Analysis of Software Fault and Defect Prediction by Fuzzy C-Means Clustering and Adaptive Neuro Fuzzy C-Means Clustering

Pushpavathi T.P, Suma V, Ramaswamy V

Abstract— Faults are related to failures and they do not have much power for indicating a higher quality or a better system above the baseline that the end-users expect.The system faults are the defects that brim in executable files. Conventional approaches employ the experts to navigate directly into the source code errors. However expansion in system size grew the complexity of task exponentially and generated a scope for new methods in fault classification.Experimental studies have shown that miniature bugs are reason of faults. In a considerable size of system the faulty labels and non-faulty labels are marked during modular phase. This paper presents the adaptive-neuro fuzzy c-means clustering for fault classification via fuzzy c-means clustering.Experimental studies confirmed that only a small portion of software modules cause faults in software systems.The NASA pc1 database is used for experiments and the results in this approach is enhanced than previous clustering based approaches.

Index Terms— Adaptive-neuro fuzzy c-means clustering, Data Classification, Defect Prediction, Fault Prediction, Fault proneness, Fuzzy clustering, Software Project Success.

1 Introduction

SOFTWARE reliability is the essential parameter in decision of software standard classification [1]. The software fault prediction technique characterizes the performance of software's performance in dependent field's application. The program attributes is scaled by quantitative representation from software metrics that play crucial role in detecting the software quality based on evaluation parameters [2]. Perlis et al. in [2] experimented and found relationship between attributes modified by faults and complexity metrics direct and true in test and validation [3]. In late 20th century the immense research in this field was oriented towards the prediction of relationship between faults detected and complexity metrics. Finding the necessary metrics depends on multiple variable models [4] in addition to fault size.

A software system is the composition of number of modules dependent on each other. Any module with faults in its functionality adverse the output and lowers its reliability. In this scenario, the detection of faulty modules in early stage (development stage) is mandatory to minimize faults in operation phase. Hence, the systems are classified in two categories i.e. with faulty/non-faulty modules in their testing phase. This classification diverts the focus to neutralize faulty sections to achieve high reliability and accuracy.

- *Pushpavathi T.P, JAIN University, Bangalore, India. E-mail: acepushpa@yahoo.co.in*
- *Suma V, Dayanandsagar College of Engineering, Bangalore, India. E-mail: sumavdsce@gmail.com*
- *Ramaswamy V, SASTRA University, Srinivasa Ramanujan Centre,Kumbakonam,Tamilnadu,India, E-mail: researchwork04@gmail.com*

A software fault or error is reason of failure in execution stage. The error message at each stage of executing the program indicates the fault in programming. Generally speaking the errors are the logical errors in software program. The prediction models of software fault proneness technique estimate the amount of faulty modules in a program. The software metrics are the attributes for process, execution and product of the software system.Various other attributes like defect density normalized work, fault proneness, maintainability, reusability etc. determines the quality of software.
Software metrics data are responsible to report about the specific attributes for the calculation of modules or functions for the whole software. These attributes are the inputs for self-learning model when co-related with weighted error or defect data. The learning mode is a system that employs the previous results of performance measure to upgrade itself so as to enhance the performance in comparison with previous results. The learning system is modeled in two phases of categorization in its working mechanism i.e. the testing dataset and the training data set. Some predictor functions in software fault proneness systems simulate the Multilayer Perceptron and Decision Tree algorithm for training and evaluation of effects with respect to testing data set.

2 Related Review

In software systems failures, faults and defects are terms co-related to each other with visible difference in their definitions [5]. However for researchers, according to IEEE Software Glossary for Terminologies of Software Engineering, a distinction is mandatory in these terms [7]. Error is considered to be human mistake (in reference with Glossary) responsible to generate undesired results. The error hinders the software or its component to produce desirable output constrained to its parameters within specific requirements. A fault [5] is the defect of the system proposed in terms of system engineering and hardware [7] and is parallel to error responsible for glitch in simulations for corresponding platform. In this research a fault in the system is associated with mistakes that are diagnosed either in development phase or by testing units and system levels. In software systems, a term "software failure" is conventional for the systems with inconsistent output, but in this paper term "fault" or "faulty modules" is accounted for the reason that the anomalies responsible for failure are tracked in coding phase itself. The faulty modules of the dataset considered in this paper are pre-released faults [6].

