Journal of Computer Science 2012, 8 (12), 2075-2082
ISSN 1549-3636

doi:10.3844/jcssp.2012.2075.2082 Published Online 8 (12) 2012 (http://www.thescipub.com/jcs.toc)

Enhanced Ensemble Prediction Algorithms for Detecting Faulty Modules in Object Oriented Systems Using Quality Metrics

[1]Neelamegam, C. and [2]M. Punithavalli

[1]Department of Computer Applications,
Sri Venkateswara College of Computer Aplications and Management, Coimbatore, India
[2]Department of Computer Applications,
Ramakrishna Engineering College, Coimbatore, India

Received 2012-02-20, Revised 2012-12-20; Accepted 2012-12-20

ABSTRACT

The high usage of software system poses high quality demand from users, which results in increased software complexity. To address these complexities, software quality engineering methods should be updated accordingly and enhance their quality assuring methods. Fault prediction, a sub-task of SQE, is designed to solve this issue and provide a strategy to identify faulty parts of a program, so that the testing process can concentrate only on those regions. This will improve the testing process and indirectly help to reduce development life cycle, project risks, resource and infrastructure costs. Measuring quality using software metrics for fault identification is gaining wide interest in software industry as they help to reduce time and cost. Existing system use either traditional simple metrics or object oriented metrics during fault detection combined with single classifier prediction system. This study combines the use of simple and object oriented metrics and uses a multiple classifier prediction system to identify module faults. In this study, a total of 20 metrics combining both traditional and OO metrics are used for fault detection. To analyze the performance of these metrics on fault module detection, the study proposes the use of ensemble classifiers that uses three frequently used classifiers, Back Propagation Neural Network (BPNN), Support Vector Machine (SVM) and K-Nearest Neighbour (KNN). A novel classifier aggregation method is proposed to combine the classification results. Four methods, Sequential Selection, Random Selection with No Replacement, Selection with Bagging and Selection with Boosting, are used to generate different variants of input dataset. The three classifiers were grouped together as 2-classifier and 3-classifier prediction ensemble models. A total of 16 ensemble models were proposed for fault prediction. The performance of the proposed prediciton models was analyzed using accuracy, precision, recall and F-measure. When comparing with single classifier systems all the proposed models produced improved classification performance and among the 16 multiple classifier models, the 3-classifier model that combined BPNN, SVM and KNN produced best results. Prediction of software module defection can be improved by combining simple and object oriented metrics with multiple classifiers.

Keywords: Multiple Classifiers, Defect Detection, Ensemble Aggregation, Software Quality Metrics

1. INTRODUCTION

IN today's revolution oriented environment, software systems play a vital role in a wide range of applications, products and services in day-to-day activities. The high usage of software system poses high quality demand from users, which results in increased software complexity. In order to meet this increasing quality demand, an engineering discipline called "Software Quality Engineering (SQE)" is used. SQE

Corresponding Author: Neelamegam, C., Department of Computer Applications, Sri Venkateswara College of Computer Aplications and Management, Coimbatore, India

consists of many quality assurance activities like testing, fault prevention, fault inspection, fault tolerance, formal verification and fault prediction (Lee *et al*., 2009). Testing, a frequently used SQE task to identify faults in software systems, has the drawback of being time consuming and expensive. This necessitates the need for alternative methods. Fault prediction, another sub-task of SQE, is designed to solve this issue and provide a strategy to identify faulty parts of a program, so that the testing process can concentrate only on those regions. This will improve the testing process and indirectly help to reduce development life cycle, project risks, resource and infrastructure costs.

Fault prediction models can be either process oriented or product oriented. Process oriented models focus on development and maintenance while product oriented models focus on design and usability issues. Software design is a task in software life cycle and is involved in developing alternatives, comparing them and selecting one alternative that provides maximum advantage in terms of cost and time. Inspite of careful design, a software system may have faults or bugs in design because of bad design practices. They include problems ranging from high-level (high complexity) to low-level (low complexity) problems. Design defects arise because often software design decays after some years and changes are applied in hasty manner and affect software tasks like maintenance, reusability and comprehensibility.

