STM JOURNALS

Journal of Artificial Intelligence Research & Advances
ISSN: 2395-6720 (Online)
Volume 6, Issue 1
www.stmjournals.com

Test Effort Estimation Based Upon Neural Fuzzy Model

Vikas Chahar[1], Pradeep Kumar Bhatia[2,*]
[1]Department of Computer Science, Vaish (P.G.) College, Rohtak, Haryana, India
[2]Department of Computer Science and Engineering, Guru Jambheshwar University of Science and Technology, Hisar, Haryana, India

Abstract

Estimating test development effort is an important task in the management of large software projects. The task is challenging and it has been receiving the attentions of researchers ever since software was developed for commercial purpose. A number of estimation models exist for effort prediction. However, there is a need for neural model to obtain more accurate estimations. The primary purpose of this study is to propose a precise method of estimation by selecting the most popular models in order to improve accuracy. In this paper, we explore the use of soft computing techniques to build a suitable model structure to utilize improved estimation of software effort; a comparison between neural network (NN) and neural fuzzy model; and the evaluation criteria are based upon MRE and MMRE. Consequently, the final results are very precise and reliable when they are applied to a real dataset in a software project. The results show that NF is effective in effort estimation.

Keywords: *Effort estimation, neural network, fuzzy model, MRE, MMRE*

****Author for Correspondence*** E-mail: pkbhatia.gju@gmail.com

INTRODUCTION

In the last three decades, many quantitative software cost estimation models have been developed. According to Marco, reliable prediction of size and effort in software development projects is a necessary prerequisite to develop reliable cost and schedule estimates. The size and development effort measures, such as function points or lines of code developed per person-month, act as technical productivity and performance indicators that facilitate the tracking and control of software developments. The size and advancement exertion measures, for example, work focuses or lines of code created per individual month, go about as specialized profitability and execution pointers that encourage the following and control of programming improvements. An exact model uses information from past undertakings to assess the present task and gets the essential formulae from examination of the specific database accessible. An explanatory model, then again, utilizes formulae in light of worldwide presumptions, for example, the rate at which engineers take care of issues and the quantity of issues accessible. Assessment of numerous product models was displayed and numerous models were investigated to give better exertion estimation creators to give an overview on the exertion and cost estimation models. This paper is created as takes after: Next part of the paper consolidates the item dependability work analyzed; then it demonstrates the proposed factors impacting programming unwavering quality. Fuzzy model is proposed, NN strategy used as a piece of paper, a neuro-fuzzy based approach is spoken to, and end results are shown in the following parts.

LITERATURE REVIEW

Software program enduring quality can assess the level of highlights that are reused in regulating application. Reusing presented programming sections is a key part in developing adequacy. The estimation of programming reuse can be speedily seen by the expanding of the open source social event of organizers and reusers of programming. The general reuse of present subprogram requires to be kept up by instruments that would commit to have the capacity to the two creators: those who make reusable subprogram

and customers who reuse existing subprogram. Measures are customary to meet the basic needs of the two organizers and clients of the thing to check that the progression of programming for reuse can outfit the two get-togethers with steady help that watches out for the issues of the two social events. Entire events live required so every mix will evaluate programming system, at various thoughts every through it change, to settle on a decision degree to that the stock is set up for utilize. Challenge facilitated undertaking and advancement has considered again a general approach of programming structure change [1]. Here, square measure amounts of sums accessible for preparing the reliableness of test arranged assents [2, 3]. These metrics are overviewed on the start of Weyuker's properties [4]. From the contingent outcomes, metrics have been seen to be sensible in finding the level of immovable quality related with the code in the procedure for class and limit outlines. It proposed neuro-fleecy model for subprogram resolute quality figuring with upgraded choice. The resolute quality assessment framework has passed on high accuracy. Therefore, the influenced structure can be utilized for indisputable affirmation and departure of OO based reusable portion from inheritance system and evaluation of set up or making reusable fragment. Poulin exhibited an arrangement of estimations utilized by IBM to evaluate the fights saved by reuse. The examination recommends the potential inclinations against the livelihoods of time and assets required to perceive and meld reusable programming into a segment. Schach and Yang proposed estimations for focusing on believability for reuse. Test comes to fruition developed that a guess of resolute quality is conceivable anyway, it contains more than the plan of estimations that are closely used. Gill dissected differing issues concerning part faithful quality and its purposes of intrigue like cost and profitability [5]. He dissected assorted issues concerning part steady quality and its purposes of intrigue like cost and capable. He gives two or three rules to extend the level of subprogram steadfast quality in section base headway. Washizaki learned an estimations suite for choosing the unflinching nature of disclosure portion made on kept information that can be obtained from the external part with no source code. Because of estimation tests, it was found that proposed estimations can adequately perceive revelation sections with high trustworthiness [5]. Rotaru *et al.* pondered adaptability, impact cutoff and capriciousness of part to depict its resolute quality. Mili *et al.* considered two points of view, ease of use and solace while REBOT (Reuse Based on Object Oriented Techniques) saw factors as specific transportability, adaptability, understandability and confirmation to evaluate the constancy. Sandhu *et al.* proposed woolen, neuro-cushioned and soft GA based ways to deal with oversee assess the immovable nature of programming areas. Fleecy GA mutt estimation is ended up being best when showed up diversely in connection to trade checks. From the beguilement and result acquired, it has been demonstrated that the percentage common blunder is minimum by excellence of Fuzzy-GA figuring and most imperative on account of feathery estimations [10-15]. The made unfaltering quality show has passed on high precision works out as intended as anybody may expect by the human experts. Sharma *et al*. proposed Artificial Neural Network (ANN) to survey the unfaltering nature of programming part. Results picked up display that neural system can foresee the reliability of programming part with hurt for accuracy.

