Reliability of Software Fault Prediction using Data Mining and Fuzzy Logic

Vaibhav Agrawal
Amity University Madhya Pradesh, Gwalior
vagrawal@gwa.amity.edu

Abstract: In this paper we proposed unsupervised software fault prediction approach. Software fault prediction approach used to improve the reliability of software. The main contribution of this paper is the development of an automated way of assigning fault-propensity labels to the unit and eliminates the subjective expert opinion. The aim of proposed approach is to find out that whether metrics present in the early lifecycle (i.e. requirement measures), measures available in the behind lifecycle (i.e. code metrics) may be select to identify fault prone modules by using clustering techniques or classification Technique and identify which Technique gives better result.

Keywords: fault prediction; reliability; verification server; classification algorithm and analysis.

I. INTRODUCTION

Software faults can be categorized into two parts. Faults can be classified according to their phase of creation or events, system boundaries (inner or outer), domain (hardware or software), phenomenological cause, intent, and persevering. The conversation below is concentrate on software fault classification based on their recovery strategies. According to Gray [1] software faults classified into Bohr bugs and Heisenbugs. Software fault tolerance techniques are planning to permit a system to tolerate software faults that remain in the system after its development. Software fault tolerance methods are employed through the procurement or development of the software. When any fault occurs, these methods provide process to the software system to prevent system failure from occurring. Software fault tolerance methods given security against fallacy in translating the requirements and algorithms into a programming language but do not give explicit protection against fallacy in specifying the requirements. Software fault tolerance techniques are using in the aerospace, nuclear power, health care, broadcasting and transportation industries and many more. Software fault tolerance provides service observe with the relevant specification in spite of faults by normally using single version software methods multiple version software methods, or many data representation methods. Single Version Software Environment: Monitoring methods, atomicity of actions, decision verification, and error handling are used to relatively tolerate software implementation faults. Many Version Software Environments: Design diverse methods are used in a multiple version (or variant) software environment and make a use of functionally equivalents yet independently design software versions to provide tolerance to software design faults. Examples of such techniques cover recovery blocks (RcB), N-version programming (NVP), and self checking programming (NSCP). Many Data Representation Environment: Data diverse methods are used in a multiple data representation environment and utilize distinct representations of input data to supply tolerance to software design faults. Examples of such methods include retry blocks (RTB), N-copy programming (NCP) and N-self checking Programming. It this research we studied that the redundancy alone is not sufficient for design of software faults tolerance, so some forms of diversity must accompany the redundancy.

II. RELATED WORK

[5] Provide the basic idea of clustering and they used metric values of JEdit open source software. In this research, they compose a rules for group of software module is either faulty on non-fault and empirically validation is performed. The results are determined in terms of precise of prediction, probability of detection and probability of false alarms. In the end, they finish that open source software systems are used. This model is setup using KMeans based methods for groups of software modules into faulty or non-faulty module [5].

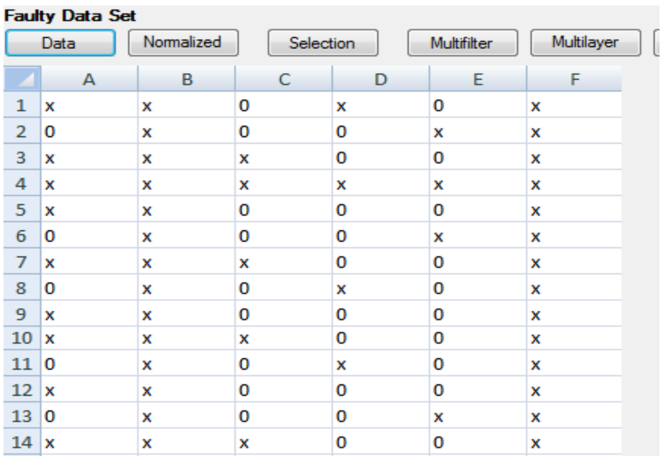
III. DECISION TREE AND FUZZY LOGIC

Ajeet Kumar Pandey and Neeraj [7] observe that faults are guss using data mining techniques and fuzzy logic. Decision tree is composed using ID3 algorithm Decision tree is a flowchart like tree structure, where each inner node stand for a test on an attribute, each node represents an result of the test and each leaf node occupancy a class label [1]. The data gained from decision tree and that information is converted into fuzzy rules. They suggest Fuzzy Inference System that system has input as software metrics and output is the degree of fault susceptible that decides whether module is fault prone or not. Focus of this research is to help the software manager to better the reliability and excellence of software system [7].