Usage of software metrics to evaluate the quality of software design has attracted software industries as they help to assess large software system quickly at low cost. Several studies have focused on evaluating the usefulness of software metrics to predict software design faults. These techniques can be loosely categorized as statistical techniques, structural patterns based techniques, software metrics based techniques, formal/relational concept analysis and software inconsistency management techniques. Classification, a frequently used data mining technique, has found wide usage in a range of problem domains such as finance, medicine, engineering, geology and physics. Combining software metrics and classification is a methodology that has gained attention recently. This study proposes a methodology that combines software metrics and a suite of classifiers (ensembling) to design a fault prediction model.

Existing design metrics include traditional simple metrics, program complexity metrics, CK Metrics and Mood Metrics all of which have been extensively used in prediction of faulty modules both in general non-Object Oriented (non-OO) systems and Object Oriented (OO) systems. The non-OO metrics have the disadvantage that they have no firm theoretical base for demonstrating normal fault prediction behavior (Babu and Parvathi, 2011) and do not consider object oriented paradigms like inheritance, encapsulation and passing of message. This makes them unsuitable for OO systems. Because of these reasons, the traditional metrics are normally combined with OO metrics while using with OO systems. The study combines traditional simple metrics with OO-metrics MOOD and MK metrics. A feature selection algorithm is used to select only those features that are relevant for fault detection during classification. To analyze the performance of these metrics on fault module detection, the study proposes the use of ensemble classifier that uses three frequently used classifiers, Back Propagation Neural Network (BPNN), Support Vector Machine (SVM) and K-Nearest Neighbour (KNN). A novel classifier aggregation method is also proposed.

2. MATERIALS AND METHODS

The main task of classifiers in fault prediction model is to identify software models as either defective or defect-free modules by performing binary classification. The proposed ensemble model fuses the efficiency of several single binary classifiers to improve the prediction efficiency. In a binary classification model the input data for a classification task is a collection of software design metrics collected from one or more object oriented software projects. The collected metrics are arranged in row-wise fashion (records). Each record is denoted as a set (X, y) where X is the set of metric values and y is the designated class label also known as target attribute. The binary classifier maps the input metrics to any of the two labels, defective and defect-free. Binary classification can be performed using several models like naïve-bayes, artificial neural network, support vector machine, decision trees and k-nearest neighbor.

According to (Park, 2010), when a perfect set of feature metrics that best describe the software set is given, the accuracy of the resultant classification depends on the classifier adopted. Thus, selection of an appropriate classifier is crucial and challenging task while designing the prediction model. One way to accommodate this challenge is by the use of multiple classifiers (Neeba and Jawahar, 2009) and then fuse their results. Using multiple classifiers (either different types of classifiers or different instantiations of the same classifier) improve the success rate of the prediction model. The concept is termed as fusion or

ensemble classification. According to (Oza and Tumer, 2008), intuitively, fusion classification allows the different needs of a difficult problem to be handled by classifiers suited to those particular needs. Mathematically, fusion classifier provide an extra degree of freedom in the classical bias/variance tradeoff, allowing solutions that would be difficult (if not impossible) to reach with only a single classifier. A general model of fusion classification is presented in **Fig. 1**.

The accuracy of a fusion prediction model depends on several factors like (i) Classifier Details (number of classifiers and type of classifier) (ii) Metrics used by the individual classifiers (iii) Partitioning method (Training and Testing sets) (iv) the aggregation method and (v) Type of training. The techniques and methods used for each of the above factor are discussed below.

