International Journal of Advanced Computational Engineering and Networking, ISSN(p): 2320-2106, ISSN(e): 2321-2063
Volume-10, Issue-2, Feb.-2022, http://iraj.in

SOFT COMPUTING TECHNIQUES FOR SOFTWARE FAULT PREDICTION: COMPARATIVE ANALYSIS

[1]GURMEET KAUR, [2]JYOTI PRUTHI, [3]PARUL GANDHI

[1]Research Scholar CST, MRU Faridabad, India
[2]Prof. CST, MRU Faridabad, India
[3]Prof. FCA, MRIIRS Faridabad, India
E-mail: [1]grmtkaur02@gmail.com, [2]jyoti@mru.edu.in

Abstract - This research paper compares the performance of software fault prediction model developed using Bayesian net, FIS and ANFIS with specific focus on process level software metrics during software development. The developed models used for comparison are designed using software metrics as input data to compute the fault density for each basic phase of SDLC. The comparative study is built on validation of the prediction accuracy for various soft-computing modeling techniques. Furthermore, the comparative study is performed on the data set from the PROMISE repository, and the outcomes of various soft computing methodology based models has been compared. The study shows that ANFIS based model for software fault prediction provides better accuracy as calculated from statistical analysis, MMRE (0.016461) and RMSE (0.0375) in contrast to other soft computing techniques based on a model on Bayesian net and FIS.

Keywords - Software development life cycle (SDLC), Software Fault Prediction (SFP), Soft Computing, Fuzzy Inference System (FIS), Neuro-Fuzzy System (NFS), Adaptive Neuro-Fuzzy Inference System (ANFIS), Software Fault, Artifical Neural Network (ANN).

I. INTRODUCTION

Faults in software systems are a serious problem. The Software Quality Assurance and Software Reliability is the core to guarantee the superior quality of software. Both these concepts are attracted all throughout the development of the software and measure. A software bug is a defect, error, failure, or flaw during the execution of a software program that permits it from acting as expected function (e.g., delivering an erroneous outcome). The software defect may be a defect that gives rise to failure of software functionally. The software failure alludes to the uncertain outcomes made because of various state and surroundings factors which make defaults during the execution of a utility program. A software fault prediction offers the benefits regarding time complexity, low cost budget, testing effort, and increases the reliability along with the quality of the software if it is applied at the starting phase of traditional and agile based software development life cycle.

Defective software modules cause software failures, reduce customer satisfaction, increased development, and maintenance costs. The objective of the development of software fault prediction models is making use of course of actions which will be acquired generally right on time for the life cycle of project development to give adequate initial assessments of the standard of developing a software project.

With the fast progression of the software business, the use of the agile development technique proposed in recent years stress on timely reacting to the changes in prerequisites as describing the insufficiency of the conventional programming development process. As the requirements of programming change frequently, the estimations should have the opportunity to be firmly checked. Changing requirements are one among the principle issues that emerge within the development process of software. Agile-based software process effectively deals with the truth of variation. Many of the existing faults prediction algorithms focus on expecting the number of faults in the modules of software using the product metrics. As there are many well-established programming measurements available for the TSD like Cyclomatic Complexity Metric (CCM), Halstead Complexity Metric (HCM), Lines of Codes (LOC), and defect density estimates the defect per function point or defect per KLOC, but a few are regularly implemented to the ASD process. Agile based process incorporates changes all through the technique because of the iterative and gradual development process. There are a couple of difficulties in estimating the presentation of the agile-based development group.

There are various kinds of soft computing techniques like Bayesian net, Fuzzy Logic System, machine learning, Neuro Fuzzy System, Artifical Neural Network, and Adaptive Neuro Fuzzy Inference System employed for Software Fault Prediction. In recent years, various researchers try to automate the fault prediction process by designing computer-based models that can perform learning from existing prediction data. The central focus of this research manuscript is the implementation of soft computing techniques and approaches like Fuzzy Logic System, and Adaptive Neuro Fuzzy Inference System on which the model is design by selecting software metrics as inputs for a dataset and comparing in what way the performance (in term of MMRE, BMMRE) of the different models affects the context. The aim of this research paper is the comparison of the

performance of different soft computing modeling like (Bayesain net, Fuzzy Logic System, and Adaptive Neuro Fuzzy Inference System) for SFP framework.

II. RELATED STUDY

As the growing number of faults affects development time, cost and quality of a software package, so software fault prediction is the way of detecting faulty components in modules before the implementation of the software. This comparative study is predicated to provide a comprehensive picture in the area of software fault prediction administered by many researchers using various soft computing techniques. The study includes comparative analysis of soft computing techniques based models developed by Fenton [10], Panday and Goyal [11], Yadav [12], and Sharma [13] for software fault prediction.