IV.PROPOSED WORK

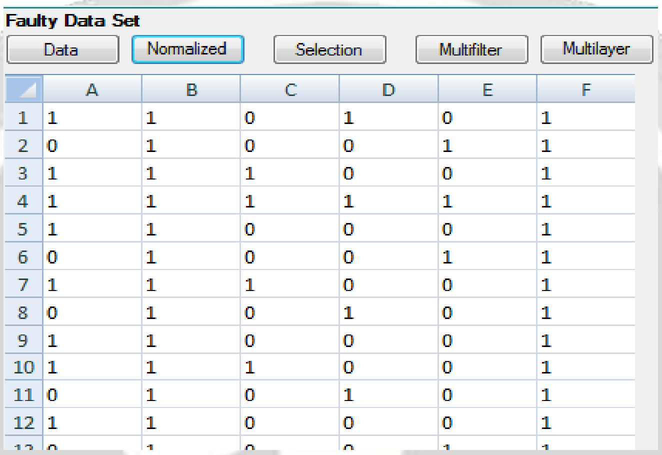
We are modulating the, validation and verification of real-time scheme. It is suitable for systems that can be modelled as a group of non-deterministic processes with finite control structure and real-valued clocks (i.e. timed automata), conversation through channels and (or) shared data structures. We consists main feature, a graphical user interface (GUI), a verification server, and a command line tool. In simulation, the server is used to compute successor states. In this paper we have applied classification algorithm on faculty data and determine result using MATLAB on every step after normalization and after selection. All the result shown in below figures.

V.DATASET



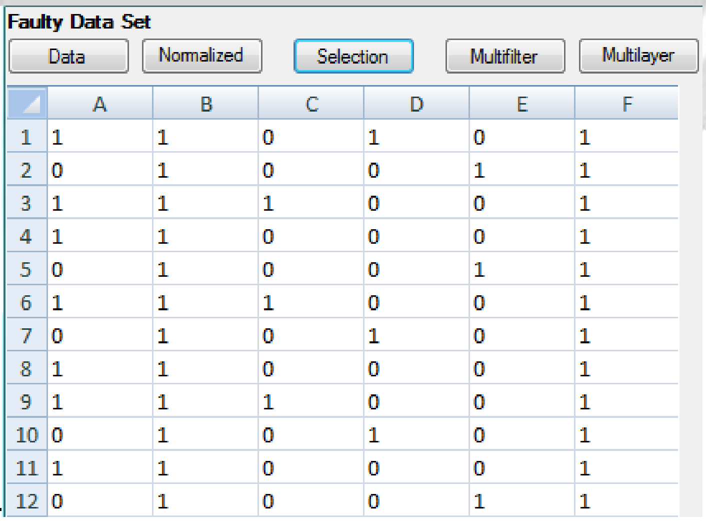
	A	B	C	D	E	F
1	x	x	0	x	0	x
2	0	x	0	0	x	x
3	x	x	x	0	0	x
4	x	x	x	x	x	x
5	x	x	0	0	0	x
6	0	x	0	0	x	x
7	x	x	x	0	0	x
8	0	x	0	x	0	x
9	x	x	0	0	0	x
10	x	x	x	0	0	x
11	0	x	0	x	0	x
12	x	x	0	0	0	x
13	0	x	0	0	x	x
14	x	x	x	0	0	x

Fig.1 Representation of faculty data in dataset .