ELEMENT DISTRESSING SOFTWARE RELIABILITY

In perspective of the investigators see, we have proposed four properties: Changeability (CH), Interface Complexity (IC), Understandability of Subprogram (UOS) and Documentation Quality (DQ) to envision subprogram relentless quality level thus using sensitive figuring frameworks. Their purposes of intrigue are according to the accompany:

Changeability

Variability is described as the capacity to adjust a section as demonstrated by the application require. Better changeability will actuate a section with better trustworthiness in applications and along these lines help in keeping up the portion. Change-cutoff of a section may shift from 0 to 1. In our examination, we sort it from low to high [16-20].

Complexity
Presence the environments state of the segment, the source code is not accessible. Application may interrelate through these fragments simply whole their especially portrayed interface. Interface goes about as basic hotspot for perception, use and execution finally upkeep for the portion. For better constancy, interface disperse quality ought to be as low as could be permitted. In our work, we compose it as low, medium and high.

Portability
The farthest point of a portion to be exchanged starts with one condition, then onto the accompanying with little change, if required. For better resolute quality, part should be reasonably and rapidly profitable to showed new conditions if and when key, with compelled porting tries and logbooks. On the off chance that the part are organize free, by then it is outstandingly useful and from now on its choice undertakings are low. The conservativeness of a segment partners effortlessly of use. The straightforwardness of transportability of an area can pick its level of reuse [21-25].

PREDICTABLE FUZZY LOGIC ILLUSTRATION

All in this area we have proposed fuzzy model and master a Fuzzy Inference System (FIS) with these four components, to be specific variability, multifaceted nature, portability and coupling, reliability as yield, as appeared in Figure 1.

All data sources can be all around engineered into fuzzy sets viz., low, medium and high. The yield subprogram faithful quality is delegated as: low, low, medium, high and very high. Remembering the ultimate objective to fuzzy the information sources, triangular enlistment work (trimf) is supported specifically, low, medium and high. Correspondingly the yield variable i.e. subprogram unfaltering quality eats up three enlistment limits (trimf), to be particular, low, medium and high [26-30]. The proposed show thinks each one of the four data sources and passes on a new estimation of subprogram steady quality using rule base. Each one of the 27 standards are inserted and a run base is made as showed up in Figure 2, which addresses all possible blends of wellsprings of information which prompts 33, i.e. 27 sets.