2.1. Classifier Details

Three classifiers are considered, namely, Feed Forward Back Propagation Artificial Neural Network (BPNN), Support Vector Machine (SVM) and K-Nearest Neighbour (KNN). BackPropagation Neural Network (BPNN) described by (Bryson and Ho, 1969) gained recognition only after 1974 (Alpaydin, 2004) is an Artificial Neural Network (ANN) where input data moves in only one direction, forward, from the input nodes, through the hidden nodes, to the output nodes. The BPNN is the most commonly used ANN where given a network with a fixed set of units and interconnections, employs rules that attempts to minimize the Mean Squared Error (MSE) between the network output values and the target values for these outputs. The BPNN training algorithm consists of two phases: Propagation and weight update (www.wikipedia.org). The propagation phase consists of forward and backward propagation. Forward propagation generates the propagation's output activations, while backward propagation uses the training dataset to generate the deltas of all output and hidden neurons. The weight update phase multiply its output delta and input activation to get the gradient of the weight and then bring the weight in the opposite direction of the gradient by subtracting a ratio of it from the weight. This ratio influences the speed and quality of learning and is called the learning rate. The sign of a weight indicates where the error is increasing. Phases 1 and 2 are repeated until performance of the network is satisfied. In this study, the training is stopped at the minimum of the Mean Squared Error (MSE) on the validation set. The MSE is the average error over all samples in the set. During experimentation, it was found that after 150 cycles, the MSE value reached its minimum (0.67 and 0.73 for training and testing respectively) and generalized the network. After this point, performance of BPNN decreased.

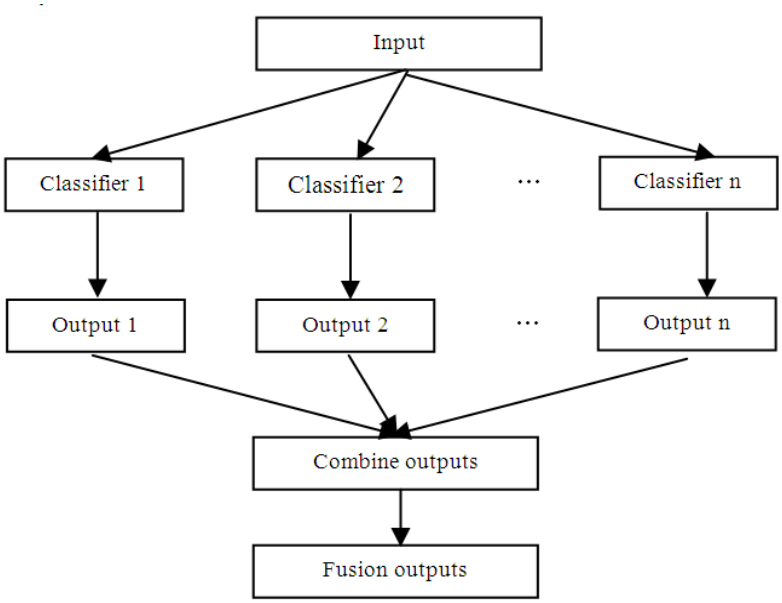

Fig. 1. Fusion classifier model

The second classifier used is Support Vector Machine (SVM), which given a set of input data and predicts, for each given input, which of two possible classes the input is a member (Gondra, 2008). This makes SVM a non-probabilistic binary linear classifier. Given a set of training examples, each marked as belonging to one of two categories (faulty or not-faulty), an SVM training algorithm builds a model that assigns new examples into one category or the other. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on. .

