proneness of classes through prediction models. Besides, the behavior of several predicting quantitative models have been observed from a very simple linear analysis to the use of more complex models such as SVM, Logistic regression, neural network and decision tree [6,7,8,11,12].

The role of data mining and machine learning approaches in decision making, by extracting and refining the useful knowledge, and reducing the information overload, have been empirically validated, and organization can extensively collect relevant and meaningful data [7,12].

But, due to inherent differences among these data mining and machine learning approaches, questions arise for the evaluation of predicted results of these approaches. In machine learning and data mining, an ensemble method is a meta level classifier, which combines one or more than one base-level classifiers and produces more accurate result on the bases of their individual prediction. [9,13].

In this study, we investigated and evaluated the performance of ensemble methods in presenting the association between CK metrics and fault proneness of classes and making accurate predictions. We compared the performance of ensembles methods to each other with group of three and five base-level classifiers. Subsequently, we also compared the performance of ensemble methods with individual performance of each base-level classifier. In this experiment, we used AdaBoostM1, Vote and StackingC meta-level classifiers of Weka tool as ensemble methods of Cascading, Voting and Stacked generalization techniques respectively. Similarly, we select NaiviBays(NB), Logistic, J48, VotedPerceptron (VP) and SMO as single classifiers and in many studies these classifiers have been used in construction of fault prediction. In order to perform the analysis of ensemble methods, we retrieved twelve datasets of open source projects from PROMISE repository [10]. In this experiment, we used k-fold (k=10) cross-validation and ROC analysis for validation. Besides, we used recall, precision, accuracy, F-value measures to evaluate the performance of ensemble methods and base-level Classifiers. Finally, we observed better performance of ensemble methods as compared to its base-level classifier, and among ensemble

ASWEC ' 15 Vol. II, September 28-October 01, 2015, Adelaide, SA, Australia © 2015 ACM. ISBN 978-1-4503-3796-0/15/09…$15.00
DOI: http://dx.doi.org/10.1145/2811681.2811699.

methods we perceived better performance of StackingC than other ensemble methods.

2. RELATED WORK

In this section, we reviewed and summarized the previous work in table-1, which are related to the association of CK metrics and fault proneness of classes. Besides, we listed the machine learning or data mining techniques in table-1, which have been used by researchers, to construct and validate the fault prediction models, and highlight the significance of CK metrics. In able-1, + symbol refers to significance of metric for corresponding model while – refers to either inverse significance or metric is not significant.

Table-1. Summary of Previous Studies

Author	Method	WMC	DIT	NOC	CBO	RFC	LCOM
Basili et al [11]	Multivariate Logistic Regression Analysis	-	+	+	+	+	-
Pai and Bechta Dugan [12]	Bayesian, Binomial Logistic Regression	+	-	-	+	+	+
Sing et al [21]	Artificial Neural Network	+	-	-	+	+	-
Sing et al [22]	Decision Tree	+	+	+	+	+	+
Sing et al [8]	Support Vector Machine	+	+	+	+	+	+
Malhotra and Jain [20]	Bagging	+	-	-	+	+	+
Rathor and Gupta [19]	RandomForrest	+	+	+	+	+	+
Briand et al [17]	Multivariate Logistic Regression Analysis	-	-	-	+	-	-
Catal et al [18]	Artificial Immune Recognition splines	+	+	+	+	+	+
R. Subramanyam and M.S. Kishnan [7]	Logistic Regression	+	+	-	+	-	-
P. Singh and S. Verma [24]	NaiveBays, J48	+	+	+	+	+	+

