International Journal of
Advances in Scientific Research and Engineering (ijasre)

E-ISSN : 2454-8006

DOI: 10.31695/IJASRE.2021.1.34014

Volume 7, Issue 5
May - 2021

Software Fault Prediction Based on Interval Type-2 Intuitionistic Fuzzy Logic System

Imo J. Eyoh, Edward N. Udo and Ini J. Umoeka

Department of Computer Science,

University of Uyo, Uyo,

Akwa Ibom State,

Nigeria.

ABSTRACT

With the continuous expansion and innovations in modern software development, the rate at which defects are present in software is directly proportional to how sophisticated and complex the software tends to be. Software fault prediction therefore continues to remain an important area of research in software engineering especially as new modeling algorithms are still emerging. In spite of the fact that the potential implementations of fuzzy set theory in software fault prediction have been explored in the past, to the best knowledge of the authors, it has not yet examined how interval type-2 intuitionistic fuzzy sets with membership and non-membership degrees could be used with parameter optimization in this domain. Therefore, this work aims to adopt an interval type-2 intuitionistic fuzzy logic system to predict fault in the requirement phase of Software Development Life Cycle. Intuitionistic fuzzy logic system deals with uncertainty using separate degrees of membership and non-membership of an element to a set as well as hesitation index, therefore becoming more appropriate and flexible tool to deal with imprecision and vagueness in data. Experimental analyses show that the obtained software prediction outputs using interval type-2 intuitionistic fuzzy logic system are very close to the actual outputs which confirm that the proposed approach is a more realistic alternative for software fault prediction.

Key Words: *Software Fault Prediction, Fuzzy Logic System, Interval Type-2 Intuitionistic Fuzzy Logic System.*

---

1. INTRODUCTION

The use of software in various domains of human life is increasing daily as humans are heavily leaning on software for their survival and functionalities [1]. This is evidenced in the fact that all human endeavours such as medicine, education, defense, transportation, administration, marketing, entertainment, industry and many more are directly or indirectly affected by software. Software have become a key determining factor and measures for successful development and operations of core human activities for industrialization, technology and evolution. As stated by Kaur and Sharma [2] all software have the tendency to contain faults (situations influencing a software to carry out its desired operation efficiently) as it is almost impossible to produce fault free software [3]. It is important to also note that not all of these faults result in failure (inability of a software system to perform as specified) [4]. Therefore, one of the objectives of fault prediction in software is to reduce bugs (faults) before failure occurs. Since there are many software defects which may cause serious problems, software fault prediction system is needed to overcome this situation [5].

With continuous expansion and innovations in modern software development, software reliability has become a key concern [6] because as software expands with the emergence of increasingly sophisticated software systems, the rate of software defects (bugs) in the software increases directly proportionally [5], [7]. It is therefore continuously necessary to ensure that software is reliable and useful at all times in areas with which they are deployed in meeting the needs of users in these varied environments. The aim of software quality practitioners is to evaluate software quality level periodically so as to identify quality issues early [8]. It is quite easy to find and fix bugs in the early stages of Software Development Life Cycle (SDLC) and this will drastically reduce the cost of testing in later stages. Knowing faulty modules prior to software testing makes testing more effective and aids in the production of a reliable software. Every organization wants to assess the quality of the software product as early as possible so that poor software design can be improved or redesigned which will lead to significant savings in the development costs, decrease in development time, hence more reliable software [9], [10], [11]. Software reliability is defined as the probability of

failure-free operation for a specified period in a specific environment [1]. Reliable software can therefore be seen as a software that is void of faults and one way of improving software reliability is predicting fault before testing phase [12]. Building fault-free software is one of the desires of software developers and a basic method of evaluating software reliability is checking the presence of defects in a software [13]. Software defect (fault) prediction is a process of constructing machine learning classifiers to predict defective code snippets using historical information in software repositories. The predicted results can assist developers to find and fix potential defects thereby improving software reliability [6]. Software fault prediction therefore aims at identifying fault-prone software modules by using some underlying properties of the software project before the actual testing process begins [14]. Software fault prediction, as an important aspect of preventive maintenance, is the application of different computing techniques to predict or identify possible faults in a software before the software actually fails [3]. Software maintenance is a continuous process that exists throughout the entire lifespan of a software product to ensure that the software performs as expected.