The third classifier considered is the K-Nearest Neighbour Classifier (Cover and Hart, 1967), which has the advantage of achieving consistently high performance, without a priori assumptions about the distributions from which the training examples are drawn. The k-NN classifier considers the k nearest points of a data point and assigning the sign of the majority. It is common to select k small and odd to break ties (typically 1, 3 or 5). Larger k values help to reduce the effects of noisy points within the training data set and the choice of k is often performed through cross-validation. It is a non-parametric classification model, where the training dataset is used to classify each member of a "target" dataset. The algorithm (Purohit *et al*., 2011) is given below:

- For each row (case) in the target dataset (the set to be classified), locate the k closest members (the k nearest neighbors) of the training dataset
- A Euclidean Distance measure is used to calculate how close each member of the training set is to the target row that is being examined
- Examine the k nearest neighbors to find the class that is very near to the category and assign this category to the row being examined
- Repeat this procedure for the remaining rows (cases) in the target set
- The best choice of k depends upon the data; generally, larger values of k reduce the effect of noise on the classification, but make boundaries between classes less distinct. In experiments, a value of 3 was set to 'k' (k = 3)

2.2. Metrics Used

In this study, the four proposed metrics are combined with existing metrics during fault prediction. Twenty existing metrics, namely, simple metrics, Mood Metrics, CK Metrics and Program Complexity Metrics (PCM), were selected for each module. These metrics were selected because of their wide usage in fault detection. Apart from this, the four proposed metrics explained in the previous section are also used. **Table 1** summarizes the selected metrics.

2.3. Dimensionality Reduction

Dimensionality reduction is performed to avoid the complexity and degradation introduced by the phenomenon called "Curse of Dimensionaility". A Dimensionality reduction algorithm aims to reduce the dimension by retaining only those data that are most relevant for the classification task. For this purpose, this study uses Sensitivity Analysis of data. Sensitivity analysis analyzes the importance of each input data in relation to a particular model and estimates the rate of change of output as a result of varying the inpu values. The resulting estimates can be used to determine the importance of each input variable (Saltelli *et al*., 2008). This study adopts the Sensitivity Casual Index (SCI) proposed by (Goh, 1993) and can be calculated as follows. For a classifier, given a set of input Vectors, $\{V_i, n \leq i \geq 0\}$, where V_i belongs to the set of metric values collected from the input dataset with 'd' dimensions, for a classifier with single output $Y = f(x_i)$, the SCI for each input dimension is calculated using Equation 1:

$$SCI_j = \sum_{i=1}^{n} | f(V_i) - f(V_i + \Delta_{ij}) | \quad (1)$$

where, $|.|$ denotes absolute value and Δ_{ij} is a small constant added to the j_{th} component V_j of V_i.

2.4. Normalization

This step is used to normalize each input to the same range and makes sure that the initial default parameter values are appropriate and every input at the start has equal important. Further, normalization of input data is performed to improve the training process of the classifier. A common practice followed is to perform normalization by estimating the upper and lower bounds for each metric value and then scale them using Equation 2:

$$V_j^{'} = \frac{V_j - \min(V_j)}{\max(V_j) - \min(V_j)} \quad (2)$$

Table 1. Design metrics

Simple metrics	RC (Response for a Class)
Total number Of Lines (LOC)	Lack of Cohesion in Methods (LCM)
BR (Number of methods)	Mood Metrics
NOP (Total Number of Unique Operators)	Method Hiding Factor (MHF)
NOPE (Total Number of Unique Operands)	Attribute Hiding Factor (AHF)
RE (Readability with Comment percentage)	Method Inheritance Factor (MIF)
VO (Volume)	Attribute Inheritance Factor (AIF)
CK Metrics	Polymorphism Factor (PF)
WMC (Weighted Methods per Class)	Coupling Factor (CF)
DIT (Depth of Inheritance Tree)	Program Complexity Metrics
NC (Number of children)	Cyclomatic Complexity (CC)
COC (Coupling between object classes)	Fan-In Fan-Out Complexity - Henry's and Kafura's (FI-FO)

where, $V_j^{'}$ is the normalized or scaled value, $\min(V_j)$ and $\max(V_j)$ are the maximum and minimum bounds of the metric 'j' from 'n' observations respectively. The result of normalization thus, maps each input value to a closed interval [0, 1].