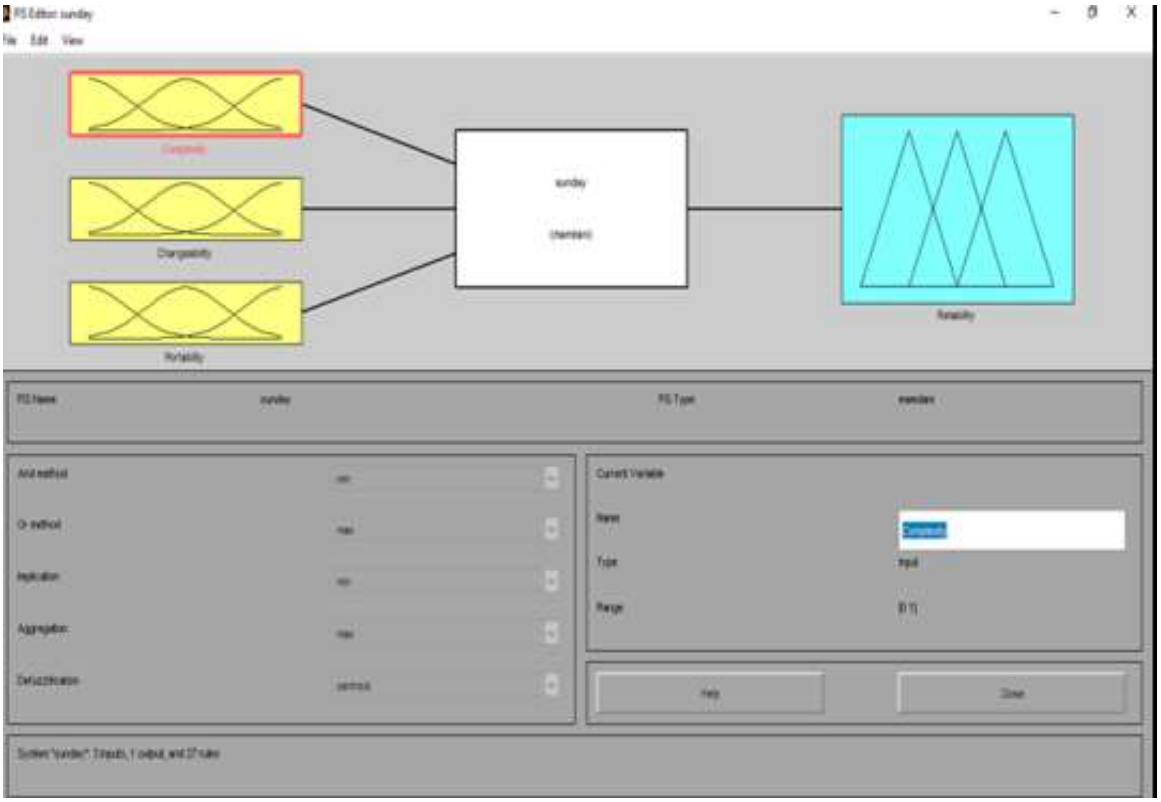

Fig. 1: *FIS with 3 Inputs, 1 Output, among 27 Rules*

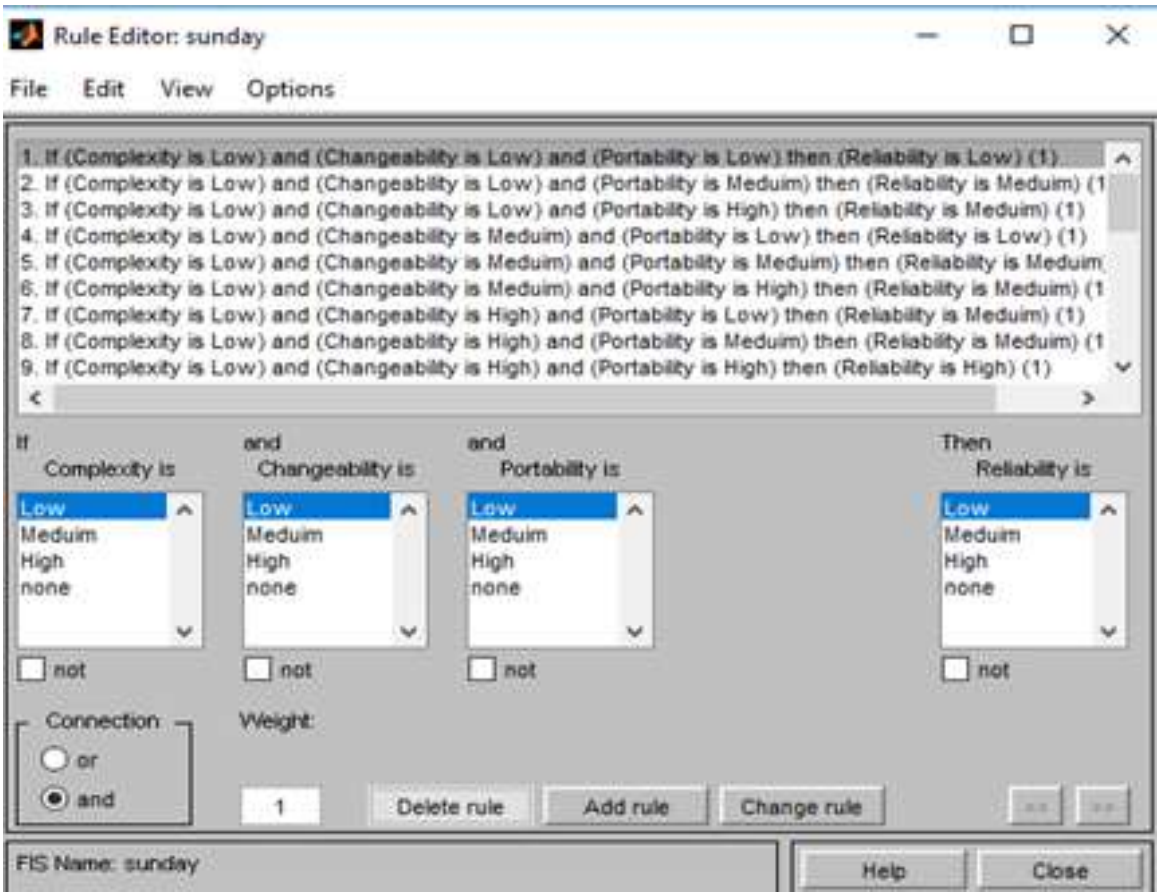

Fig. 2: *Rule Support for Predictable Fuzzy Model.*

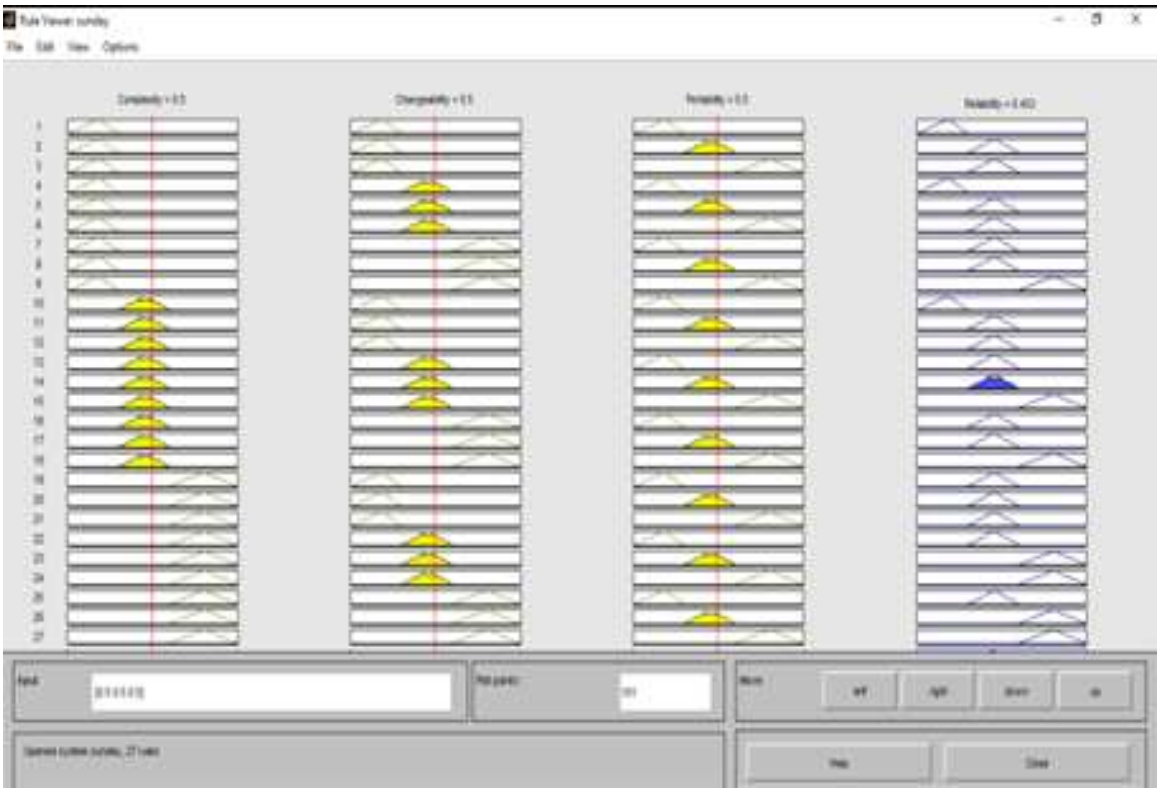

Fig. 3: *Considered Fuzzy Model Rule Viewer.*

Mamdani style of induction is used as showed up in Figure 1. Using the run watcher, yield i.e. subprogram resolute quality is looked for particular game plan of data sources using MATLAB Fuzzy Tool Box as showed up in Figure 3. Defuzzification of the above yield can be gotten by finding the center of gravity of the above fuzzy yield. The aftereffect of these frameworks was perceived by envisioning the model in MATLAB Fuzzy Tool Box. The proposed fuzzy model is used for influencing the data to set with partitioned yield as a commitment to the neural network for support endorsement.

DESIGNED NEURAL NETWORK MODEL

Here we have explored the capacity of machine learning frameworks i.e. neural frameworks as they are flexible, have learning capacities and non-parametric. The enlightening accumulations created by the fuzzy model presently go about as data datasets to our proposed neural network model. It is seen that NN model can be used to show any self-assertive data yield mapping and are talented of approximating any quantifiable limit. So NN should have the ability to show the helpfulness of the subprogram reliability additionally. In a wide sense, the neural framework itself is a model in light of the fact that the topology and trade components of the centers are ordinarily figured to facilitate the present issue [6]. The NN show is moreover affirmed on the two endorsement sets delivered by the Fuzzy models. In our examination back spread coordinated estimation is used. Back spread is the most unmistakable getting ready computation for multilayer systems [7]. The model is set up with four wellsprings of data to be particular: Changeability, Interface Complexity, Understandability of Subprogram and Documentation Quality with one yield, i.e. subprogram reliability. Trainbr, getting ready limit is used to set up the framework. We have reused the straight transmission limits Tansig for the testing. The framework is best arranged with four data sources, one yield and four neurons at the covered layer as showed up in Figure 4.