Software systems are versatile to most of technological/non-technological advances of the society. Creating software with 100% efficiency is an imaginary thought hence software fault prediction got attention by a variety of researchers. Software quality is verified in distinct parameters like formal verification, inspection, fault tolerance and testing. Automated methods employs software metrics [8] [9] of previous versions to develop a prediction model. Fault prediction is the union of two steps i.e. training and prediction. Training phase is based on the metrics (method-level or class metrics) model developed by evaluation parameters of earlier software versions. This metrics predicts the fault-proneness of modules liable to be faulty for the upgraded versions of software. In literature survey the methods described are supervised methods that have classified data of faulty modules. Today the software fault detection systems are designed with 85% accuracy in their analysis subjected to dataset considered and evaluation parameters taken into account. This level of accuracy is applicable in economical world of software industry.

The fault prediction techniques are sourced by means of historical data. Research work suggest that the system under development is prone to fault if the software metrics measures similar properties of software and faulty modules developed and sensed previously in same environment [10]. The conventional applications provide us a platform for fault proneness and methodology for techniques for fault prediction and mitigation. The supervised learning methods stated in literature are combined study of fault data and software metrics that implements different learning algorithms. In late 90's and starting years of 21st century many techniques surfaced as the solution for this problem. Neural networks [11, 12, 13], Genetic Algorithm [14] were developed for large datasets to generate the generalized relation. Dampster-Shafer Networks [15] believed the data can be treated as faulty depending on combined evidence from different sources. Naïve Bayes [16, 17] stated that the data may be considered as faulty irrespective of its nature, if its parameters match the predefined threshold for faulty systems. Decision trees [18] map the observations to predict the possible outcomes. Artificial Immune Systems [19] is an appreciated algorithm as it defines the cognitive patterns once trained to update data. Support Vector Machines [20, 21] employs associated learning algorithm used in classification to recognize and analyze datasets. Case-based Reasoning [22] keeps the track of solutions and detects problems based on these results. Ant-Colony Optimization [23] is probabilistic approach to minimize the computational problems by graphical method. Fuzzy logic [24] is the clustering algorithm to collect fault data with high accuracy. Basili et al. 1996 [25] proposed the logistic regression that employs domain specific knowledge for determination of software metrics (input) and software fault-proneness (output) relation.

The algorithms are evaluated for their strength and weaknesses by evaluation parameters in experimental setups. Recently Metrics Data Program NASA IV&V facility made its data available for public research. Yue Jiang [26] in 2008 studied the model evaluation techniques in software engineering studies to comprehend strength and weakness of performance evaluation techniques. Authors in [27] stated that, the comparison of fault-prone models is a multi dimensional problem. No single model or the modeling technique proves to be the best for all possible uses in software quality assessment. The supervised learning approaches are constrained to the knowledge of faulty data. In case the faulty data is undetermined, new methods of classification surfaced. A Ripper Down Rule (RIDOR) algorithm [27] [28] generates a default rule, for the exceptions to be generated in a tree like structure until according to rule set all training instances are classified correctly. Hassan Najadat and Izzat Alsmadi [29] modified this rule in terms of accuracy and effectiveness (i.e. generating less number of rules). The enhanced RIDOR algorithm imports benefits of CLIPER algorithm and RIDOR. The attributes are encoded as symbols and are compacted or merged to only two. The method proved to be efficient on a number of datasets considered by authors. Yue Jiang, Bojan Cukic and Tim Menzies [30] experimented to find the suitability of metrics in early development to find fault prone software module. Also authors demonstrated a predictive module using metrics to characterize textual requirements. The model was tested on 5 datasets. The early lifecycle metrics plays significant role in project management but the requirement metrics is unable to predict data itself.

3 Methodology

The Fuzzy C-Means clustering for classification of faults in software fault prediction is conventional approach. The Fuzzy C-Means clustering method [31] is the reference of adaptive method that improve the performance index in faults classification sector for software systems. The enhancement of this method is the collective co-relation of feed-forward neural network [33] with fuzzy c-means to outperform the assignment of mean deviation and absolute error to a cluster to minimize the distance for fault prediction.