In his study, E. Alpaydin [13] describe statistical, syntactical, neural, fuzzy and bayesian as machine learning paradigms, and there is no single machine learning paradigm which producesalways accurate results. Each paradigm dictates a model that comes with some assumptions, and such thing leads to high bias if assumptions do not hold. Moreover, Alpaydin also describe that assumption of single learner may be best suited to different parts of training data if it is ensembled. In data mining, ensemble methods are widely used to improve the accuracy of model by combining the prediction of models. In the domain of software quality, Khosgoftaar et al [14], introduced a new software quality model using majority voting concept to combine the prediction of multiple learners tempted on multiple training datasets. They concluded that multimethods do no yield a significant increase in accuracy. In their study, Vinaykumar and Ravi [15] used two ensemble methods by combining different learners, and reported the success in accurate results of only one. Similarly, E. Kocaguneli et al [16] used ensemble techniques for effort estimation. In their study, they used 90 solo methods with 9 learners and 10 preprocessing options. They select the best 13 solo methods which shows stable performance across multiple datasets and error measures, and generate 12 multimethods. Furthermore, they compare the performance of multimethods with solo methods and observed significant effect of multimethods. S. Dezeroski and and B. Zenko [27] have evaluated several state-of-the-art methods for constructing ensembles of heterogeneous classifiers with stackingC and showed better performance of stacking ensemble methods as compared to other classifiers. Opitz and Maclin [28] have empirically evaluated the performance of boosting ensemble method with 25 classifiers and used neural network and decision trees as classification algorithm. They evaluated their result on 23 datasets. Finally, they concluded the performance of boosting methods depending upon the characteristics of the data set being examined. In their study, Ali and Pazzani [25] show a correlation between the amount of error reduction and the number of individual learners in prediction. Similarly, Hansen and Salamon [26] define that success in classification can be increases with increase in number of voting classifiers.

Currently, there are vast number of ensemble methods available for researchers and practitioners, and several taxonomies in literature regarding the selection criteria for classifiers and categorizing ensemble methods. Such as Sharkey[29] proposed a taxonomy of neural networks with three dimensions that is 1) ensemble's members are competitive or cooperative, 2) ensemble is created either in top-down or bottom-up mode, and 3) combine ensemble is either modular or hybrid. Similarly, Valentini and Masulli [32], Ho [30] dichotomized the ensemble techniques into two categories. First category is relevant to decision optimization methods to find an optimal combination of classifiers, that is selection of fixed set of highly specialized classifiers. Second category is relevant to coverage optimization methods, which refer to generate as set of mutually complementary and generic classifiers to improve predictive performance. Kuncheva [31] proposed layers taxonomy for ensemble methods and its members. The four layers of Kuncheva taxonomy are 1) combination level, which define ways to combine classifier's decisions, 2) classifiers level, refers to selection of base classifiers, 3) feature level, refers to feature subset for classifiers and 4) data level, refers to use of datasets for each base classifier.

In our work, we follow the taxonomy of the Valentini and Masulli [32], Ho [30] for the selection of base classifiers and ensemble methods. Our work is related to evaluate and compare the performance of ensemble methods in predicting the fault proneness classes and present its association with CK metrics.

3. EXPERIMENT SETUP

In our study, we used weka tool to conduct experiment with three ensemble methods (a.k.a meta-level) AdaBoostM1, Vote and StackingC and five base-level classifiers NaiveBays(NB), Logistic, SOM, J48 and Votedperceptron(VP) . The objectives of our study are as follows.

- **Objective-1:** To present an association between CK metrics and fault proneness of classes through ensemble methods.
- **Objective-2:** To compare the performance of ensemble methods with different number of base-level classifiers.
- **Objective-3:** To compare and evaluate the performance of ensemble methods with performance of base-level classifiers.

- **Objective-4:** To investigate the effect of bugs data distribution in a dataset over the performance of ensemble methods.

3.1 Dataset

In this study, we retrieved datasets of twelve open source projects from PROMISE repository [10]. We selected datasets with different bugs data distributions to evaluate and compare the performance of ensemble methods and to achieve our objective-4. Distribution of data is according to the percentage of classes without or with at least one bug. The descriptive statistics of datasets is shown in table-2.

Table-2. Descriptive Statistics of Open Source Projects

Project	Total Classes	Classes with Zero Bug	Percentage (Bugs Free)
Ant-1.7	745	578	77.58
Camel-1.6	965	776	80.41
e-learning	64	59	92.19
Forrest-0.8	32	30	93.75
Jedit-4.3	492	481	97.76
Tomcat	858	781	91.02
Xalan-2.7	909	11	1.21
Xerces-1.4	588	151	25.68
Zuzel	29	16	55.17
Berek	43	27	62.79
Pbean2	51	41	80.39
Velocity-1.6	229	195	85.15

3.2 Base-level Classifiers

In our experiment, we selected those base-level classifiers which have been used by researchers in many studies such as NaiveBayes [24], Logistic [7,11,17], J48 [22,24], Votedperceptron[21] and SMO [8].