Tentative Design

The preliminary is coordinated to examine the potential results of utilization of NN for reviewing subprogram steadfast quality level. The framework used as a piece of this work has a place with multilayer maintain forward frameworks and is implied as M-H-Q arrange with M source centers, H center points in hid sanctum layer and Q centers in the yield layer [8].

The once-over of ANN used as a piece of this examination is showed up in Table 1. Around there we demonstrate the examination performed to find association send between self-sufficient components (changeability, interface flightiness, understandability of subprogram and documentation quality) and one time tested factor i.e. subprogram steadfast quality. The informational index contains 80 unique informational collections created by the control base of fuzzy rationale. The execution importance utilized here is Mean Absolute Error (MRE) and Mean Absolute Relative Error (MARE) [9]. The model is prepared utilizing preparing informational collections and was assessed on two approval informational indexes. Table 2 shows the Mean Absolute Relative Error, and Mean Relative Error results of ANN demonstrate assessed on affirmation informational collection. Examination of the exploratory outcomes and estimated subprogram unwavering quality of the two approval; from which it winds up plainly obvious that our proposed show produces unwavering quality levels that are practically identical to the deliberate dependability levels as anticipated by the proposed fuzzy model.

Comparison of the experimental results and measured software reliability of the two validation sets. From which it becomes evident that our proposed model produces reliability levels that are comparable to the measured reliability levels as predicted by the proposed fuzzy model.