The data of PC1 (NASA) is input to the system. For the Fuzzy C-Means to cluster the faulty data requires pre-processing of data to minimize time consumption. The output of C-Means clustering is stored for comparison. The output of C-Means is fed to adaptive Neuro-Fuzzy C-Means clustering algorithm. The algorithm tuned by Neural Network trains the data and improve performance index. The output of algorithm is compared with output of Fuzzy C-Means in terms of accuracy, reliability, RMSE and MAE.

TABLE 1

PC1 DATASET (SOURCE: NASA)

	Faulty Information	Faultless Information
PC1 Da-	23	77

taset		

3.1 Data Pre-processing

This paper refers PCI dataset as input. The dataset is the matrix ($m \times n$) with distinguish features.

$X_{ij} = A \times B$

Xij is the initial PC1 data matrix.
A is the size of dataset.
B is the number of distinguish features.

$$s = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2} \quad (1)$$

Where $\bar{x}$ is $\frac{1}{n}\sum_{i=1}^{n} x_i$

$\boldsymbol{n}$ is the number of elements in sample

$\boldsymbol{x_i}$ is the random value of B taken from a finite data set $x_1, x_2, \dots, x_n$.

$\boldsymbol{s}$ is the standard deviation.

$$t = \frac{x_1 + x_2 + \dots + x_n}{n} \quad (2)$$

Where t is the mean value of distinguish features

B = $x_1, x_2, \dots, x_n$

From eq. (1) and eq. (2) the new preprocessed dataset will be (matrix representation).:

$$\varphi_{ij} = s \times t \quad (3)$$

Where s is the standard deviation and
t is the mean of distinguish features.

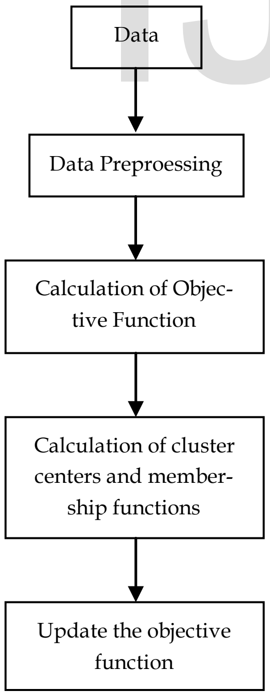

Data
Data Preprocessing
Calculation of Objective Function
Calculation of cluster centers and membership functions
Update the objective function

Fig. 1. Flow Diagram of FCM

3.2 Fuzzy C-Means Clustering

The Fuzzy C-Means Clustering (FCM) also called ISODATA is the clustering algorithm executed for determination of each point's degree that belongs to cluster. Generally, the restrictive constraint scale the performance of various hard clustering algorithms to cluster original data into different cluster groups. The Bezdek [32] proposed the Fuzzy C-Means Clustering to improve the Hard C-Means Clustering (HCM) in which the overlapping between different groups takes place.

The reference Fuzzy C-Means clustering considered is studied in the work of Zhiwei Gao et al. [31]. In this method the output vector of preprocessing (eq. 3) taken as input ($\varphi_i \, for \, (i = 1,2,\dots,n)$) are classified in c number of clusters and each clustering center is calculated. The primary difference that outdates the HCM against FCM is fuzzy partition. This partition is responsible for determination of the degree by which every data point belongs to every group using the membership function in the range 0-1. 'U' the element of membership matrix is allowed to vary in range 0 to 1, and also derives the Fuzzy partition. Based on the rules of normalization the total of membership elements in a data set is equal to 1 [31].

$$\sum_{i=1}^{c} u_{ij} = 1, \forall j = 1, \dots, n \quad (4)$$

Where, c is number of clusters.
The objective function (or cost function) of FCM is the generalization of equation:

$$J(U, c_1, \dots, c_c) = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c}\sum_{j}^{n} u_{ij}^{m} d_{ij}^{2} \quad (5)$$

where, u_{ij} is between 0 and 1, c_i is the clustering centers of fuzzy group i, $d_{ij} = \|c_i - \varphi_j\|$ is the Euclidean distance between the ith clustering centers and the jth data point, x_j is the jth data point, $m \in [1, \infty]$ is weighted index.