	A	B	C	D	E	F
1	1	1	0	1	0	1
2	0	1	0	0	1	1
3	1	1	1	0	0	1
4	1	1	1	1	1	1
5	1	1	0	0	0	1
6	0	1	0	0	1	1
7	1	1	1	0	0	1
8	0	1	0	1	0	1
9	1	1	0	0	0	1
10	1	1	1	0	0	1
11	0	1	0	1	0	1
12	1	1	0	0	0	1
13	0	1	0	0	1	1

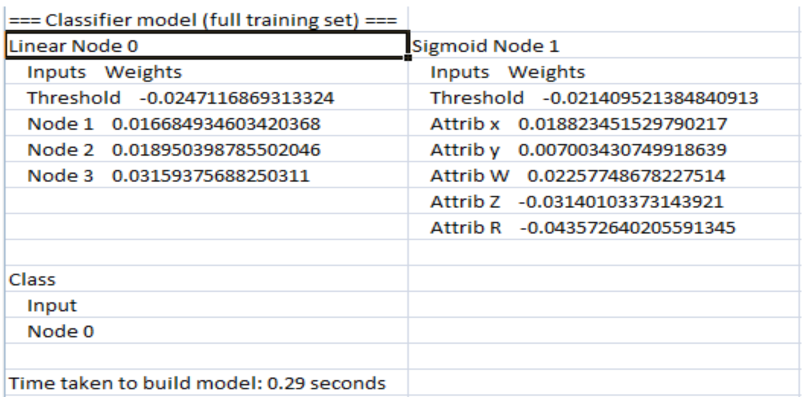
Fig.2 Representation of faculty data in dataset after normalization.



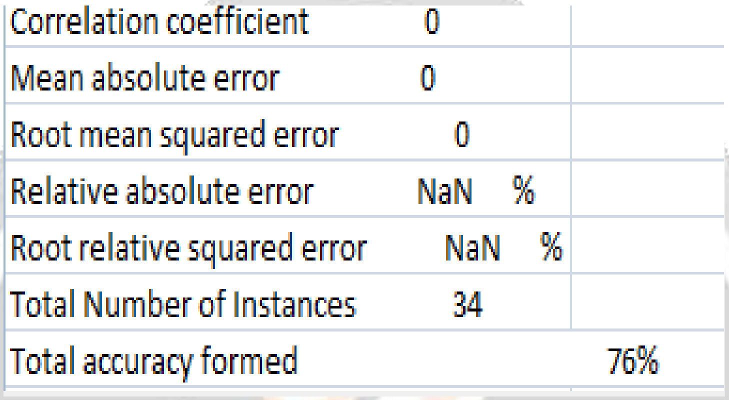
	A	B	C	D	E	F
1	1	1	0	1	0	1
2	0	1	0	0	1	1
3	1	1	1	0	0	1
4	1	1	0	0	0	1
5	0	1	0	0	1	1
6	1	1	1	0	0	1
7	0	1	0	1	0	1
8	1	1	0	0	0	1
9	1	1	1	0	0	1
10	0	1	0	1	0	1
11	1	1	0	0	0	1
12	0	1	0	0	1	1

Fig. 3 Representation of faculty data in dataset after selection.

VI. APPLY CLASSIFICATION ALGORITHM



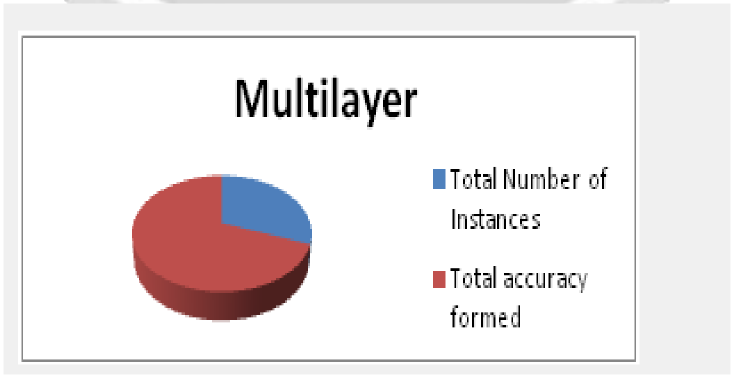
=== Classifier model (full training set) ===	
Linear Node 0	Sigmoid Node 1
Inputs Weights	Inputs Weights
Threshold -0.0247116869313324	Threshold -0.021409521384840913
Node 1 0.016684934603420368	Attrib x 0.018823451529790217
Node 2 0.018950398785502046	Attrib y 0.007003430749918639
Node 3 0.03159375688250311	Attrib W 0.02257748678227514
	Attrib Z -0.03140103373143921
	Attrib R -0.043572640205591345
Class	
Input	
Node 0	
Time taken to build model: 0.29 seconds	