Chatterjee, S., Maji, B (2018) [1] planned a model to arrive at a object assessment of weaknesses during initial development of programming lifecycle and to anticipate the total number of faults. The author applied Interval type-2 fuzzy reasoning for getting the unexpected probability esteems inside the node probability tables of the assumption network.

The proposed Bayesian construction makes use of the programming faculty to understand the predefined information about programming estimations at the initial stage for achieving focused on various deficiencies. The benefit of the suggested model are that it can oversee the product engineers for accomplishing a focused on condition of the whole number of programming defects and anticipate the complete number of faults.

Bilgaiyan, S., Mishra, S. and Das, M (2019) [2] performed near-investigation of anticipated for different kinds of ANNs and spotlighted on two kinds of neural network as ANN based on feed forward back-propagation network and Elman neural network. The proposed work uses three particular execution estimations to measure the performance of the model for example, mean square error (MSE), mean size of relative error (MMRE), and assumption (PRED (X)) to appear at. The feed forward back-propagation neural network has high algorithm speed, fixed algorithm time and adaptation to non-critical failure concerning Elman organization.

Kapil Juneja (2019) [3] proposed a system using inter version and inter project assessment to recognize the product defect. During the working of the framework, the past assignments or adventure variations are taken for preparing sets, and thus, the present version or exercises are taken as testing sets. The author performed investigation taken from PROMISE store, even as on PDE and JDT adventures and other more nine open source adventures. The cooperation on these two parameters is applied by making use of the symbolic or fuzzy logic to recognize the successful task measurements. The assessment results showed that the suggested system showed outstandingly good outcomes for Eclipse-PDE and Eclipse-JDT based projects.

The SFP model affects in various areas of software development which in turn offer benefits in terms of quality, reliability, time and cost of completion.

III. SOFT-COMPUTING TECHNIQUES

A. Byesian Net

The Bayesian net is useful for handling missing data, for uncertain input/output variables. The Bayesian net is composed of mainly two components, and these are a directed acyclic graph and a probability distribution. The Bayesian net works on the basis of a directed acyclic graph where nodes are variables and links between nodes show the relationships between variables. It provides an easier way to understand the relationship between variables using a graph. The values of output variables computed using probability distribution conditionally or unconditionally for the node/variables.

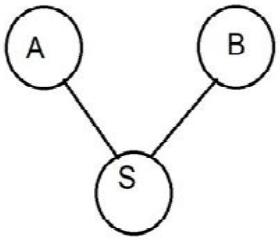
Bayesian net relates to the field of probabilistic graphical model and is also popular as a belief networks. A Bayesian network over Z= {z1, z2, ... zn} set of random variables from a finite domain is a pair Y=(B, θB) that presents a joint probability distribution of Zi.

Where B is a directed acyclic graph in which nodes corresponds to a random variable z1, z2, ... zn and edge of the graph represent the relationship between the variables.

And θ_B represents parameter for each value z_i of Z.

Y represents unique joint probability distribution over Z given by:

$$\Pi_{zi} P_v(z_{1,} z_{2,} \ldots .. z_n) = \prod_{i=1}^{n} Py(Zi/\Pi_{zi}) \quad (1)$$



A	B	Prob. (S:A,B)
0	0	0
0	1	0
1	0	0
1	1	0.99

Fig. 1 Sample of Bayesian network

Consider the sample network in figure 1. Here the node A, B, and S represents the events “Fuel”, “Battery life”, and “Vehicle move” respectively. It is assumed that all the variables have binary value represented in the table as shown in the figure. Using joint probability distribution for S will require four parameters. It can be noticed from table that when A=1, B=1 then probability of S is 99.5 i.e. “Vehicle will move”.

B. Fuzzy logic

Fuzzy logic launched in 1965 by Prof. Zadeh. It is basically derived from the fuzzy set theory. It is a

methodology for solving problems which are difficult to understand quantitatively. Figure 2 presents the fuzzy methodology.

A fuzzy set is derived from a crisp set. A fuzzy set permits partial membership whereas a crisp set permits either no membership or full membership absolutely. A characteristic function is to use to describe the membership or non-membership of item z in a crisp set Y, where in a fuzzy set this concept is illustrated by using partial membership. The degree or grade of membership of a fuzzy set is classified by a membership function which belongs to every element in the fuzzy set a real value from the closed interval [0, 1].