To obtain the required parameters new objective functions are structured that makes equation (5) into minimum

$$\bar{J}(U, c_1, \dots, c_c, \lambda_1, \dots, \lambda_n) = J(U, c_1, \dots, c_c) + \sum_{j=1}^{n}\lambda_j\left(\sum_{i=1}^{c} u_{ij} - 1\right) \quad (6)$$

$$\sum_{i=1}^{c}\sum_{j}^{n} u_{ij}^{m} d_{ij}^{2} + \sum_{j=1}^{n}\lambda_j\left(\sum_{i=1}^{c} u_{ij} - 1\right)$$

Where $\lambda_j, j = 1,2,\dots,n$ is the Lagrange multiplier of n inhibited formula described in equation (4). The required values that minimize the equation (5) are as follows:

$$c_i = \frac{\left(\sum_{j=1}^{n} \mu_{ij}{}^{m} x_j\right)}{\sum_{j=1}^{n} \mu_{ij}{}^{m}} \quad (7)$$

$$\mu_{ij} = 1 / \sum_{k=1}^{c} (d_{ij}/d_{ik})^{(2/(m-1))} \quad (8)$$

Fuzzy C-Means becomes the simple iterative process by virtue of the equations (7) and (8). The steps below [31] are structured to calculate the center c_i and membership matrix U

Step1: Select a number anonymously in the range 0 and 1 for calculation of membership matrix U and satisfy the number in eq (4).

Step2: Calculating the value of clustering centers c_i from eq (4).

Step 3: In accordance with equation (5), calculate the cost function. The iteration in algorithm will be terminated once it is less than certain value of threshold in comparison with change in last cost function value.

Step 4: Calculate the next matrix U (based on eq. 8) and return to step 2.

Finally it can be concluded that weighted index *m* and cluster c are required in FCM algorithm. *m* decides the flexibility of algorithm i.e. if it high in value the cluster effect will be deprived and if in case if it is too small, the algorithm will be near to Hard C-Means Clustering (HCM). If the condition c>1 holds true, the c will be far less than total number of cluster samples.

4 Adaptive Neuro Fuzzy Inference

The adaptive neuro fuzzy inference system is reasoning fuzzy system that is trained by neural network for computation of membership function parameters. The method tracks the input output data as a non-linear relation with inputs x,y and f as output.

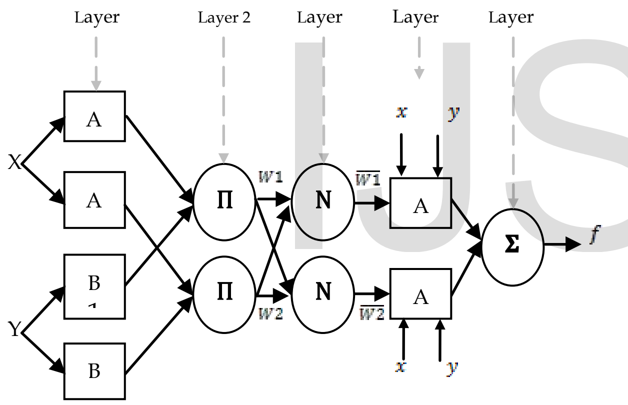


Figure 2: Adaptive Neuro Fuzzy Inference System Architecture [33]

By the training of system to a number of epochs the knowledge base is developed. The fuzzy inference system is defined in following steps for minimization of error rate.
Layer 1: In this layer the degree of membership is upgraded in reference to the parameters of fuzzy sets [34].

$$i = \mu_{A_i}(x), i = 1,2 \qquad (9)$$

$$i = \mu_{B_i}(y), i = 1,2 \qquad (10)$$

Layer 2: In this layer the fuzzy value of inputs is calculated. In the range [0, 1] the membership value of fuzzy set is determined [34].

$$w_i = \mu_{A_i}(x)\mu_{B_i}(y), i = 1,2 \qquad (11)$$

Layer 3: this layer normalizes the firing strength [34].

$$\overline{w}_i = \frac{w_i}{w_1+w_2}, i = 1,2 \qquad (12)$$

Layer 4: the input of this layer is the fuzzy output of layer 2 while the output is a single number. The parameters are defined by eq. 13 [35].