2.5. Partitioning Method

Four methods are used in this work to generate different variants of input dataset that can be used as input to classifiers. The selected methods are Sequential Selection (SS), Random Selection with No Replacement (RSNR), Selection with Bagging (SBA) (Breiman, 1996) and Selection with Boosting (SBO) (Freund and Schapire, 1996). The resultant dataset is then partitioned into training and testing set using hold method. The holdout method randomly partitions the dataset into two independent sets, training and testing. Generally, two-thirds of the data are allocated to be the training set and remaining one-third is allocated as test set. The method is pessimistic because only a portion of the initial data is used to derive the model.

2.6. Proposed Aggregation Method

The study uses a combination of majority voting and weighting scheme for aggregating the results of the classifiers. The modified majority vote scheme that combines weighting scheme is explained below. Let the decision of the i^{th} classifier be defined as $d_{t,j} \in \{0, 1\}$, t = 1, …, T and j = 1, …, C, where T is the number of classifiers and C is the number of classes. If the i^{th} classifier chooses class ω_j, then $d_{t,j} = 1$ and 0, otherwise. In majority voting scheme, a class ω_j is chosen, if Equation 3:

$$\sum_{t=1}^{T} d_{t,J} = \max_{j=1}^{c} \sum_{t=1}^{T} d_{t,j} * w^t \quad (3)$$

Here w^t is the weight assigned to the classifier t and is calculated using Kuncheva (2004) method (Equation 4):

$$w^t = \log \frac{p^t}{1 - p^t} \quad (4)$$

2.7. Type of Training

There are various methods used while training a multiple classifier system. They are, (i) Training of the individual classifiers and applying aggregation that does not require further training (ii) Training of the individual classifiers followed by training the aggregation (iii) Simultaneous training of the whole scheme. The present scheme uses the first method where after training the individual classifier, further classification is not required. This method is selected because the fusion classification depends on the result of the individual classifier.

3. RESULTS

The proposed fault-detection classifier systems using software metrics was developed using MATLAB 2009 and all the experiments were conducted on a Pentium IV machine with 4GM RAM. The NASA IV and V Facility MDP data (http://mdp.ivv.nasa.gov/repository.html), consists of error data from several projects. This study uses KC1 project, which consist of records related to a real-time project written in C++ consisting of 43000 LOC. The dataset has a total of 1571 modules out of which 319 are faulty modules while 1252 are non-faulty modules. The feature vector created has 20 dimensions each representing one selected metric. This vector was first normalized to an interval [0, 1] to ensure that all the 20 values have equal importance. Dimensionality reduction was next performed on this set to select discriminating metrics by calculating SCI of

each input dimension over the entire normalized dataset with $\Delta = 0.1$. After calculation of SSI, the metrics were arranged in descending order of SSI and the top 15 metrics were selected. The resultant feature vector, after dimensionality reduction consists of LOC, BR, RE, WMC, DIT, NC, COC, RC, LCM, MHF, AHF, MIF, AIF, PF and CF. It can be seen that the resultant reduced dataset consists of only those metrics which has impact on complexity measure. The reduced dataset with 15 metrics is then divided into training (943 modules) and testing (628) datasets.

Four classification performance metrics were used during evaluation. They are accuracy, precision, recall and F-measure, which are derived from the confusion matrix. A 10-fold cross validation method was used with all experiments. The performance of the single classifiers was compared with that of ensemble classifiers. For SVM classifier, the regularization parameter was set to 1, the kernel function used was Gaussian and bandwidth of the kernet was set to 0.5. For K-NN classifier, k was set to 3. For BPNN classifier, 2 hidden nodes with learning rate of 0.2 were used. T-Test was performjed at 95% confidence level (0.05 level) to analyze the significant difference between SVM and BPNN, SVM and KNN. The T-test method adopted was proposed by Nadeau and Bengio (2003). This method was adopted because it is more suited for classifiers adapting 10-fold cross-validation method (Dietterich, 1998). The traditional student 't' test, method produces more false significant differences due to the dependencies that exists in the estimates. Further, the affect of the proposed metrics in classification performance is ascertained by running the experiments with the existing metric set containing 20 metrics and analyzing the classification accuracy. From the three single classifiers, 16 ensemble prediction models as listed in **Table 2** were built. Models 1-3 are single classifiers BPNN, KNN and SVM. Models 4-15 are single classifiers with different variants created using SS, RSNR, SBA and SBO techniques. Models 16-19 (2- and 3- classifiers) use full normalized data set and do not use of SS, RSNR, SBA and SBO techniques.