In a view to ensuring the development of quality software product, several software defect prediction schemes have been proposed by using different software metrics, statistical methods and techniques as well as adopting different types of machine learning prediction and classification algorithms. Some of these approaches use past data related to faults and software metrics to predict the faulty modules [15] while some approaches build prediction models using historical as well as current development data to make predictions about faultiness of software subsystems and modules [11]. For instance, Basili et al. [16] investigated the impact of object-oriented design metrics suite on the prediction of fault-prone classes using logistic regression; Khoshgoftaar et al. [17] investigated the use of neural network as a model for predicting software quality using large telecommunication system to classify modules as fault prone or not fault-prone; Khoshgoftaar et al. [18] applied regression trees with classification rule to classify fault-prone software modules using data from a very large telecommunications system.; Parvinder et al. [19] used genetic algorithm for early detection of defect in software modules; Kanmani et al. [20] applied neural network in object-oriented software to predict defect; Malhotra and Jain [9] estimated fault proneness using object-oriented Chidamber and Kemerer (CK) metrics and Quality Model for Object Oriented Design (QMOOD) metrics by applying logistic regression and six machine learning methods (Random Forest, Adaboost, bagging, Multi-layer perceptrons, Support Vector Machine (SVM) and Genetic Programming). Nair et al. [10] used decision tree in predicting software defect and compared the performance with three other intelligence techniques namely Logistic Regression, Multi-layer Perceptrons and SVM; Reshi and Singh [21] used smell prediction model based on SVM to predict defects in releases of eclipse software; Ranjan et al. [4] explored the different techniques used in software fault prediction such as soft computing, data mining and machine learning approaches as well as their accuracy rates using Random Forest, Decision Tree Regression, Neural Network, Genetic Algorithm, SVM, Artificial Neural Network and Fuzzy Logic; Fan et al. [6] proposed a framework called Defect Prediction via Attention-based Recurrent Neural Network (DP-ARNN) which automatically learned syntactic and semantic features for accurate defect prediction. Seven Java projects in Apache were chosen to validate the framework using F1 measure and Area Under Curve (AUC) as evaluation criteria.

Some researchers proposed the use of alternative modeling and ensemble techniques to predict software fault. For example; Jin et al. [22] used Artificial Neural Networks (ANNs) and SVMs to predict software fault-proneness. They used ANNs to discard poor attributes and then used the SVM to predict fault-proneness using the selected features. Kumar and Rath [23] applied genetic algorithm with neural network on CK metrics in prediction software fault in early stages of software development. Singh et al. [3] developed an integrated system that can find useful software metrics (features) and simultaneously generate a set of human interpretable fuzzy rules for software fault prediction. Erturk and Sezer [24] applied fuzzy logic and artificial neural network to predict software fault. Arasteh, B. [12] combined neural network and naïve bayes algorithm to build software fault prediction model. Five traditional fault datasets were used to construct and evaluate the prediction model. All these authors used different performance measures (precision, recall, receiver operating characteristic (ROC), Matthews correlation coefficient (MCC), Accuracy, Balance, Geometric Mean, Root Mean Squared Error (RMSE), ROC-AUC) indicating that defection prediction process lacks a standard method of evaluating the proposed models [7]. The opinion that their proposed models predict software fault accurately may not be false, but the results regarding the superiority of one technique over another are difficult to assess across different studies due to different experimental setups [25], use of different datasets, performance and accuracy measures, different evaluation criteria [6], set of attributes used as input to design the predictor as well as the modeling tool used [3].