$$\overline{w}_i f_i = \overline{w}_i(p_i x + q_i y + r_i) \qquad (13)$$

5. Layer 5: the overall output is computed by summation of each incoming signal [34].

$$\sum_i \overline{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \qquad (14)$$

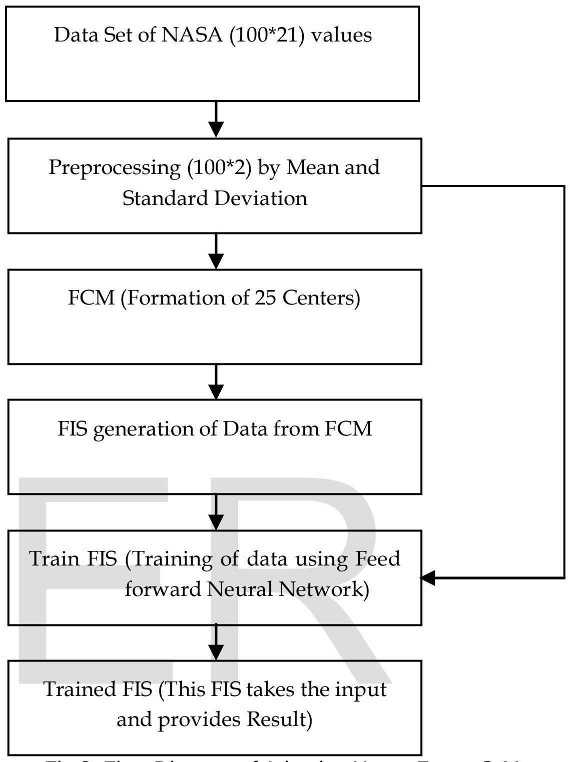

Data Set of NASA (100*21) values
Preprocessing (100*2) by Mean and Standard Deviation
FCM (Formation of 25 Centers)
FIS generation of Data from FCM
Train FIS (Training of data using Feed forward Neural Network)
Trained FIS (This FIS takes the input and provides Result)

Fig 3. Flow Diagram of Adaptive Neuro Fuzzy C-Means Clustering

5 Experimental Setup

Software fault detection by Fuzzy C-Means Clustering:
The dimension of data in matrix form is (100x21) indicating the 100 cases of faulty/non-faulty modules with 21 properties of each. The methodology pre-process the data (calculation of standard deviation and mean) is done for each software stream. This reduces the data matrix to (100x21) i.e. 100 cases with 2 properties of each.
In original dataset Class Distribution: the class value (defects) is discrete

% false: 77 = 6.94%

% true: 1032 = 93.05%

The pre-processed data is fed to Fuzzy C-Means clustering to obtain 25 clusters. The first cluster of data is labeled as non-faulty while remaining 24 fields indicates faulty data. Fuzzy C-Means finds the location of cluster center and assigns each stream to a cluster centre.For testing the standard deviation and mean of stream is calculated, assigned to cluster from which it shows minimum distance. The stream is classified as faulty if the cluster is faulty else vice-versa.

Software Fault Detection Using Adaptive Neuro-Fuzzy C-Means Clustering:
The dimension of data in matrix is again (100x21) which indicated 100 cases with 21 properties each. The pre-processing of data reduces the matrix to the order (100x2) i.e. 100 cases with 2 properties of each.On basis of above a fuzzy inference system is generated using FCM. Both systems till this point share same level of accuracy. This inference system is fed to feed forward neural network with training data. In training stage neural network modifies the structure of fuzzy inference system to obtain high level of accuracy.

6 Results and Coclusion

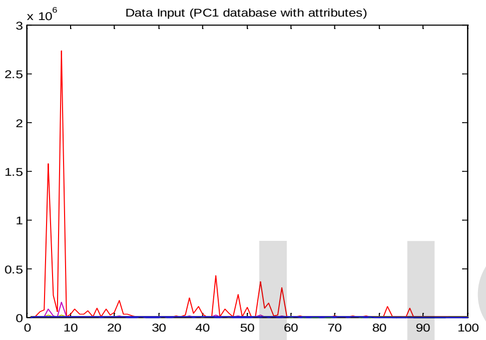


Fig. 4. Graphical representation of PC1 data distribution of faulty and non-faulty modules

In training dataset taken as input for Class Distribution: the class value (defects) is discrete in nature.