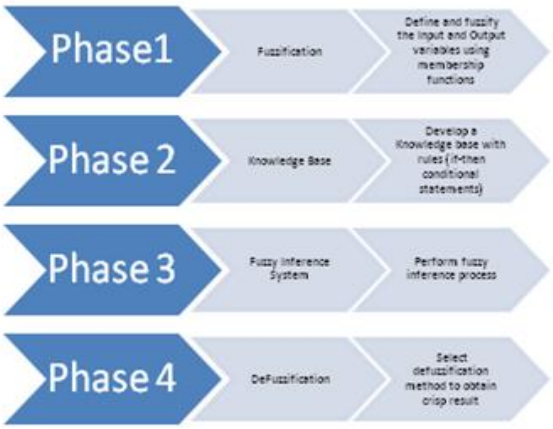


Fig. 2 Fuzzy methodology

The value of a fuzzy number is imprecise, not exact like single numbers. The domain of fuzzy number is defined, normally the set of real numbers, and the range is the real positive numbers from the closed interval [0, 1]. Fuzzy logic is designed when fuzzy sets are used with the logical expression.

1). Fuzzification

Here every linguistic variable can be a triangular Fuzzy numbers, TFN (a, m, b), a≤m, b≥ m. The membership function (μ(x)) for TFN is explained below:

$$\mu(x) = \begin{cases} 0, X \leq a \\ \frac{X-a}{m-a}, a \leq X \leq m \\ \frac{b-X}{b-m}, m \leq X, \leq b \end{cases} \quad (2)$$

To describe the fuzziness of input metrics, a membership function is used. Membership function can be developed either using real data or with the help of an expert of a particular domain. As the performance of a method turns on the membership functions applied, so the designing of the membership function is very crucial. Thus, it is required to describe how the membership functions are acquired. There are various methods like: rank ordering, intuitive, inference, neural networks, angular fuzzy sets, genetic algorithm for allocating membership values to fuzzy variables. But there are no classic rules or guidelines that can be applied for the proper membership function generated method. The intuitive method assigned membership value on the basis of the human intelligence and their level of knowledge. The various types of membership function are triangular, trapezoidal, parabolic etc. The figure 3 shows the representation of TFN (a, m, b).

2). Fuzzy Rule

There are various sources such as analyzing historical data, experts of particular domain, and knowledge base from literature to formulate the fuzzy rule base [4] [5]. Here, fuzzy rule is explained using a conditional statement as IF-THEN. IF portion of the rule is known as a predecessor and THEN portion is a successor [6].

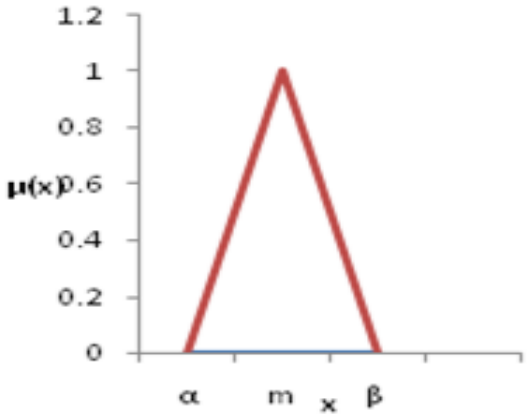


Fig. 3 Graphical presentation of triangular membership function

3). Fuzzy Reasoning and Defuzzification

Fuzzy reasoning system analyzes computes and collects the outputs of each fuzzy rule. FIS use various defuzzification methods to draw fuzzy set into a crisp number like, max-min centroid and bisection. The figure 4 shows the defuzzification method.

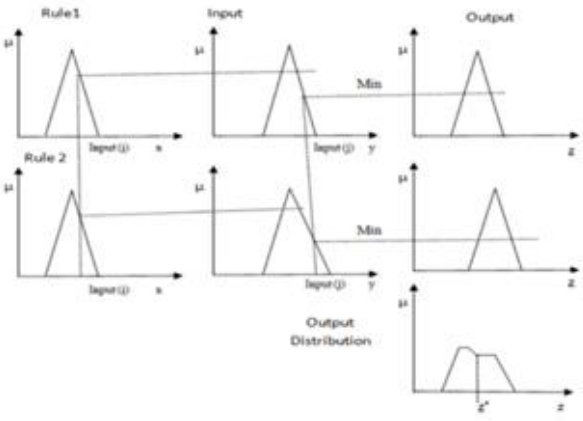


Fig. 4 Defuzzification method

The centroid method is general and physically attractive among all the defuzzification methods. It is also known as center of gravity/area [7].