Some researchers have used fuzzy logic for the prediction of software faults due to the presence of vagueness associated with software data. Those that applied type-1 include Pandey and Goyal [26], Erturk and Sezer [27], Chatterjee and Maji [28], Kakkar et al. [13], Jaikumar and Ramani [29], Adak [7] while Chatterjee et al., [30] utilized type-2 fuzzy set. These authors opined that fuzzy reasoning is similar to human reasoning and also provides effective way to deal with vagueness and uncertainty associated with measurement of numeric values which makes fuzzy logic very easy to comprehend. Despite the extensive use of type-1 fuzzy systems, it has been established from previous work that type-1 fuzzy logic only handles uncertainty to a certain degree in many applications and may not minimize their effects in some real world applications [31]. Type-2 fuzzy systems handle uncertainties that type-1 cannot handle because their membership grades are fuzzy and can be used to reduce uncertainties in the vague linguistic values of software attributes [30]. Although a great number of studies have been conducted in the area of software fault prediction, unfortunately the accuracies of these models are not quite satisfactory up till date, so more efforts need to be exerted

DOI: 10.31695/IJASRE.2021.1.34014

especially as new modeling algorithms are periodically being developed taking cognizance of the shortcomings of the previous studies [32]. As far as software engineering is concerned, approaches to software defect prediction are still emerging as a potential field of research [11].

In this work therefore, an interval type-2 intuitionistic fuzzy logic system (IT2IFLS) is proposed for the first time to predict software fault at the requirement phase of SDLC. Intuitionistic fuzzy set (IFS) [33] proposed by Atanassov deals with uncertainty using degrees of membership function (MF) and non-membership function (NMF) of an element to a fuzzy set as well as hesitation index, hence it becomes the appropriate tool to deal with imprecise and vague information [34]. However, because the IFS of type-1 is designed around a single point of view and may not capture various forms of uncertainties, Eyoh et al. [31] presented a rule-based interval type-2 intuitionistic fuzzy logic system (IT2IFLS) that are defined using fuzzy MFs and NMFs together with hesitation indexes. The IT2IFLS has successfully been applied to solve many applications in different domains with impressive performance such as load forecasting [35], time series prediction [36], [37], [38], [39], [40], [41], identification and prediction problems [31], [42], [43], regression problems [44], [45] transportation problems [46] and many more. To the best knowledge of the authors, this is the first study that adopts IT2IFLS with MF, NMF and hesitation indices for software fault prediction.

The intention is to evaluate the performance of IT2IFLS with the existence of NMF and hesitation indices with those of IT2FLS and other traditional fuzzy systems. The rest of the paper is arranged as follow:

Section 2 has the preliminaries where basic terms are defined. The methodology is presented in Section 3 while parameter update is presented in Section 4. In section 5, experimental set-up is described and Section 6 is the performance evaluation with discussion on experimental findings and conclusion is drawn in Section 7.

2. PRELIMINARIES

***DEFINITION 1*:** An intuitionistic fuzzy set (IFS), A* is characterized by MF, $\mu_{G^*}(x)$ and NMF, $\nu_{G^*}(x)$ and

satisfies the condition $\mu_{G^*}(x) + \nu_{G^*}(x) \leq 1$ [33]. There exists also a hesitation index, π , also known as the intuitionistic fuzzy index such that: $\pi_{G^*}(x) = 1 - (\mu_{G^*}(x) + \nu_{G^*}(x))$. Obviously, when $\mu_{G^*}(x) + \nu_{G^*}(x) = 1$, a classical FS is recovered.

***DEFINITION 2*:** An IT2IFS, $\tilde{G}^*$ is characterized by fuzzy MFs and fuzzy NMFs and defined as:

$\tilde{G}^* = (\underline{\mu}_{\tilde{G}^*}(x), \overline{\mu}_{\tilde{G}^*}(x), \underline{\nu}_{\tilde{G}^*}(x), \overline{\nu}_{\tilde{G}^*}(x))$ such that $0 \leq \overline{\mu}_{\tilde{G}^*}(x) + \underline{\nu}_{\tilde{G}^*}(x) \leq 1$ and $0 \leq \underline{\mu}_{\tilde{G}^*}(x) + \overline{\nu}_{\tilde{G}^*}(x) \leq 1$ [47].