% data with positive attribute: 23 = 23%

% data with negative attribute: 77 = 77%

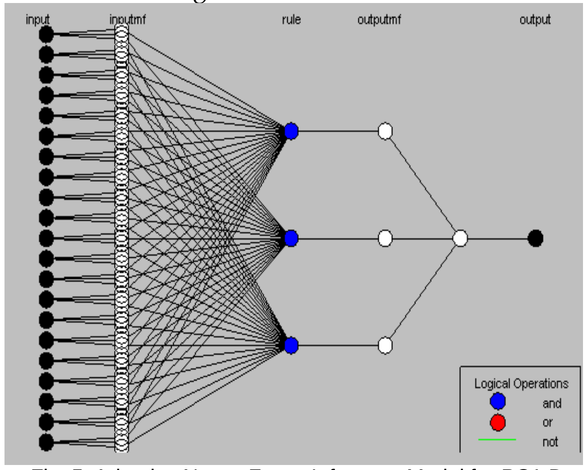
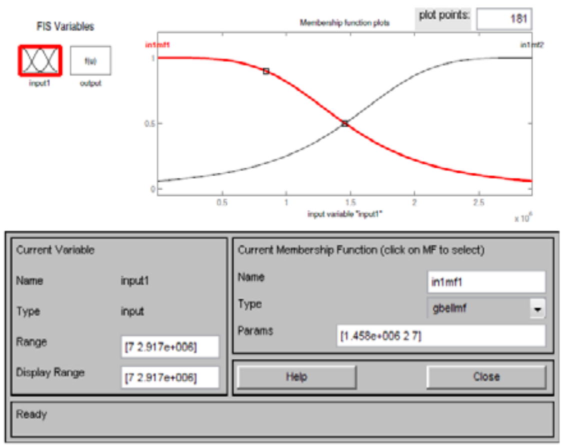


Fig. 5. Adaptive Neuro-Fuzzy Inference Model for PC1 Dataset with 3 Rules.



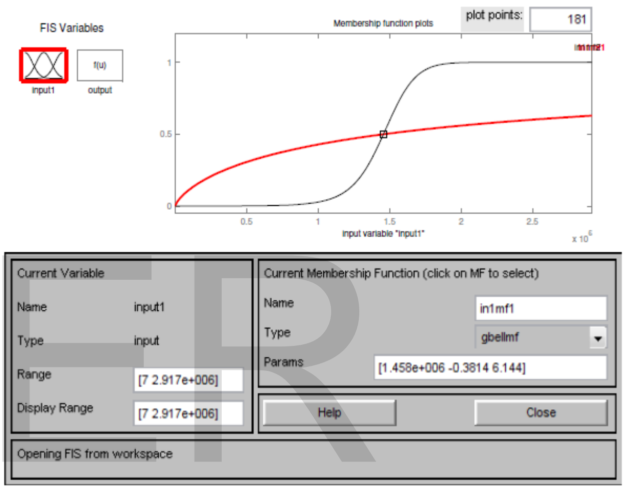
Current Variable
Name input1
Type input
Range [7 2.917e+006]
Display Range [7 2.917e+006]

Current Membership Function (click on MF to select)
Name in1mf1
Type gbellmf
Params [1.458e+006 2 7]

Help Close

Ready

Fig.6. Membership Function Plot of Fuzzy Inference System



Current Variable
Name input1
Type input
Range [7 2.917e+006]
Display Range [7 2.917e+006]

Current Membership Function (click on MF to select)
Name in1mf1
Type gbellmf
Params [1.458e+006 -0.3814 6.144]

Help Close

Opening FIS from workspace

Fig.7. Membership Function Plot of Adaptive Neuro Fuzzy Inference System

Table 2
Comparative Table Based on Simulation Factors for Fuzzy C-Means and Adaptive Neuro Fuzzy C-Means Clustering