C. ANFIS

Fuzzy Systems requires human expertise knowledge to design a rule base and implement fuzzy inference system to deduce the complete or final output. The earlier knowledge about the system is very important for designing of membership functions and corresponding if-then rules. But, there is no organized method to convert knowledge and experiences of humans into the knowledge base of a fuzzy inference system. Also to achieve results with the low fault rate, there is a requirement for modification of training algorithms. But, ANN training methods do not depend on human knowledge and experience. Also because of the uniform design of ANN, it is not possible to obtain organized knowledge from the arrangement of the ANN.

ANN is a Black box and uses training for scratch while FIS is Interpretable and employs heuristics and linguistic knowledge. The characteristics of artificial neural network (ANN) and fuzzy inference system (FIS) are interrelated which promote the aspect of the Neuro-Fuzzy System (NFS) that include benefits of the capability of the fuzzy inference system to preserve human knowledge & experience and the capability of training of the ANN. The Adaptive Neuro-Fuzzy Inference System (ANFIS) is architecture which provides a general way to employ a training method to a Fuzzy Inference System (FIS) to present it in a particular architecture, like artificial neural networks (ANN). The robustness and fastest convergence is the key to success of ANFIS. The ANFIS architecture [8] is illustrated in figure 5.

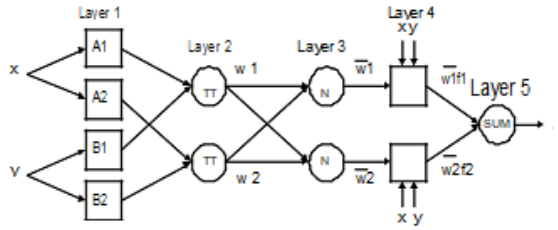


Fig. 5 ANFIS architecture with the two inputs and two rules

A general first order Sugeno fuzzy model for adaptive network is described using IF-THEN rule with two fuzzy inputs as follow:

First rule

If (x = A1) and (y = B1)

then

$f_1=p_1x+q_1y+r_1$

Second rule

If (x = A2) and (y = B2)

then

$f_2=p_2x+q_2y+r_2$

The result can be explained using z from Figure 5 as:

$$z = \frac{w1}{w1+w2} f1 + \frac{w2}{w1+w2} f2$$

$$z = \overline{w_1}\,(p_1x+q_1y+r_1) + \overline{w_2}\,(p_2x+q_2y+r_2)$$

$$z=(\overline{w_1}x)p_1+(\overline{w_1}y)q_1+(\overline{w_1})r_1+(\overline{w_2}x)p_2+(\overline{w_2}y)q_2+(\overline{w_2})r_2 \quad (3)$$

The adaptive neural network of a first order Sugeno fuzzy model is functionally equivalent to equation (3). It described that the ANFIS model presented in figure 5 be framed as a linear equation of the subsequent boundary (layer 4), when the estimates of the premise boundary (layer 1) are set.

Generating Initial FIS

The problem with fuzzy inference system is identification of rules. Rules are generated either by using grid partitioning or subtractive clustering methods in ANFIS.

The algorithms for learning/training of ANFIS are classified as two types:

Backward Pass: It works in the backward direction, using gradient descent method, and generates output errors in the backward direction.

Hybrid learning algorithm: It works in the forward direction, applies the least squares method to recognize subsequent parameters, and generates output errors in forward until the last layer.

IV. RESULT AND DISCUSSION

In order to compare the performances of the software fault model using soft computing methodology, a data sets of twenty plus real software projects are selected from PROMISE repository for case studies [9] and reconstruct in Appendix Table 1 where the selected software metrics are stated with qualitative value of as Very Low (VL), Low (L), Medium (M), High (H) and Very High (VH) and Table II represents the data set with centroid values.

This research paper described that Fenton [10], Pandey and Goyal [11], Yadav [12], and P. Sharma and A.L. Sangal [13] used dataset [9] in their software fault prediction model to predict faults using soft computing modeling techniques respectively.

Fenton [10] developed a model which based on Bayesian net to estimate the software defect for dataset from PROMISE repository [9]. The author designed model using seven requirements metrics, seven design and development metrics, and four testing phase metrics as input for prediction of faults. He also considered five linguistic levels very high, high, moderate, low, and very low for all input/output variables. The model works on the basis of directed acyclic graph where nodes are variables and links between nodes shows the relationships between variables. The values of output variables computed using probability distribution conditionally or unconditionally for the node/variables.