Table 3-5 shows the 1-classifier, 2-classifer and 3-classifier PEM performance of the proposed BPNN, KNN and SVM based ensemble predictors based on Accuracy, Precision, Recall and F Measure. To analyze the advantage obtained by the proposed predictors the proposed models are compared with their traditional single classifier counterparts. In these tables, SD denotes the standard deviation and the column Sig denotes the status of significance.

Table 2. Proposed Prediction Ensemble Models (PEM)

Single classification models: 1. BPNN, 2. KNN, 3. SVM

	1-Classifier PEM	
BPNN	KNN	SVM
4. BPNN + SS	8. KNN + SS	12. SVM + SS
5. BPNN + RSNR	9. KNN + RSNR	13. SVM + RSNR
6. BPNN + SBA	10. KNN + SBA	14. SVM + SBA
7. BPNN + SBO	11. KNN + SBO	15. SVM + SBO
2-Classifier PEM	**3-Classifier PEM**	
16. BPNN + KNN	19. BPNN + KNN + SVM	
17. KNN + SVM		
18. BPNN + SVM		

Table 3. Performance of BPNN based ensemble prediction models

	Accuracy			Precision			Recall			F Measure		
Model	Mean	SD	Sig	Mean	SD	Sig	Mean	SD	Sig	Mean	SD	Sig
1	77.38	3.562		80.12	2.981		84.01	3.015		82.02	3.298	
4	84.26	2.1	Yes (+)	85.74	2.441	Yes (+)	89.87	2.64	Yes (+)	87.76	2.221	Yes (+)
5	81.92	0.96	Yes (+)	84.11	1.569	No (-)	88.14	1.01	Yes (+)	86.08	0.674	Yes (+)
6	82.74	1.703	Yes (+)	85.18	2.258	No (-)	88.57	1.27	Yes (+)	86.84	1.188	Yes (+)
7	82.16	1.201	Yes (+)	84.76	2.697	No (-)	88.22	1.18	Yes (+)	86.46	1.047	Yes (+)
16	89.91	1.236	Yes (+)	97.36	0.899	Yes (+)	93.44	0.587	Yes (+)	95.36	0.745	Yes (+)
18	94.55	1.579	Yes (+)	98.93	0.371	Yes (+)	92.94	1.574	Yes (+)	95.84	0.361	Yes (+)
19	96.17	1.314	Yes (+)	99.94	0.012	Yes (+)	94.16	1.122	Yes (+)	96.96	0.202	Yes (+)

Table 4. Performance of KNN based ensemble prediction models

	Accuracy			Precision			Recall			F Measure		
Model	Mean	SD	Sig	Mean	SD	Sig	Mean	SD	Sig	Mean	SD	Sig
2	84.98	2.416		89.72	0.126		95.42	0.124		92.48	0.397	
8	89.26	1.841	Yes (+)	91.76	0.441	Yes (+)	96.42	0.441	Yes (+)	94.03	0.241	Yes (+)
9	87.89	0.306	Yes (+)	89.97	0.314	Yes (+)	95.89	0.467	No (-)	92.84	0.978	Yes (+)
10	88.98	0.566	Yes (+)	91.12	0.876	Yes (+)	96.16	0.978	No (-)	93.57	0.618	Yes (+)
11	87.81	0.382	Yes (+)	90.76	0.924	Yes (+)	96.02	0.997	No (-)	93.32	0.344	Yes (+)
16	89.91	1.236	Yes (+)	97.36	0.899	Yes (+)	93.44	0.587	Yes (+)	95.36	0.745	Yes (+)
17	90.26	1.077	Yes (+)	97.94	0.821	Yes (+)	92.67	0.687	Yes (+)	95.23	0.798	Yes (+)
19	96.17	1.314	Yes (+)	99.94	0.012	Yes (+)	94.16	1.122	Yes (+)	96.96	0.202	Yes (+)