In this research, the IT2IFS is defined using intuitionistic type-2 Gaussian function with uncertain standard deviation (see Figure 1). Thus, for IT2IFSs, two footprints of uncertainties (FOUs) are generated namely MF and NMF FOUs defined as in (1) and (2). Any system that adopts IT2IFS in either the antecedent or consequent part is called interval type-2 intuitionistic fuzzy logic system (see Figure 2).

$$FOU_{\mu}(\tilde{G}^*) = \bigcup_{\forall x \in X}[\underline{\mu}_{\tilde{G}^*}(x), \overline{\mu}_{\tilde{G}^*}(x)] \quad (1)$$

$$FOU_{\nu}(\tilde{G}^*) = \bigcup_{\forall x \in X}[\underline{\nu}_{\tilde{G}^*}(x), \overline{\nu}_{\tilde{G}^*}(x)] \quad (2)$$

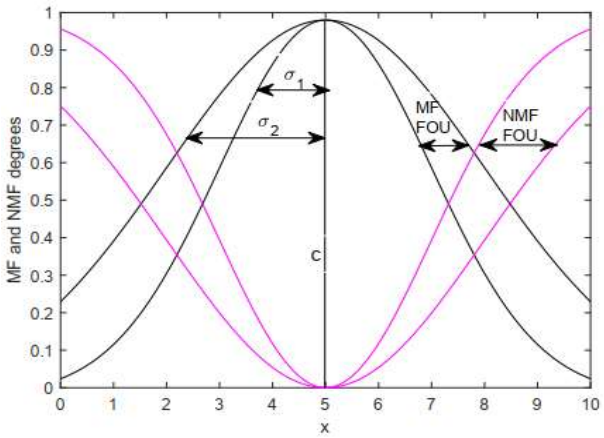

Figure 1: Interval type-2 intuitionistic fuzzy set

DOI: 10.31695/IJASRE.2021.1.34014

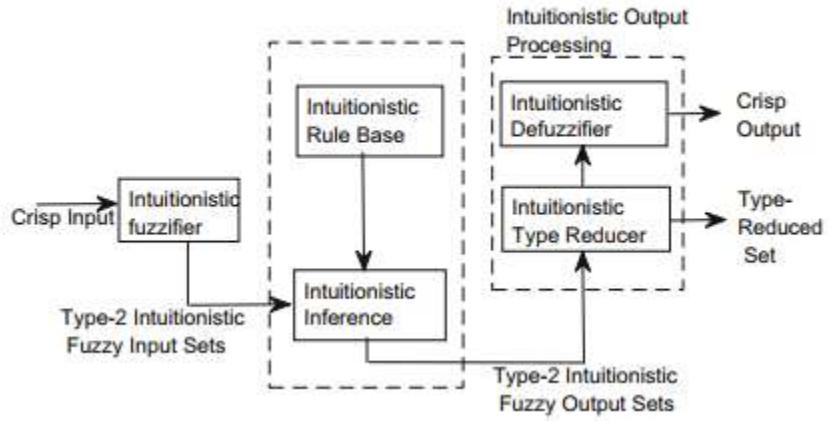

Figure 2. Architecture of the proposed IT2IFLS for software fault prediction

3. METHODOLOGY

The methodology employed in this study is the interval type-2 intuitionistic fuzzy logic system trained with gradient descent algorithm. The IT2IFLS comprises the intuitionistic - fuzzifier, rule base, inference engine and an output processing module (type-reducer and defuzzifier).

3.1. Fuzzification

First, the crisp inputs are propagated through the IT2IFLS to obtain the MFs (upper and lower) and NMFs (upper and lower) as represented in (3) - (6). In this work, the MF and NMF are adapted from type-1 IFS [48] definition and are defined as in [31]:

$$\bar{\mu}_{ik}(x_i) = exp\left(-\frac{(x_i - c_{ik})^2}{2\underline{\sigma}_{2,ik}^2}\right) * (1 - \pi_{c,ik}(x_i)) \quad (3)$$

$$\underline{\mu}_{ik}(x_i) = exp\left(-\frac{(x_i - c_{ik})^2}{2\underline{\sigma}_{1,ik}^2}\right) * (1 - \pi_{c,ik}(x_i)) \quad (4)$$

$$\underline{v}_{ik}(x_i) = (1 - \underline{\pi}_{var,ik}(x_i)) - \bar{\mu}_{ik}(x_i) \quad (5)$$

$$\bar{v}_{ik}((x_i) = (1 - \bar{\pi}_{var,ik}(x_i)) - \underline{\mu}_{ik}(x_i) \quad (6)$$

where π_c is the hesitation degree for the center and π_{var} is the hesitation degree for the variance. The parameters σ_2, σ_1, π_c, and π_{var} constitute antecedent parameters for MFs and NMFs of IT2IFS.