Pandey and Goyal [11] designed model using three metrics at requirements analysis phase, two metrics at design, two metrics at coding, and three metrics at testing phase as input. The results of the model are obtained at the end of each phase of SDLC as

indicator of fault density. The author designed a model using fuzzy logic and applied Matlab for implementation. The basic steps for the implementation of model included the recognition of process-level software metrics, designing of fuzzy rules to define links across inputs and output variables, and prediction of the fault by using fuzzy inference system (FIS).

As input/output variables are fuzzy number and author selected triangular fuzzy numbers (TFN) as type of the membership function of input/output variables. Also TFN are easy to work with real numbers.

The author has considered five linguistic level very high, high, moderate, low, and very low for all input variables and selected seven level very very High, very high, high, moderate, low, very low, and very very low for all output variables.

The author collected knowledge from various sources like literature, existing data, and domain experts to frame the fuzzy rules [14] [15]. The author designed rules on the basis of software engineering and project management concepts.

The designed fuzzy inference system performs mapping for the list of fuzzy rules for each inputs on to an output. Each rule describes the functionality of the mapping [16] [17]. The author considered Mamdani fuzzy inference system for the processing of dataset [18]. The defuzzification required to get a crisp number from a fuzzy set. The author considered centroid method for defuzzification because it returns the centre of gravity/area under curve [18].

Yadav [12] presented a software defect prediction model using fuzzy logic for the reliability metric as input for four basic phases of software development. The author used nine software metrics for four phases are requirement analysis, designing, coding, and testing to predict fault density. The input/output variables are fuzzy number and the author selected trapzodial fuzzy numbers as type of the membership function of input variables and triangular fuzzy numbers as type of the membership function of output variables. The author has considered three linguistic level high, moderate, and low for all input variables and selected five level very high, high, moderate, low, and very low for output variables.

The author designed the fuzzy rules on the basis of domain knowledge, related study, and software engineering concepts. The author used fuzzy inference tool of Matlab for implementation of model for each phase of software development and obtained results as defect density value for each phase.

P. Sharma and A.L. Sangal [13] presented a model design using the fuzzy linguistic system to calculate the defect density for different process metrics. The author presented framework using 21 process metrics for four phases are requirement analysis, designing, coding, and testing to predict fault density. The author selected triangular fuzzy numbers (TFN) as type of the membership function for input/output variables and considered five linguistic level very high, high, moderate, low, and very low for input/output variables.

At first, the author fuzzified process metrics with the use of membership functions defined with lingual data. After this, FIS was designed to compute the faults. Also, the author applied the back propagation algorithm to train the fuzzy set of rules so that the accuracy of prediction of the designed fuzzy inference system can be improved. The author repeated the training of model for each phase of software development at 100 epoch to get minimum error rate. Finally, to get crisp value from aggregated fuzzy set, the author used centroid method for defuzzification for each phase of software development.

The comparison of result for Software fault prediction based on various soft computing methodologies is described in table I. The prediction results of SFP model depend on and vary according to dependent variables. To measure the level of accuracy of the SFP model, there are various analytical methods available that can be apply to compare the performance of the designed SFP model. These methods can be either Continuous or Categorical studies. The Continuous studies related to difference between actual and predicted outcomes, goodness-of-fit, classified and appropriate outcomes, and accuracy. These measurements focused on predicting the number of defects.

To compare the accuracy of software fault prediction models using various soft computing modeling different analytical measures used are MMRE, BMMRE, RMSE, and NRMSE and computed measures for worst case.

Case study #	Project #	Size in KLOC	Actual Defect	Fenton (Bayesian Net) [10]	Pandey (FIS) [11]	Yadav (FIS) [12]	Sharma ANFIS [13]
1	1	6.02	148	75	56	155	156
2	2	0.9	31	52	5	30	37
3	3	53.86	209	254	210	205	191
4	5	14	373	349	232	----	362
5	7	21	204	262	113	209	185

6	8	5.79	53	48	53	53	52
7	10	4.84	29	203	26	31	36
8	11	4.37	71	51	40	84	80
9	12	19	90	347	176	97	85
10	13	49.1	129	516	336	142	131
11	14	58.3	672	674	697	----	620
12	15	154	1,768	1526	1650	1740	1730
13	16	26.67	109	145	127	101	102
14	17	33	688	444	135	733	690
15	19	87	476	581	573	446	422
16	20	50	928	986	869	955	935
17	21	22	196	259	105	192	174
18	22	44	184	501	291	194	156
19	23	61	680	722	690	-----	686
20	27	52	412	430	400	----	380
21	29	11	91	116	110	91	78
22	30	1	5	46	6	5	7