Table 1: *Neural Network Model Review.*

Structural Design	
Number of Neurons at Layer 1	03
Number of Neurons at Layer 2	01
Input unit	03
Output Unit	01
Training	
Used Algorithm	Back Propaga
Function used for Training	Trainbr
Transfer Function	Tansig

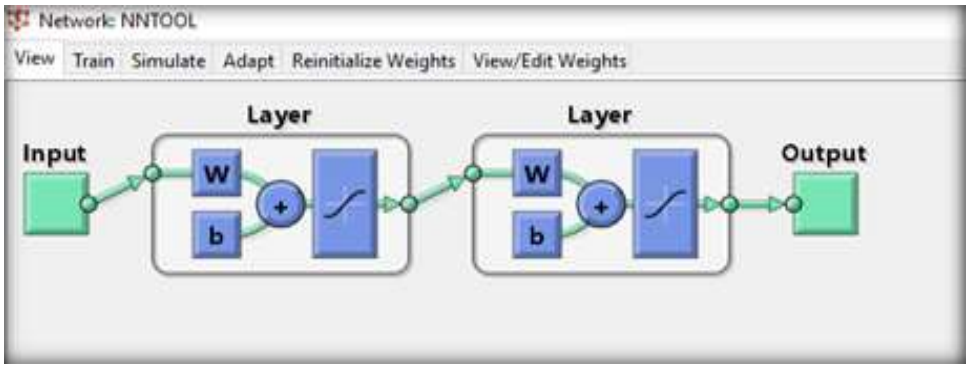

Fig. 4: *Prepare Neural System Model by Three Input and One Output and Three Neurons at the Hidden Level.*

DELIBERATE NEURO-FUZZY SYSTEM

Fluffy rationale approach is profitable for assessing the immovable nature of subprogram fragments as the conventional model based strategies are difficult to be realized. Disastrously, with the extension in the disperse nature of the issue being shown distinctive comprises for assessing the unfaltering quality, has incited relying upon another framework which, and a short time later through yield enrolment limits is by and large known as neuro-fluffy approach. The neuro-fluffy cross breed structure joins the benefits of fluffy rationale system, which deal through express data that can be illuminated and appreciated, and neural frameworks, which oversees comprehended data, which can be acquired by learning. Neuro-fuzzy inference structure using given information/yield dataset has been assembled a Fuzzy Inference System (FIS) whose interest work parameters are tuned (adjusted) using either a back inciting computation alone or in mix with the base squares kind of strategy. The parameters related with the support work (trimf is used here) will change through the learning system. The estimation of these parameters (or their change) is supported by an edge vector, which gives a proportion of how well the fluffy construing system is exhibiting the data/yield for a given game plan of parameters. Sugeno compose Neuro-Fluffy finding system is used for the preliminary purposes, as Sugeno yield part works are either predictable or coordinate.

Neuro-Fuzzy Inference System Training

To prepare an ANFIS, preparing informational collection (informational collections produced by fuzzy model) is stacked from the heap information choice of ANFIS editorial manager as appeared in Figure 5.

In the fluffy induction system, semantic variables are then named to the data parameters in light of their regards.

The assignment of the semantic factors depends upon the extent of the data estimation. CH, IC, UOS and DQ are assigned three phonetic factors Low, Medium and High. The yield i.e. steady quality is consigned five etymological factors Low, Medium and High as inspected in the proposed fluffy model of this paper. A framework composes structure like that of a neural framework (as demonstrated Figure 6) which maps commitment through data enlistment work and related parameters to yield, is used to unravel the data/yield layout. After viably stacking the data, FIS structure will be delivered using cross section overseeing as showed up in Figure 6. The slip-up strength is used to make planning ending criteria, which is related to the goof gauge. The readiness will stop after the arrangement data botch remains inside this flexibility. In our examination, we have kept screw up protection from a regard 0.0044. To start getting ready, cross breed change (the default, mixed scarcest squares and back causing) procedure is used [31-33].

The quantities of preparing ages is set to 7 and the preparation blunder diminishes after every age as appeared by the Figure 7 and balances out at the mistake estimation of 0.0374, so now the system is said to be focalized.

Table 2: *Authentication Conclusion of Neural Network Model.*

Performance Measures	Verification Set 1	Verification Set 2
Mean absolute Relative Error	0.20	0.40
Mean Relative Error	0.25	0.90