Subsequently, in order to achieve objective-2, we did our experiment with two batches of base-level classifiers. First batch comprises of Naivibayes, Logistic and J48 classifiers, while we added two other classifier Votedperceptron and SMO in the second batch. Logistic classifier is used for binary classification either into positive or negative cases of predictions. Votedperceptron is proposed by Collins and is based on the perceptron algorithm of Rosenblatt and Frank. This classifier is preferable when there is noisy or un-separable data and data with the largest margin. J48 is an example to create a uni-variate decision tree and in this technique splitting is performed by using one attribute at internal nodes. Sequential Minimal Optimization (SMO) classifier used in weka as an alternative of Support Vector Machine (SVM). SMO is also called iterative algorithm for solving the optimization problem. NaiveBayes is a simple probabilistic classifier, which is based on applying Bayes' rule with strong independence assumptions.

3.3 Ensemble Techniques and Classifiers

There are two methodologies to ensemble/combine multiple learners namely representational and architectural methodology. In case of representational methodology, when the inputs of all combine classifiers are represented in the same way then it is called uni-representation; otherwise, it is called multi-representation. Similarly, in case of architectural methodology, there are two ways to combine classifier that is multiexpert and multi-stage method. Furthermore, in case of multi-expert architectural methodology, classifiers are combined in a parallel way that is all classifier are trained on all patterns and a separate combiner compute the final decision. Voting, mixture of expert and stacked generalization are common examples of multi-expert method. But in case of multi-stage architectural methodology, combine classifiers work in serial that is next classifier is trained only for pattern which are rejected by previous classifier, for instance cascading is the example of multi-stage method.

In our study, we selected three ensemble methods to combine one or more out of five base-level classifiers to achieve our objective-1 and objective-3. All these classifiers are uni-representational, but two of them, Vote and StackingC, are multi-expert methods while one AdaBoostM1 (i.e. cascading) is multi-stage method.

3.4 Validation Method and Performance Measures

In our study, we used K-fold (k=10) cross-validation and ROC analysis to validate the prediction accuracy of all individual base-level classifier and ensemble methods. Similarly, we evaluate the prediction performance of ensemble methods, and single classifier by using two widely adopted Information Retrieval (IR) metrics that are Precision and Recall [23].We also used F-Measure as an aggregate indicator harmonic mean of precision and recall.

3.5 Result and Discussion

In the shadow of objectives of our study, we have done experiments with two batches of base-level classifiers and observed the following results.

3.5.1 Performance Comparison of Ensemble methods

First, we observed and analyzed the performance of ensemble methods with batch-1 classifiers over 12 datasets. We summarized the result values of Precision (Prec.), Racall(Rec.) and F-Measure(F) and present the variation in performance of classifiers shown in Figure-3. In case of multi-expert ensemble methods such as vote and stackingC in our study, we can see variation in performance of both methods but average performance of StackingC remained significant as compared to Vote method. Similarly, in case of multi-stage methods such as AdaBoostM1_NB, AdaBoostM1_Logistic and AdaBoostM1_J48, the performance of AdaBoostM1_Logistic remained better as compared to other methods. Finally, for batch-1 classifiers, we can observe the overall performance of stacking remain better as compared to all ensemble methods shown in Figure-1. Second, we repeat same pattern with batch-2 classifiers over 12 datasets and results are shown in Figure-2. We observed the average performance of StackingC and AdaBoostM1_Logistic better as in case of batch-1 classifiers.

3.5.2 Performance Comparison of Ensemble method with Base-level Classifiers.

Similarly, to compare the performance of ensemble methods with base-level classifiers of batch-1 and batch-2, we used Precision (Prec.), Racall(Rec.) and F-Measure(F).

We present the variation in the performance results of first batch classifiers and its ensemble methods in Figure-3 (For batch-1) and Figure-4(For batch-2). In both cases, we observed that the performance of ensemble methods either remains same or is better than the performance of all individual classifiers, except for the performance of J48 for tomcat dataset. For batch-1, the performance of J48 (with F-value= 95.6%) classifier remain better as compared to other classifiers but for batch-2 classifiers the

performance of J48 and StackingC (both with F-Value=95.6%) seems to be same.