3.2. Rules

The inputs are further combined to form rules in the rule base which activates the inference engine. The inference system adopted in this research is the Takagi-Sugeno-Kang (TSK) inference. Equation (7) is an example of IT2IFLS-TSK rule formation:

$$R_k : IF\ x_1\ is\ \tilde{G}_{1k}^*\ and\ x_2\ is\ \tilde{G}_{2k}^*\ and \cdots and\ x_n\ is\ \tilde{G}_{nk}^*\ THEN\ p_k = w_{1k}x_1 + w_{2k}x_2 + \cdots + w_{nk}x_n + b_k \quad (7)$$

where $\tilde{G}_{ik}^*$ are IT2IFSs (i = 1. 2, 3, …, n) and p_k (k = 1, 2, …, K) are the outputs of the k_{th} rule.

3.3. Inference

In this study, a TSK-based inference called A2-C0 is employed with IT2IFSs as inputs in the antecedent (A2) and a linear combination of the input as output (C0).

3.4. Output processing

Existing works in the literature adopt Karnik-Mendel iterative type-reduction procedure to reduce a T2FS to T1FS. Literature shows that type reduction procedure is very tedious and computationally expensive [49]. Alternative methods have been developed to by-pass the type-reduction procedure and directly compute the output of a T2FLS. These alternative methods include

DOI: 10.31695/IJASRE.2021.1.34014

Begian-Melek-Mendel (BMM) [50], Nie-Tan [51], Wu and Tan [52] to mention but a few. This work benefits from BMM IT2FLS procedure and directly computes the outputs of IT2IFLS as reported in [45] and shown in (8):

$$p = \frac{(1-\beta)\sum_{k=1}^{M}\left(\underline{f_k^{\mu}} + \overline{f_k^{\mu}}\right)p_k^{\mu}}{\sum_{k=1}^{M}\underline{f_k^{\mu}} + \sum_{k=1}^{M}\overline{f_k^{\mu}}} + \frac{\beta\sum_{k=1}^{M}\left(\underline{f_k^{v}} + \overline{f_k^{v}}\right)p_k^{v}}{\sum_{k=1}^{M}\underline{f_k^{v}} + \sum_{k=1}^{M}\overline{f_k^{v}}} \tag{8}$$

where:

$$\underline{f_k^{\mu}} = \underline{\mu}_{\tilde{G}_{1k}^{*}}(x_1) * \underline{\mu}_{\tilde{G}_{2k}^{*}}(x_2) * \ldots * \underline{\mu}_{\tilde{G}_{nk}^{*}}(x_n) \tag{9}$$

$$\overline{f_k^{\mu}} = \overline{\mu}_{\tilde{G}_{1k}^{*}}(x_1) * \overline{\mu}_{\tilde{G}_{2k}^{*}}(x_2) * \ldots * \overline{\mu}_{\tilde{G}_{nk}^{*}}(x_n) \tag{10}$$

$$\overline{f_k^{v}} = \overline{v}_{\tilde{G}_{1k}^{*}}(x_1) * \overline{v}_{\tilde{G}_{2k}^{*}}(x_2) * \ldots * \overline{v}_{\tilde{G}_{nk}^{*}}(x_n) \tag{11}$$

$$\underline{f_k^{v}} = \underline{v}_{\tilde{G}_{1k}^{*}}(x_1) * \underline{v}_{\tilde{G}_{2k}^{*}}(x_2) * \ldots * \underline{v}_{\tilde{G}_{nk}^{*}}(x_n) \tag{12}$$

The ($\underline{f_k^{\mu}}$, $\overline{f_k^{\mu}}$) and ($\underline{f_k^{v}}$ and $\overline{f_k^{v}}$) represent the firing strengths for MF (lower and upper) and NMF (lower and upper) respectively. The parameter β determines the influence of MF and NMF to the final output and lies between 0 and 1. If $\beta = 0$, the MF alone contributes to the final output and if $\beta = 1$, NMF only contributes to the final output. If $0 \leq \beta \leq 1$, both MF and NMF contribute to the final output.