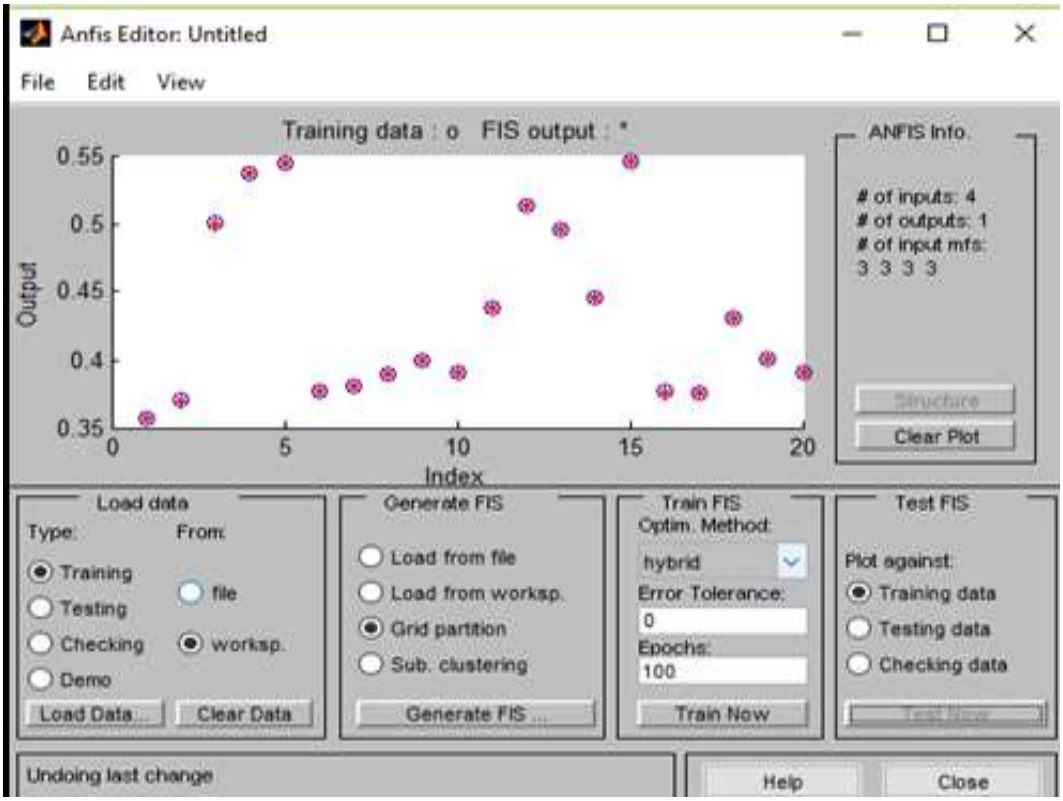

Fig. 5: *Training Statistics Set for Artificial Neural Fuzzy Inference System.*

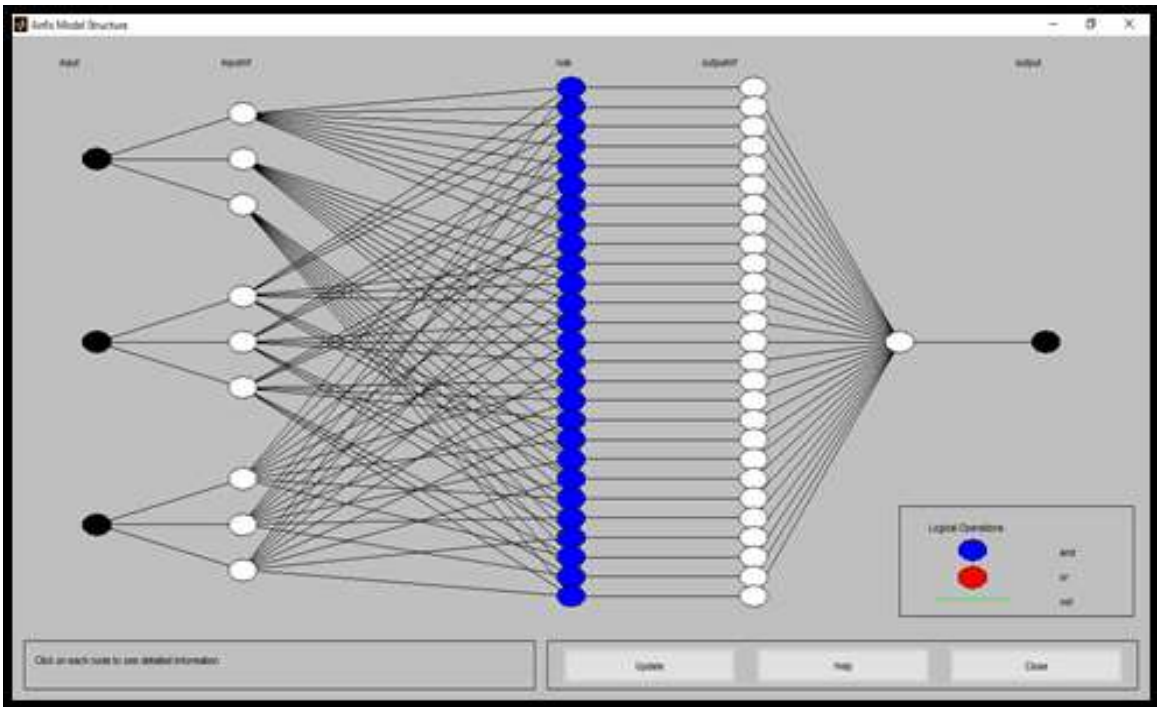

Fig. 6: *NN Integrating the FIS.*

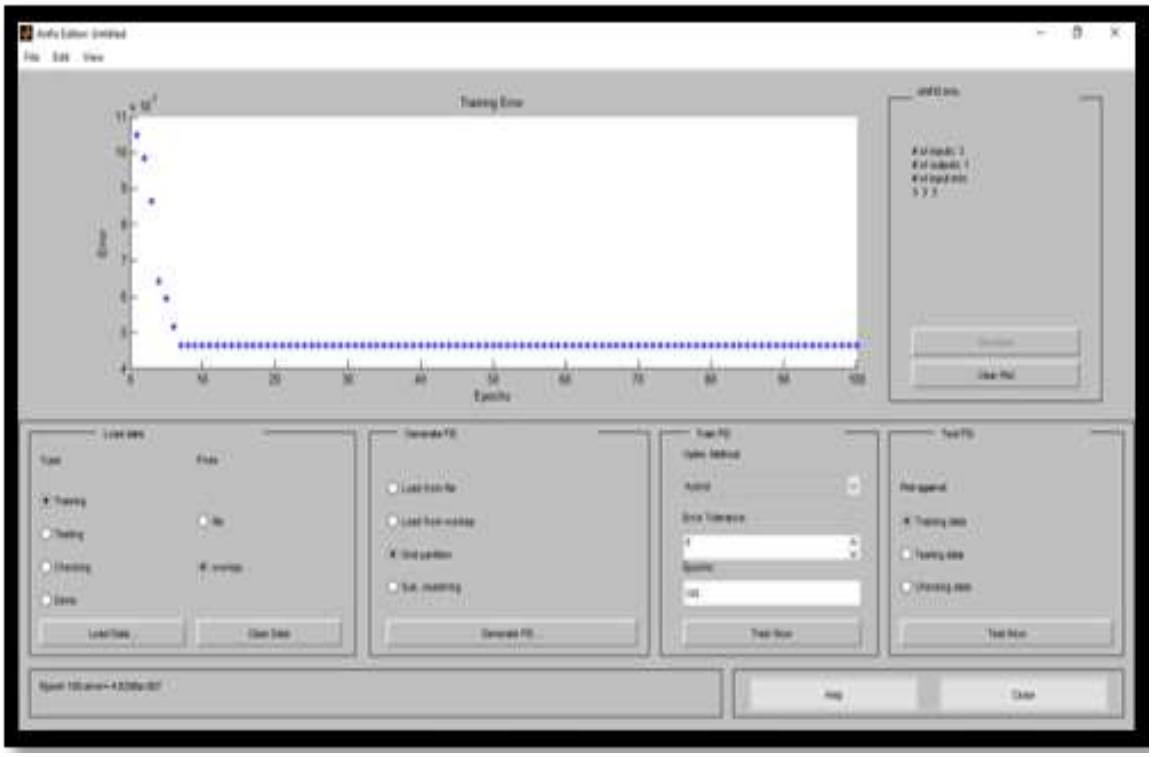

Fig. 7: *Design of Instruction Error vs. Epochs.*

Table 3: *Substantiation Results of NF Model.*

Performance Measures	Verification Set 1	Verification Set 2
Mean Absolute Relative Error	0.19	0.29
Mean Relative Error	0.18	0.34

Extent Fuzzy Inference Structure

Through the troublesome stage, when made structure is attempted against the endorsement enlightening accumulations and a typical testing botch 0.0374 is gotten. The execution work used here is Mean Relative Error (MRE) and Mean Absolute Relative Error (MARE) [9]. The model is readied using getting ready enlightening lists and is surveyed on two endorsement instructive accumulations. Table 3 shows the Mean Absolute Relative Error and MRE delayed consequences of neuro-fluffy model evaluated on two endorsement enlightening accumulations. The table in like manner demonstrates that the proposed exhibit arranged and predicts well using neuro-fuzzy approach.

CONCLUSIONS

The presentation of the neural network based effort estimation system and the other model, the results show that the neural fuzzy has the lowest MRE and MARE values i.e. 0.19 and 0.18 respectively. The second best performance is shown by software estimation system with 0.20 and 0.25 as MRE and MARE values. Hence, the proposed neuro based system is able to provide good estimation capabilities. It is suggested to use of neuro based technique to build suitable generalized type of model that can be used for the software effort estimation of all types of the projects.

REFERENCES

1. Boraso M, Montangero C, Sedehi H. *Software Cost Estimation: An Experimental Study of Model Performances.* Tech. Report; 1996.
2. Menzies T, Port D, Chen Z, *et al.* Validation Methods for Calibrating Software Effort Models. In *ICSE'05*: *Proceedings of the 27th International Conference on Software Engineering*, New York, NY, USA: ACM Press; 2005; 587–595p.
3. Benediktsson O, Dalcher D, Reed K, *et al.* COCOMO Based Effort Estimation for Iterative and Incremental Software Development. *Software Qual J.* 2003; 11(4): 265–281p.