TABLE I
COMPARISON OF RESULT FOR SOFTWARE FAULT PREDICTION BASED ON VARIOUS SOFT COMPUTING METHODOLOGIES

Magnitude of Relative Error (MRE): The most frequently applied accuracy metrics are the magnitude of relative error. It can be computed on the basis of either the mean or the median. The MRE is always less than 1[13].

$$\text{MRE} = \frac{(\text{Actual defect} - \text{Predicted defect})}{\text{Actual defect}} \quad (4)$$

Mean MRE (MMRE): It is the average of magnitude of relative error values over N projects. The problem of the Mean MRE is its sensitivity to anomaly.

Median MRE (MdMRE): It is the median of MRE values over N projects. It is less sensitive to anomaly.

Balanced MMRE (BMMRE): As MMRE is unbalanced, so for this reason BMMRE is used as [13][19]:

$$\text{BMMRE} = \frac{1}{n}\sum_{i=1}^{n} \frac{|\text{Actual defect} - \text{Predicted defect}|}{\min(\text{Actual defect}, \text{Predicted defect})} \quad (5)$$

Root Mean Square Error (RMSE): It is calculated as the difference between actual and predicted values over N projects. The low value of RMSE gives the better result [13].

$$\text{RMSE} = \sqrt{1/n}\sum_{i=1}^{n}(\text{Actual defect} - \text{predicted defect})^2 \quad (6)$$

Normalized Root Mean Square Error (NRMSE): It is the same as the RMSE. It is computed by dividing RMSE to the range of actual value [13].

$$\text{NRMSE} = \frac{\text{RMSE}}{\text{Max (Actual Defect)} - \text{Min (Actual Defect)}} \quad (7)$$

	Fenton (Bayesian Net) [10]	**Pandey (FIS) [11]**	**Yadav (FIS) [12]**	**Sharma (ANFIS) [13]**
MMRE	0.7948	0.2523	0.02171	0.01646
BMMRE	0.7998	0.6055	0.0229	0.016861
NRMSE	3.454	0.627	0.0426	0.0327
RMSE	324.7	350.49	55.161	48.524

TABLE II
PERFORMANCE MEASURE

It shown from Table II and Figure 6 (a, b) that the Sharma [13]'s ANFIS based model shows better prediction accuracy for fault predictions in comparison of model by Fenton [10], Pandey and Goyal [11], and Yadav [12] model.

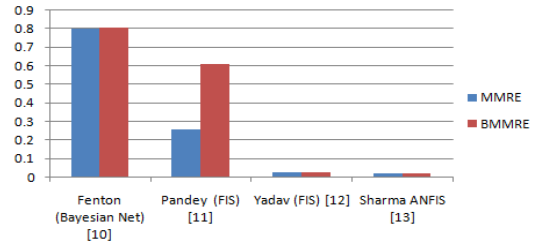


Fig. 6(a) Performance of various software fault prediction models based on soft computing approaches

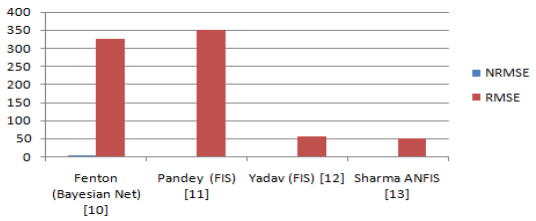


Fig. 6(b) Performance of various software fault prediction models based on soft computing approaches

V. CONCLUSION

This research paper compares the performance of fault prediction model designed based on soft computing approaches such as Bayesian net, FIS and ANFIS during software development by employing metrics from PROMISE data set.

Fenton [10] described software fault prediction model using Bayesian net technique.

Pandey and Goyal [11] predicted fault contents in a software product by applying a FIS based model which takes into account the process metrics. The author considered Mamdani fuzzy inference system for the processing of dataset and considered the centroid method to derive a crisp number from a fuzzy set, for defuzzification using MatLab tool.

Yadav [12] used fuzzy inference tool of Matlab for implementation of model for each phase of software development and obtained results as defect density value for each phase using nine software metrics.