Table 5. Performance of SVM based ensemble prediction models

	Accuracy			Precision			Recall			F Measure		
Model	Mean	SD	Sig	Mean	SD	Sig	Mean	SD	Sig	Mean	SD	Sig
3	90.62	1.161		90.34	0.04		98.43	0.068		94.21	1.014	
12	93.99	1.991	Yes (+)	92.34	1.461	Yes (+)	98.77	0.241	Yes (+)	95.45	0.166	Yes (+)
13	92.96	0.989	Yes (+)	91.27	0.785	Yes (+)	98.01	0.114	No (-)	94.52	0.045	Yes (+)
14	93.41	1.562	Yes (+)	92.08	1.318	Yes (+)	98.54	0.981	No (-)	95.2	0.681	Yes (+)
15	93.16	1.199	Yes (+)	91.76	0.978	Yes (+)	98.12	0.457	No (-)	94.83	0.457	Yes (+)
17	90.26	1.077	Yes (+)	97.94	0.821	Yes (+)	92.67	0.687	Yes (+)	95.23	0.798	Yes (+)
18	94.55	1.579	Yes (+)	98.93	0.371	Yes (+)	92.94	1.574	Yes (+)	95.84	0.361	Yes (+)
19	96.17	1.314	Yes (+)	99.94	0.012	Yes (+)	94.16	1.122	Yes (+)	96.96	0.202	Yes (+)

In the Sig column, 'Yes' denotes that there is a significance performance difference between single prediction model and the corresponding ensemble prediction model, while a 'No' represents insignificant performance. A '+' sign at the end denotes that ensemble prediction model has outperformed the corresponding single prediction model, while '– ' sign denotes the opposite.

4. DISCUSSION

From the results it could be seen that the application of ensembling concept to predict faulty modules in object oriented systems has improved the performance of the prediction classifiers. Among the four data selection algorithms, the Sequential Selection method produced significant improvement to classification performance. The statistical result of models 5, 6 and 7 showed negative insignificance with respect to precision when compared with its base model. But, the recall parameter, which plays more important role in classification, achieved positive significant difference. The same models when compared with F measure (which is amalgamation of precision and recall) also showed significant difference and outperformed the base model. While comparing the three classifiers, the performance of SVM-based prediction models is better when compared with BPNN and KNN. While considering the number of classifiers, the 3-classifier ensemble model ranked first when compared with all other models. Thus, among the 16 proposed models, the best performance was produced by the model that used fusion techniques that combines BPNN, KNN and SVM classifiers.

5. CONCLUSION

This study analyzes the application of ensemble classification prediction algorithm to predict faulty modules in object oriented systems using design metrics. For this purpose, 20 metrics that are related to with the complexity factor of a system were selected. Sensitivity index was used to select relevant metrics for classification after normalization. Three classifiers, namely, BPNN, SVM and KNN with four data selection algorithms, namely, SS, RSNR, SBA and SBO, were used to generate ensemble classifiers. These classifiers are termed as 1-classifier ensemble prediction models. The three classifiers were grouped together to form four ensemble models and these were identified as 2-classifier and 3-classifier prediction ensemble models. Thus, a total of 16 ensemble models were proposed for fault prediction in OO systems using design metrics. The performance was analyzed using accuracy, precision, recall and F-measure. When comparing with single

classifier systems all the proposed models produced improved classification performance and among the 16 models, the 3-classifier model that combined BPNN, SVM and KNN produced best results. Future research is planned in the direction of development of new design metrics and their use with the proposed classifiers.