	Fuzzy C-Means	Adaptive Neuro Fuzzy C-Means
Accuracy	77%	91%
Reliability	72.98%	73.98%
RMSE	0.068	0.09
MAE	0.23	0.13

This paper empirically evaluates and compares the performance of Fuzzy C-Means clustering technique and adaptive Neuro-Fuzzy C-Means clustering for software fault prediction. The platform for the testing is MATLAB 2010A on the PC1 testing database. The proposed Adaptive Neuro-Fuzzy C-Means Clustering based prediction technique shows 91% accuracy in results. We observe that the implementation that has achieved a better performance.C-Means clustering for the prediction of faulty prone classes. It can be further accomplished that the Adaptive Neuro-Fuzzy C-Means clustering is satisfactory in finding the faulty/fault-free area.

REFERENCES

[1] M. R. Lyu, Handbook of software Reliability Engineering IEEE Computer Society Press, McGraw Hill, 1996.

[2] F. G. Sayward A. J. Perlis and M. Shaw, Software Metrics: An Analysis and Evaluation, MIT Press, Cambridge, MA, 1981.

[3] V. Y. Shen, T.Yu, S. M. Thebaut, and L. R. Paulsen, "Identifying errorprone software—an empirical study," IEEE Trans. on Software Engineering, vol. SE-11, pp. 317–323, April 1985.

[4] S. G. Crawford, A. A. McIntosh, and D. Pregibon, "An analysis of static metrics and faults in C software," J. Syst. Sofyware, vol. 5, pp. 27–48, 1985.

[5] Bose, S.K., Presenting a Novel Neural Network Architecture for Membrane Protein Prediction, Intelligent Engineering Systems, 2006. INES '06. Proceedings. International Conference

[6] Attarzadeh, I., Proposing an Enhanced Artificial Neural Network Prediction Model to Improve the Accuracy in Software Effort Estimation, Computational Intelligence, Communication Systems and Networks (CICSyN), 2012 Fourth International Conference

[7] Yuan Chen,Research on software defect prediction based on data mining, Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on (Volume:1)

[8] N. E. Fenton and M. Neil. A critique of software defect prediction models. IEEE, Transactions on Software Engineering, 25(5):675–689, 1999.

[9] N. E. Fenton and N. Ohlsson. Quantitative analysis of faults and failures in a complex software system. IEEE Transactions on Software Engineering, 26(8):797–814, 2000

[10] Khoshgoftaar TM, Allen EB, Ross FD, Munikoti R, Goel N, Nandi A (1997) Predicting fault-prone modules with case-based reasoning. The Eighth International Symposium on Software Engineering (ISSRE '07). IEEE Computer Society, pp 27–35

[11] Thwin, M. M.; Quah, T. Application of Neural Networks for Software Quality Prediction using Object-oriented Metrics. // ICSM 2003. / Amsterdam, The Netherlands, 2003, pp. 113-122.

[12] Quah, T. S. Estimating Software Readiness using Predictive Models. // Information Sciences, 179, 4(2009), pp. 430-445.

[13] Kanmani S,Uthariaraj V. R. Sankaranarayanan V, Thambidurai, P. Object-oriented Software Fault Prediction using Neural Networks.Information and Software Technology, 49, 5(2007), pp. 483-492.

[14] Evett, M.; Khoshgoftaar, T.; Chien, P.; Allen, E. GP-based Software Quality Prediction. // Proceedings of the 3rd Annual Genetic Programming Conference / San Francisco, CA, 1998, pp. 60-65.

[15] Guo, L.; Cukic, B.; Singh, H. Predicting Fault Prone Modules by the Dempster-Shafer Belief Networks. // Proceedings of the 18th IEEE Int'l Conf. on Automated Software Eng. / Montreal, Canada, 2003, pp. 249-252.

[16] Menzies, T.; Greenwald, J.; Frank, A. Data Mining Static Code Attributes to Learn Defect Predictors. // IEEE Transactions on Software Engineering, 33, 1(2007), pp. 2-13.

[17] Zhang, M. L.; Peña, J. M.; Robles, V. Feature Selection for Multi-label Naive Bayes Classification. // Information Sciences, 179, 19(2009), pp. 3218-3229.