P. Sharma and A.L. Sangal [13] fuzzified 21 process metrics with the use of membership functions defined with lingual data. After this, FIS was designed to compute the number of faults. Also, the author applied the back propagation algorithm to train the fuzzy set of rules so that the accuracy of prediction of the designed fuzzy inference system can be improved. To compare the validation of Pandey and Goyal [11], Yadav [12], P. Sharma and A.L. Sangal [13] model, data set from PROMISE repository related to twenty-plus projects has been used. Different analytical standard such as MMRE, BMMRE, RMSE, and NRMSE have been used for measuring the performance. It is remarked from the performance measures that P. Sharma and A.L. Sangal [13] ANFIS based model presents better fault prediction capability in contrast with the Fenton [10] Bayesain net based model, Pandey and Goyal [11], Yadav[12] FIS based model.

REFERENCE

[1] Chatterjee, S., Maji, B., "A bayesian belief network based model for predicting software faults in early phase of software development process", *Appl. Intell.*, vol. 48 pp 2214–2228, 2018. https://doi.org/10.1007/s10489-017-1078-x

[2] Bilgaiyan, S., Mishra, S. & Das, M., "Effort estimation in agile software development using experimental validation of neural network models", *Int. j. inf. tecnol.*, vol.11 pp 569–573, 2019. https://doi.org/10.1007/s41870-018-0131-2

[3] Kapil Juneja, "A fuzzy-filtered neuro-fuzzy framework for software fault prediction for inter-version and inter-project evaluation", *Elsevier Applied Soft Computing* ,vol. 77 pp 696-713, 2019 https://doi.org/10.1016/j.asoc.2019.02.008

[4] L. A. Zadeh, Knowledge representation in fuzzy logic, *IEEE Transactions on Knowledge and Data Engineering*, vol. 1 no.1 pp. 89–100, 1989.

[5] X. Zhang, H. Pham, An analysis of factors affecting software reliability, *Journal of Systems and Software*, vol. 50 no. 1 pp. 43–56, 2000.

[6] M. Li, C. Smidts, A ranking of software engineering measures based on expert opinion, *IEEE Transaction on Software Engineering*, vol. 29 no. 9 pp. 811–824, 2003.

[7] T. J. Ross, Fuzzy logic with engineering applications, *John Wiley & Sons publications*, 2nd Edition, 2009.

[8] Jang J S. R.,C.T. Sun, E. Mizutani, (1997) "Neuro Fuzzy and Soft Computing A Computation Approach to Learning and Machine Intelligence", *Matlab Curriculum Series*, Prentice Hall.

[9] http://tunedit.org/repo/PROMISE/DefectPrediction/qqdefects_numeric.arf

[10] Fenton, N. Neil, N. Marsh, W. Hearty, P. Radlinski, L. Krause, P., "On the efectiveness of early life cycle defect prediction with Bayesian nets", *Empirical Softw. Eng.*, vol. 13 pp 499–537, 2008.

[11] A. K. Pandey, N. K. Goyal, "Multistage model for residual fault prediction", *In Early Software Reliability Prediction, Springer India*, pp. 59–80, 2013.

[12] Yadav H.B., Yadav D.K., "A fuzzy logic based approach for phase-wise software defects prediction using software metrics" *Inf. Softw. Technol*, vol. 63 pp 44–57, 2015. https://doi.org/10.1016/j. infsof.2015.03.001

[13] Sharma P., Sangal A.L., "Building and Testing a Fuzzy Linguistic Assessment Framework for Defect Prediction in ASD Environment Using Process-Based Software Metrics", *Arabian Journal of Science and Engineering*, vol. 45 no.12 pp.10327–10351, 2020. https://doi.org/10.1007/s13369-020-04701-5.

[14] Xie, M., Hong, G.Y., and Wohlin, C., "Software reliability prediction incorporating information from a similar project", *The Journal of Systems and Software*, vol. 49 pp 43-48, 1999.

[15] Zhang, X., and Pham, H., "An Analysis of Factors Affecting Software Reliability", *The Journal of Systems and Software*, vol. 50(1) pp 43-56, 2000.

[16] Bowles, J.B., and Pelaez, C.E., "Application of fuzzy logic to reliability engineering", *IEEE Proceedings*, vol. 83 no. 3 pp. 435-449, 1995 .

[17] Zadeh, L.A., "Knowledge representation in fuzzy logic", *IEEE Transactions on Knowledge and Data Engineering*, vol.1 pp 89-100, 1989.

[18] Mamdani, E.H., "Applications of fuzzy logic to approximate reasoning using linguistic synthesis", *IEEE Transactions on Computers*, vol. 26(12) pp 1182-1191, 1977.

[19] T. Sethi, , "Improved approach for software defect prediction using artificial neural networks", *Proceedings of the 5th International Conference on in Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, pp. 480-485, 2016.