6. REFERENCES

Alpaydin, E., 2004. Introduction to Machine Learning. 1st Edn., MIT Press, Cambridge, Mass., ISBN-10: 0262012111, pp: 415.

Babu, S. and R.M.S. Parvathi, 2011. Design dynamic coupling measurement of distributed object oriented software using trace events. J. Comput. Sci., 7: 770-778. DOI: 10.3844/jcssp.2011.770.778

Breiman, L., 1996. Bagging predictors. Mach. Learn., 24: 123-140. DOI: 10.1007/BF00058655

Bryson, A.E. and Y.C. Ho, 1969. Applied Optimal Control: Optimization, Estimation and Control. 1st Edn., Blaisdell Publishing Company, Waltham, pp: 481.

Cover, T. and P. Hart, 1967. Nearest neighbor pattern classification. IEEE Trans. Inform. Theory, 3: 21-27. DOI: 10.1109/TIT.1967.1053964

Dietterich, T.G., 1998. Approximate statistical tests for comparing supervised classification learning algorithms. Neural Comput., 10: 1895-1923. DOI: 10.1162/089976698300017197

Freund, Y. and R.E. Schapire, 1996. Experiments with a new boosting algorithm. Proceedings of the 13th International Conference on Machine Learning, (ML' 96), Morgan Kaufmann, pp: 148-156.

Goh, T.H., 1993. Semantic extraction using neural network modelling and sensitivity analysis. Proceedings of the International Joint Conference on Neural Networks, Oct. 25-29, IEEE Xplore Press, pp: 1031-1034. DOI: 10.1109/IJCNN.1993.714088

Gondra, I., 2008. Applying machine learning to software fault-proneness prediction. J. Syst. Software, 81: 186-195. DOI: 10.1016/j.jss.2007.05.035

Kuncheva, L.I., 2004. Combining Pattern Classifiers: Methods and Algorithms. 1st Edn., John Wiley and Sons, Hoboken, New Jersey, ISBN-10: 0471660256, pp: 300.

Lee, J.S., O. Jeong and J. Ryu, 2009. Performance evaluation framework for software quality engineering. Proceedings of the 9th International Conference on Quality Software, Aug. 24-25, IEEE Xplore Press, Jeju, pp: 438-443. DOI: 10.1109/QSIC.2009.65

Nadeau, C. and Y. Bengio, 2003. Inference for the generalization error. Mach. Lear., 52: 239-281. DOI: 10.1023/A:1024068626366

Neeba, N.V. and C.V. Jawahar, 2009. Empirical evaluation of character classification schemes. Proceedings of the 7th International Conference on Advances in Pattern Recognition, Feb. 4-6, IEEE Xplore Press, Kolkata, pp: 310-313. DOI: 10.1109/ICAPR.2009.41

Oza, N.C. and K. Tumer, 2008. Classifier ensembles: Select real-world applications. J. Inform. Fusion, 9: 4-20. DOI: 10.1016/j.inffus.2007.07.002

Park, D.C., 2010. Image classification using partitioned-feature based classifier model. Proceedings of the IEEE/ACS International Conference on Computer Systems and Applications, May 16-19, IEEE Xplore Press, Hammamet, pp: 1-6. DOI: 10.1109/AICCSA.2010.5586971

Purohit, A., K. Arora, N. Pandit, S. Sharma and S. Bansal, 2011. Genetic algorithm for classification of web documents. Int. J. Comput. Sci. Appli.

Saltelli, A., K. Chan and E.M. Scott, 2008. Sensitivity Analysis. 1st Edn., John Wiley and Sons, New York, ISBN-10: 0470743824, pp: 494.