4. PARAMETER UPDATE RULE

This work adopts gradient descent backpropagation for updating the parameters of the IT2IFLS with the aim of minimizing the cost function which is defined as in (13) for a single output.

$$E = \frac{1}{2}(p^{a} - p)^2 \tag{13}$$

where p^{a} represents the actual output and p is the predicted output of IT2IFLS. The update rules based on GD are represented as in (14) to (18):

$$w_{ik}(t+1) = w_{ik}(t) - \gamma\frac{\partial E}{\partial w_{ik}} \tag{14}$$

$$b_k(t+1) = b_k(t) - \gamma\frac{\partial E}{\partial b_k} \tag{15}$$

$$c_{ik}(t+1) = c_{ik}(t) - \gamma\frac{\partial E}{\partial c_{ik}} \tag{16}$$

$$\sigma_{1,ik}(t+1) = \sigma_{1,ik}(t) - \gamma\frac{\partial E}{\partial \sigma_{1,ik}} \tag{17}$$

$$\sigma_{2,ik}(t+1) = \sigma_{2,ik}(t) - \gamma\frac{\partial E}{\partial \sigma_{2,ik}} \tag{18}$$

where γ is the learning rate, w; b are the parameters of the consequent parts and c; $\sigma 1$; $\sigma 2$ are parameters of the antecedent parts. The adjustment of the consequent parameters of IT2IFLS are as follows:

$$\frac{\partial E}{\partial w_{ik}} = \frac{\partial E}{\partial p}\frac{\partial p}{\partial p_k}\frac{\partial p_k}{\partial w_{ik}} = \sum_{k=1}^{M}\frac{\partial E}{\partial p}\left[\frac{\partial p}{\partial p_k^{\mu}}\frac{\partial p_k^{\mu}}{\partial w_{ik}} + \frac{\partial p}{\partial p_k^{v}}\frac{\partial p_k^{v}}{\partial w_{ik}}\right]$$

DOI: 10.31695/IJASRE.2021.1.34014

$$= (p^a(t) - p(t)) * \left[(1 - \beta) \left(\frac{\underline{f}_k^{\mu}}{\sum_{k=1}^{M} \underline{f}_k^{\mu} + \sum_{k=1}^{M} \bar{f}_k^{\mu}} + \frac{\bar{f}_k^{\mu}}{\sum_{k=1}^{M} \underline{f}_k^{\mu} + \sum_{k=1}^{M} \bar{f}_k^{\mu}} \right) + \beta \left(\frac{\underline{f}_k^{v}}{\sum_{k=1}^{M} \underline{f}_k^{v} + \sum_{k=1}^{M} \bar{f}_k^{v}} + \frac{\bar{f}_k^{v}}{\sum_{k=1}^{M} \underline{f}_k^{v} + \sum_{k=1}^{M} \bar{f}_k^{v}} \right) \right] * x_i$$

(19)

$$\frac{\partial E}{\partial b_k} = \frac{\partial E}{\partial p} \frac{\partial p}{\partial p_k} \frac{\partial p_k}{\partial b_k} = \sum_{k=1}^{M} \frac{\partial E}{\partial p} \left[\frac{\partial p}{\partial p_k^{\mu}} \frac{\partial p_k^{\mu}}{\partial b_k} + \frac{\partial p}{\partial p_k^{v}} \frac{\partial p_k^{v}}{\partial b_k} \right]$$

$$= (p^a(t) - p(t)) * \left[(1 - \beta) \left(\frac{\underline{f}_k^{\mu}}{\sum_{k=1}^{M} \underline{f}_k^{\mu} + \sum_{k=1}^{M} \bar{f}_k^{\mu}} + \frac{\bar{f}_k^{\mu}}{\sum_{k=1}^{M} \underline{f}_k^{\mu} + \sum_{k=1}^{M} \bar{f}_k^{\mu}} \right) + \beta \left(\frac{\underline{f}_k^{v}}{\sum_{k=1}^{M} \underline{f}_k^{v} + \sum_{k=1}^{M} \bar{f}_k^{v}} + \frac{\bar{f}_k^{v}}{\sum_{k=1}^{M} \underline{f}_k^{v} + \sum_{k=1}^{M} \bar{f}_k^{v}} \right) \right] * 1$$