APPENDIX

#	Project	RC	RS	RIW	DTE	PM	CTE	DPF	TTE	SI	SIZE	Faults
		F1	S7	S3	D1	P9	D2	D3	T2	P5	K	TD
1	1	M	L	VH	L	H	H	H	H	H	6.02	148
2	2	L	H	VH	L	H	H	H	H	H	0.9	31
3	3	H	H	VH	H	VH	VH	H	H	VH	53.86	209
4	5	H	M	H	L	H	M	H	M	M	14	373
5	7	L	M	VH	M	H	VH	H	M	VH	21	204
6	8	M	H	H	H	M	H	M	M	H	5.79	53
7	10	M	H	H	H	H	H	H	M	H	4.84	29
8	11	H	H	H	H	H	H	H	H	H	4.37	71
9	12	H	L	H	VH	H	M	M	H	H	19	90
10	13	H	L	M	H	H	H	H	M	H	49.1	129
11	14	VH	H	H	H	H	H	H	H	H	58.3	672
12	15	H	VL	H	H	H	H	H	H	VH	154	1,768
13	16	L	M	H	H	H	H	H	H	VH	26.67	109
14	17	L	M	M	M	H	M	H	L	M	33	688
15	19	H	M	H	H	H	H	H	M	H	87	476
16	20	VH	VL	M	VL	H	VL	L	VL	H	50	928
17	21	L	M	H	H	H	H	H	H	H	22	196
18	22	M	L	M	H	H	M	L	M	H	44	184
19	23	H	M	VH	L	H	H	H	H	H	61	680
20	27	H	M	VH	M	H	L	M	M	M	52	412
21	29	M	VH	VH	VH	H	VH	H	VH	VH	11	91
22	30	L	VH	VH	H	H	H	H	H	VH	1	5

TABLE I
DATA SET WITH LINGUISTICVALUE

#	Project	RC	RS	RIW	DTE	PM	CTE	DPF	TTE	SI	SIZE	Faults
1	1	0.34	0.15	0.92	0.15	0.75	0.63	0.75	0.63	0.63	0.15	148
2	2	0.15	0.63	0.92	0.15	0.75	0.63	0.75	0.63	0.63	0.15	31
3	3	0.63	0.63	0.92	0.63	0.92	0.86	0.75	0.63	0.86	0.86	209
4	5	0.63	0.34	0.75	0.15	0.75	0.34	0.75	0.34	0.34	0.34	373
5	7	0.15	0.34	0.92	0.34	0.75	0.86	0.75	0.34	0.86	0.63	204
6	8	0.34	0.63	0.75	0.63	0.5	0.63	0.5	0.34	0.63	0.15	53
7	10	0.34	0.63	0.75	0.63	0.75	0.63	0.75	0.34	0.63	0.15	29
8	11	0.63	0.63	0.75	0.63	0.75	0.63	0.5	0.63	0.63	0.15	71
9	12	0.63	0.15	0.75	0.86	0.75	0.34	0.5	0.63	0.63	0.34	90
10	13	0.63	0.15	0.5	0.63	0.75	0.63	0.75	0.34	0.63	0.63	129
11	14	0.86	0.63	0.75	0.63	0.75	0.63	0.75	0.63	0.63	0.86	672
12	15	0.63	0.05	0.75	0.63	0.75	0.63	0.75	0.63	0.86	0.86	1,768
13	16	0.05	0.34	0.75	0.63	0.75	0.63	0.75	0.63	0.86	0.63	109
14	17	0.05	0.34	0.5	0.34	0.75	0.34	0.75	0.15	0.34	0.63	688
15	19	0.63	0.34	0.75	0.63	0.75	0.63	0.75	0.34	0.63	0.86	476
16	20	0.86	0.05	0.5	0.05	0.75	0.05	0.25	0.05	0.63	0.86	928
17	21	0.05	0.34	0.75	0.63	0.75	0.63	0.75	0.63	0.63	0.63	196
18	22	0.34	0.15	0.5	0.63	0.75	0.34	0.25	0.34	0.63	0.63	184
19	23	0.63	0.34	0.92	0.15	0.75	0.63	0.75	0.63	0.63	0.86	680
20	27	0.34	0.34	0.92	0.34	0.75	0.15	0.5	0.34	0.34	0.86	412
21	29	0.34	0.86	0.92	0.86	0.75	0.86	0.75	0.86	0.86	0.34	91
22	30	0.05	0.86	0.92	0.63	0.75	0.63	0.75	0.63	0.86	0.15	5

TABLE II
DATA SET WITH CENTRIOD VALUE

★★★