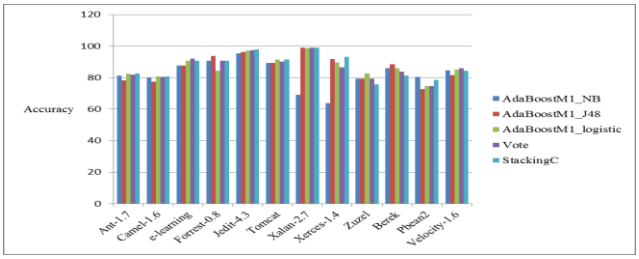

Figure 1. Performance Comparison of Ensemble Methods with Batch-1

3.5.3 Effect of fault distributed data over performance of ensemble methods

To present the effect of bugs data distribution over the performance of ensemble methods, we considered two worst cases of datasets that are Jedit-4.3 (with 97.76% bugs free classes) and Xalan-2.7 (with 1.21% bugs free classes). It is clearly represented in Figure-1 and Figure-2, the performance of all ensemble methods can be observed better for both datasets.

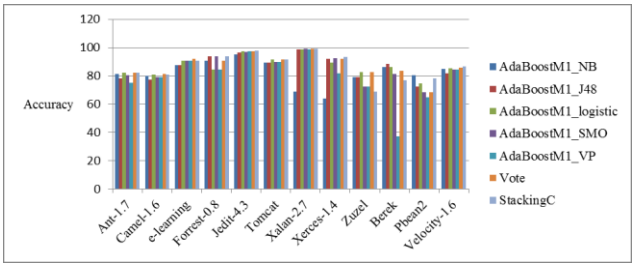

Figure 2. Performance Comparison of Ensemble Methods with Batch-2

It means, the performance of ensemble methods cannot be affected with different data distributions.

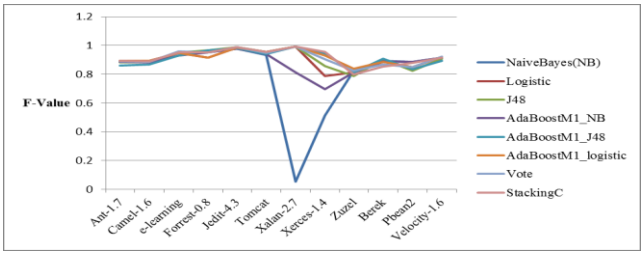

Figure 3. F-Value Comparison of Single and Ensemble methods for Batch-1.

3.5.4 Effect of number of combine classifiers over the performance

It has been defined by Hensen and Salamon [26] that success in classification can be increased with increase in number of voting classifiers. In order to present a relation between the number of classifier and the performance of ensemble methods, we compared the performance of vote and stackingC with two batches of base-level classifiers shown in Figure-5 and Figure-6. In case of vote method, its accuracy remains same for average number of datasets, but over few datasets the performance of vote method with batch-2 classifiers seems better. Moreover, same observation can be seen for StackingC method shown in Figure-6.

4. THREATS TO VALIDITY

In our study, we also find some threats. The first threat is relevant to selection process of base level classifiers for ensemble methods. We selected those classifiers which have been used by researchers for fault prediction in many studies. The performance of ensemble methods can be evaluated with classifiers of different paradigms. The second threat is relevant to our results which are based on the limited number of datasets with different distributions of bugs data. We can further evaluate the performance of ensemble methods by increasing the number of datasets. The Third threat is related to the selection of ensemble methods. We selected only three ensemble methods and concluded results on the basis of their performance.

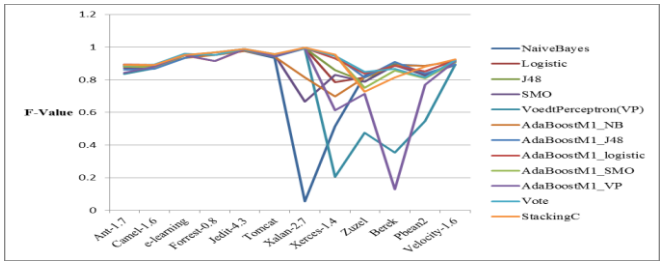

Figure 4. F-Value Comparison of Single and Ensemble methods for Batch-2.