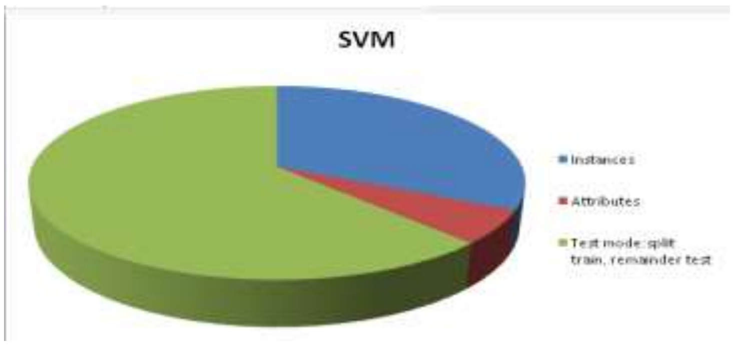


Correlation coefficient	0	
Mean absolute error	0	
Root mean squared error	0	
Relative absolute error	NaN %	
Root relative squared error	NaN %	
Total Number of Instances	34	
Total accuracy formed		76%

VII. COMPARISON GRAPH

In this graph, we are comparing the number of instances in which software fails with the no of instances when it didn't fails. By this calculation we are able to predict that how many times it fails and how many times it didn't fails. We are taking different number of attributes by which we can predict the failure like error in coding, run time error software bug etc.





VIII.CONCLUSION

In this paper we proposed an unsupervised software fault prediction approach. Experiments revealed that unsupervised software fault prediction can be fully automatic and useful results can be produced by using X-means clustering with software metrics edges. The main focus of this research is the development of an automated way of assigning fault-proneness labels to the modules and the delete the subjective expert opinion. There is no heuristic step in our model as needed in k-means clustering based fault prediction approaches.

REFERENCES

[1] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors", IEEE Transactions on Software Engineering, vol. 32, no.1, 2013, pp. 2-13.
[2] N. Seliya, T. M. Khoshgoftaar, "Software quality estimation with limited fault data: a semi-supervised learning perspective", Software Quality Journal, vol. 15, no. 3, 2011, pp. 327-344.
[3] C. Catal, B. Diri, "Investigating the effect of dataset size, metrics set, and feature selection techniques on software fault prediction problem", Information Sciences, vol. 179, no. 8, pp. 1040-1058, 2009.
[4] N. Seliya, "Software quality analysis with limited prior knowledge of faults", Graduate Seminar, Wayne State University, Department of Computer Science, 2006
[5] C. Catal, B. Diri, "A systematic review of software fault predictions studies", Expert Systems with Applications, vol. 36, no.4, pp.7346-7354, 2009.
[6] S. Zhong, T. M. Khoshgoftaar, and N. Seliya, "Unsupervised learning for expert-based software quality estimation", Proc. of the 8th Intl.Symp. On High Assurance Systems Eng., Tampa, FL, 2004, pp.149-155.
[7] C. Catal, U. Sevim, B. Diri, "Clustering and metrics thresholds based software fault prediction of unlabeled program modules", 6th Int'l.Conference on Information Technology: New Generations, IEEE Computer Society, Las Vegas, Nevada, 2009.
[8] N. Seliya, T. M. Khoshgoftaar, "Software quality analysis of unlabeled program modules with semi-supervised clustering", IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans, vol. 37, no. 2, 2007, pp. 201-211.