4. Devnani-Chulani S. *Modeling Software Defect Introduction.* Tech. Report; 1997.
5. Clark B, Devnani-Chulani S, Boehm B. Calibrating the COCOMO II Post-Architecture Model. In *ICSE'98: Proceedings of the 20th International Conference on Software Engineering*, (Washington, DC, USA). IEEE Computer Society; 1998; 477–480p.
6. Chulani S, Boehm B. *Modeling Software Defect Introduction and Removal: Coqualmo (Constructive Quality Model).* Tech. Report; 1999.
7. Chulani S, Boehm B, Steece B. Calibrating Software Cost Models Using Bayesian Analysis. *IEEE Trans Softw Eng.* Jul–Aug 1999; 25(4): 573–583p.
8. Shepper M, Schofield C. Estimating Software Project Effort Using Analogies. *IEEE Trans Softw Eng.* 1997; 23(12): 736–743p.
9. Witting G, Finnie G. Estimating Software Development Effort with Connectionist Models. In *Proceedings of the Information and Software Technology Conference*. 1997; 469–476p.
10. Bailey JW, Basili VR. A Meta Model for Software Development Resource Expenditure. In *Proceedings of the International Conference on Software Engineering.* 1981; 107–115p.
11. Sentas P, Angelis L, Stamelos I, *et al.* Software Productivity and Effort Prediction with Ordinal Regression. *J Inf Softw Technol.* 2005; 47(1): 17–29p.
12. Witting G, Finnie G. Using Artificial Neural Networks and Function Points to Estimate 4GL Software Development Effort. *J Information Systems.* 1994; 1(2): 87–9p.
13. Khoshgoftaar TM, Allen EB, Xu Z. Predicting Testability of Program Modules Using a Neural Network. *Proc. 3rd IEEE Symposium on Application-Specific Systems and Sof. Eng. Technology*. 2000; 57–60p.
14. Chulani S, Boehm B. *Modeling Software Defect Introduction and Removal: Coqualmo (Constructive Quality Model).* Tech. Report; 1999.
15. Chulani S, Boehm B, Steece B. Calibrating Software Cost Models Using Bayesian Analysis. *IEEE Trans Softw Eng.* Jul–Aug 1999; 573–583p.
16. Shepper M, Schofield C. Estimating Software Project Effort Using Analogies. *IEEE Trans Software Engineering.* 1997; 23(12): 736–743p.
17. Witting G, Finnie G. Estimating Software Development Effort with Connectionist Models. In *Proceedings of the Information and Software Technology Conference.* 1997; 469–476p.
18. Bailey JW, Basili VR. A Meta Model for Software Development Resource Expenditure. In *Proceedings of the International Conference on Software Engineering*. 1981; 107–115p.
19. Sentas P, Angelis L, Stamelos I, *et al.* Software Productivity and Effort Prediction with Ordinal Regression. *Journal Information and Software Technology*. 2005; 47(1): 17–29p.
20. Witting G, Finnie G. Using Artificial Neural Networks and Function Points to Estimate 4GL Software Development Effort. *J Information Systems.* 1994; 1(2): 87–9p.
21. Khoshgoftaar TM, Allen EB, Xu Z. Predicting Testability of Program Modules Using a Neural Network. *Proc. 3rd IEEE Symposium on Application-Specific Systems and Sof. Eng. Technology*. 2000; 57–60p.
22. Heiat A. Comparison of Artificial Neural Network and Regression Models for Estimating Software Development Effort. *J Inf Softw Technol.* 2002; 44(15): 911–922p.
23. Huang SJ, Lin CY, Chiu NH. Fuzzy Decision Tree Approach for Embedding Risk Assessment Information into Software Cost Estimation Model. *J Inf Sci Eng.* 2006; 22(2): 297–313p.
24. Srinivasan K, Fisher D. Machine Learning Approaches To Estimating Software Development Effort. *IEEE Trans Softw Eng.* 1995; 21(2): 126–137p.
25. Prasad Redd PVGD, Hari CHVMK, Srinivasa Rao T. Multi Objective Particle Swarm Optimization for Software Cost Estimation. *International Journal of Computer Applications (IJCA) (0975-8887).* Oct 2011; 32(3): 13–17p.
26. Peram Subba Rao, Venkata Rao K, Suresh Varma P. A Novel Software Interval Type-2 Fuzzy Effort Estimation Model using S-Fuzzy Controller With Mean and

Standard Deviation. *International Journal of Computer Engineering & Technology (IJCET)*. 2013; 4(3): 477–490p.
27. Rao BT, Sameet B. A Novel Neural Network Approach for Software Cost Estimation Using Functional Link Artificial Neural Network. *Int J Comput Sci Netw Security*. Jun 2009; 9(6): 126–131p.
28. Zeng H, Rine D. Estimation of Software Defects Fix Effort Using Neural Network. *IEEE 28th Annual International Computer Software and Applications Conference (COMPSAC'04),* Los Alamitos. 28–30 Sep 2004; 2: 20–21p.
29. Agarwal KK, Chandra P, *et al*. Evaluation of Various Training Algorithms in a Neural Network Model for Software Engineering Applications. *ACM SIGSOFT Software Engineering Notes*. Jul 2005; 30(4): 1–4p.
30. Nageswaran S. Test Effort Estimation Using Use Case Points (UCP). *14th International Software/Internet Quality Week*, San Francisco. 29 May–1 Jun 2001.
31. Hastings TE, Sajeev ASM. A Vector-Based Approach to Software Size Measurement and Effort Estimation. *IEEE Trans Softw Eng*. Apr 2001; 27(4): 337–350p.
32. Kushwaha DS, Misra AK. Software Test Effort Estimation. *ACM SIGSOFT Software Engineering Notes.* May 2008; 33(3).
33. Sandhu PS, Bassi P, Brar AS. Software Effort Estimation Using Soft Computing Techniques. *World Academy of Science, Engineering and Technology*. 2008; 46: 488–491p.

Cite this Article
Vikas Chahar, Pradeep Kumar Bhatia. Test Effort Estimation Based Upon Neural Fuzzy Model. *Journal of Artificial Intelligence Research & Advances*. 2019; 6(1): 76–85p.