Moreover, we can repeat this study by selecting other ensemble methods such as Multischme and Randomforrest in order to support our assumptions regarding observed results.

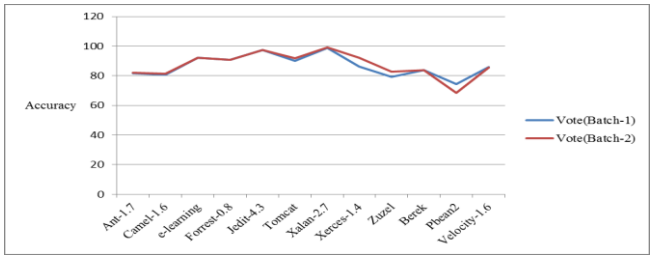

Figure 5. Comparison of Vote Ensemble Method with Batch-1 and Batch-2

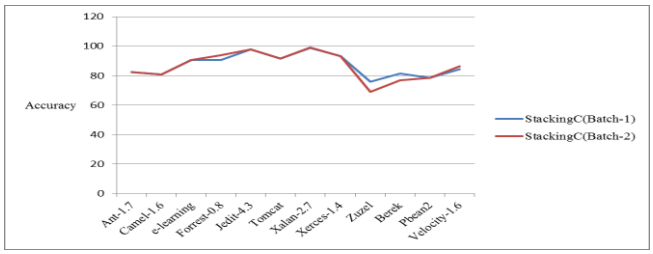

Figure 6. Comparison of StackingC Ensemble Method with Batch-1 and Batch-2

5. CONCLUSION

The objective of this study is to investigate the performance of ensemble methods to present the relationship between software design metrics (CK metrics) and faults processes classes in order to accurately predict faults. In our experiment, we selected two multi-expert ensemble methods that are Vote and StackingC, and one multi-stage method AdaBoostM1 with five base-level classifiers, which have been used by researchers in many studies. Twelve open source projects from PROMISE repository have been used in the study. In the experiment, we used k-fold (k=10) cross-validation and ROC analysis for validation purpose. Results were measured by using recall, precision, accuracy, F-value measures to evaluate the performance of ensemble methods and base-level classifiers. Result shows significant performance improvement of ensemble methods as compared to its base-level classifier, and among ensemble methods we observed StackingC

outperformed other selected ensemble methods, making StackingC an ideal approach for software fault predictions.

6. ACKNOWELDGEMENT

This research is supported in part by the City University of Hong Kong research fund (Project No. 7200354, 7004222).

7. REFERENCES

[1] S.R Chidamber and C.F. Kemerer, A Metrics Suite for OO Design, IEEE Transaction on SE, Vol 20, No 6, Jun 1994.

[2] F.B Abreu and R. Carapuc, Object-Oriented software engineering: measuring and controlling the development Process, Proceedings of the 4th international conference on software quality, vol 186, 1994.

[3] J. Bansiya. and C.G. Davis. A hierarchical model for OO design quality assessment, IEEE Transaction on SE, 2002

[4] L.C Briand, W.L. Melo, J. Wu, "Assessing the Applicability of Fault Proneness Models Across OO Software Projects", IEEE Transaction on SE, 28 (7), pp. 706-720, 2002.

[5] D. Radjenovic et al, Software Fault Prediction Metrics : A systematic literature review, Journal of Information and Software Technology, 2013

[6] S. Kanmani et al, Object-oriented software fault prediction using neural networks, Jounal of Information and Software Technology, 9(5): p. 483-492, 2007.

[7] R. Subramanyam and M.S. Kishnan, Empirical Analysis of CK Metrics for OO Design Complexity: Implication for Software Defects, IEEE Transaction on SE, Vol. 29, 2003

[8] Yogesh Singh, Arvinder Kaur and Ruchika Malhotra, Software Fault Proneness Prediction Using Support Vector Machines, Proceedings of the World Congress on Engineering (WCE), Volume I. 2009.

[9] Zenko B, Todorovski L. and Dzeroski S., A comparison of stacking with MDTs to bagging, boosting and other Stacking methods, In Proceedings of the First IEEE International Conference on Data Mining, pp 669-670, 2001.