[18] Khoshgoftaar, T. M.; Seliya, N. Software Quality Classification Modeling using the SPRINT Decision Tree Algorithm. // Proc.of the 4th IEEE Int'l Conf. on Tools with AI. / Washington, DC, 2002, pp. 365-374.

[19] Catal, C.; Diri, B. Investigating the Effect of Dataset Size, Metrics Sets, and Feature Selection Techniques on Software Fault Prediction Problem. // Information Sciences, 179, 3(2009), pp. 1040-1058.

[20] Elish, K. O.; Elish, M. O. Predicting Defect-Prone Software Modules using Support Vector Machines. // Journal of Systems and Software, 81, 5(2008), pp. 649-660.

[21] Gondra, I. Applying Machine Learning to Software Faultproneness Prediction. // Journal of Systems and Software, 81, 2(2008), pp. 186-195.

[22] El Emam, K.; Benlarbi, S.; Goel, N.; Rai, S. Comparing Case-based Reasoning Classifiers for Predicting High Risk Software Components. // Journal of Systems and Software, 55, 3(2001), pp. 301-320.

[23] Vandecruys, O.; Martens, D.; Baesens, B.; Mues, C.; De Backer, M.; Haesen, R. Mining Software Repositories for Comprehensible Software Fault Prediction Models. // Journal of Systems and Software, 81, 5(2008), pp. 823-839.

[24] Yuan, X.; Khoshgoftaar, T. M.; Allen, E. B.; Ganesan, K. An Application of Fuzzy Clustering to Software Quality Prediction. // Proc. of the 3rd IEEE Symp. on Application-Specific Systems and Software Eng. Technology, / Washington, DC, 2000, pp. 85-90.

[25] Basili VR, Briand LC, Melo WL (1996) A validation of object-oriented design metrics as quality indicators. IEEE Trans Softw Eng 22(10):751–761. doi:10.1109/32.544352

[26] Yue Jiang, Bojan Cukic, Tim Menzies, "Fault Prediction using Early Lifecycle Data". Lane Department of Computer Science and Electrical Engineering,West Virginia University, Morgantown, WV 26506, USA

[27] Witten, I.H. and Frank, E., Data Mining: Practical Machine Learning Tools and Techniques (San Francisco, CA: Morgan Kaufmann), 2nd edition (2005)

[28] Gaines, B.R., Compton, P.: Induction of Ripple-Down Rules Applied to Modeling Large Databases. J. Intell. Inf. Syst. 5(3), 211–228 (1995)

[29] Hassan Najadat and Izzat Alsmadi, Enhance Rule Based Detection for Software Fault Prone Modules. International Journal of Software Engineering and Its Applications Vol. 6, No. 1, January, 2012

[30] Zhiwei Guo, Chengqing Yuan, Peng Liu, "Study on Identification Model of Cylinder Liner-Piston Ring Using Vibration Analysis Based on Fuzzy C-means Clustering" The Open Mechanical Engineering Journal, 2012, 6, (Suppl 2: M2) 126-132 1874-155X

[31] James C. Bezdek, Robert Ehrlich, William Full, "FCM: The Fuzzy C-Means Clustering Algorithm" Computers & Geosciences Vol. 10, No. 2-3, Pp. 191-203, 1984.

[32] Pejman Tahmasebi, Ardeshir Hezarkhani, Application of Adaptive Neuro-Fuzzy Inference System for Grade Estimation; Case Study, Sarcheshmeh Porphyry Copper Deposit, Kerman, Iran" Australian Journal of Basic and Applied Sciences, 4(3): 408-420, 2010

[33] Ahmed A. M. Emam, Eisa Bashier M. Tayeb, A. Taifour Al, Ammar Hassan Habiballh, "ADAPTIVE NEURO-FUZZY INFERENCE SYSTEM IDENTIFICATION OF AN INDUCTION MOTOR" European Journal of Science and Engineering Vol. 1, Issue 1, 2013.

[34] M. H. Olyaee, H. Abasi, M. Yaghoobi, "Using Hierarchical Adaptive Neuro Fuzzy Systems And Design Two New Edge Detectors In Noisy Images" Volume 2013, Year 2013 Article ID jsca-00030, 10 Pages doi:10.5899/2013/jsca-00030.