[10] M. Jureczko and L. Madeyski, Towards identifying software project clusters with regard to defect prediction. Proceedings of the 6th International Conference on Predictive Models in Software Engineering (PROMISE '10), USA, 2010.

[11] V. Basili, L. Briand, and W. Melo, A Validation of Object-OO Metrics as Quality Indicators, IEEE Transaction on SE, vol. 22, no. 10, pp. 751-761, Oct. 1996.

[12] G.J. Pai and B. Dugan, Empirical Analysis of Software Fault Content and Fault Proneness Using Bayesian Methods, IEEE Transaction on SE, vol. 33, no. 10, Oct 2007.

[13] E. Alpaydm, Techniques for Combining Multiple Learners, Proceddings of Engineering of Intelligent Systems'98 Conference, Vol 2, 6-12, 1998.

[14] T.M Khoshgoftaar, P. Rebours and N. Seliya, Software Quality analysis by Combining Multiple Projects and Learners, Journal of Software Quality Control, vol 55, pp. 119-139, 1997.

[15] M.C.K Vinaykumar and V. Ravi, Software Cost Estimation Using Soft Computing Approaches, Headbook of Research on Machine Learning Applications and Trends: Algorithms, Methods and Techniques, pp. 4999-518, 2010.

[16] E. Kocaguneli, T. Menzies and J.W. Keung, On the Value of Ensemble Effort Estimation, IEEE Transaction on Software Engineering, vol. 38, no. 6, Nov. 2012.

[17] L.C Briand and J. Wust, Empirical studies of quality models in object-oriented systems. Elsevier, Vol 56:98–167, 2002

[18] C. Catal and B. Diri, A systematic review of software fault prediction studies, Journal of Expert Systems with Applications, Vol 36 Issue 4, pp. 7346-7354 May 2009.

[19] S. S. Rathore and A. Gupta, Investigating object-oriented design metrics to predict fault-proneness of software modules, Proceeding os of 6th CSI international conference on software engineering, pp 1–10, 2012.

[20] R. Malhotra and A. Jain, Fault prediction using statistical and machine learning methods for improving software quality, Journal of Information Processing Systems, Vol.8, No.2, June 2012.

[21] Y. Singh , A. Kaur and R. Malhotra , Application of logistic regression and artificial neural network for predicting software quality models. Proceeding of PROFES, 2008.

[22] Y. Singh A. Kaur A and R. Malhotra R, Empirical validation of object-oriented metrics for predicting fault proneness models. Journal of Software Quality , 2010.

[23] R. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval.Reading, MA, USA: Addison-Wesley, 1999.

[24] P. Singh and S. Verma, Empirical investigation of fault prediction capability of object oriented metrics of open source software. Proceeding of the international joint conference on CS and SE, IEEE, pp 323–327, 2012

[25] K.M. Ali and M. J. Pazzani, Error Reduction through learning Multiple Descriptiond, TR 95-39, Deaprtment of ICS University of Ca, Irvine, 1995.

[26] L.K. Hansen and P. Salamon, Neural Network Ensembles, IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol 12, pp. 993-1001, 1990

[27] S. Dezeroski and and B. Zenko. Is Combining Classifiers with Stacking Better than Selecting the Best One, Kluwer Academics Publishers, Manufactured in The Netherland, 54, 255-273, 2004.

[28] D. Opitz and R. Maclin, Popular Ensemble Methods: An Empirical Study, Journal of Artificial Research, 1 1: 169-198, 1999.

[29] A. J. C. Sharkey, Types of multinet system, in: Proc. Int. Workshop on Multiple Classifier Systems (LNCS 2364), Springer, Calgiari, Italy, 2002, pp. 108-117.

[30] T. K. Ho, Data complexity analysis for classifier combination, in: Proc. Int. Workshop on Multiple Classifier Systems (LNCS 2096), Springer, Cambridge, UK, 2001, pp. 53-67

[31] m L. I. Kuncheva, Combining Pattern Classifiers, Wiley Press 2005.

[32] G. Valentini and F. Masulli , Ensembles of learning machines. In R. Tagliaferri and M. Marinaro, editors, Neural Nets, WIRN, Vol. 2486 of Lecture Notes in ComputerScience, Springer, pp. 3-19, 2002.