(20)

where p_k is the output of the k_{th} rule. The updates for the center and standard deviations of the IT2IFLS are computed as follows:

$$\frac{\partial E}{\partial c_{ik}} = \sum_{k=1}^{M} \frac{\partial E}{\partial p} \left[\frac{\partial p}{\partial \underline{f}_k^{\mu}} \frac{\partial \underline{f}_k^{\mu}}{\partial \underline{\mu}_{ik}} \frac{\partial \underline{\mu}_{ik}}{\partial c_{ik}} + \frac{\partial p}{\partial \bar{f}_k^{\mu}} \frac{\partial \bar{f}_k^{\mu}}{\partial \bar{\mu}_{ik}} \frac{\partial \bar{\mu}_{ik}}{\partial c_{ik}} + \frac{\partial p}{\partial \underline{f}_k^{v}} \frac{\partial \underline{f}_k^{v}}{\partial \underline{v}_{ik}} \frac{\partial \underline{v}_{ik}}{\partial c_{ik}} + \frac{\partial p}{\partial \bar{f}_k^{v}} \frac{\partial \bar{f}_k^{v}}{\partial \bar{v}_{ik}} \frac{\partial \bar{v}_{ik}}{\partial c_{ik}} \right] \quad (21)$$

$$\frac{\partial E}{\partial \sigma_{1,ik}} = \sum_{k=1}^{M} \frac{\partial E}{\partial p} \left[\frac{\partial p}{\partial \underline{f}_k^{\mu}} \frac{\partial \underline{f}_k^{\mu}}{\partial \underline{\mu}_{ik}} \frac{\partial \underline{\mu}_{ik}}{\partial \sigma_{1,ik}} + \frac{\partial p}{\partial \bar{f}_k^{v}} \frac{\partial \bar{f}_k^{v}}{\partial \bar{v}_{ik}} \frac{\partial \bar{v}_{ik}}{\partial \sigma_{1,ik}} \right] \quad (22)$$

$$\frac{\partial E}{\partial \sigma_{2,ik}} = \sum_{k=1}^{M} \frac{\partial E}{\partial p} \left[\frac{\partial p}{\partial \bar{f}_k^{\mu}} \frac{\partial \bar{f}_k^{\mu}}{\partial \bar{\mu}_{ik}} \frac{\partial \bar{\mu}_{ik}}{\partial \sigma_{2,ik}} + \frac{\partial p}{\partial \underline{f}_k^{v}} \frac{\partial \underline{f}_k^{v}}{\partial \underline{v}_{ik}} \frac{\partial \underline{v}_{ik}}{\partial \sigma_{2,ik}} \right] \quad (23)$$

where:

$$\frac{\partial p}{\partial \underline{f}_k^{\mu}} = \frac{\partial p}{\partial \bar{f}_k^{\mu}} = (1 - \beta) \left[\frac{p_k^{\mu}}{\sum_{k=1}^{M} \underline{f}_k^{\mu} + \sum_{k=1}^{M} \bar{f}_k^{\mu}} - \frac{p^{\mu}}{\sum_{k=1}^{M} \underline{f}_k^{\mu} + \sum_{k=1}^{M} \bar{f}_k^{\mu}} \right] \quad (24)$$

where:

$$p^{\mu} = \frac{\sum_{k=1}^{M} \left(\underline{f}_k^{\mu} + \bar{f}_k^{\mu} \right) * p_k^{\mu}}{\sum_{k=1}^{M} \underline{f}_k^{\mu} + \sum_{k=1}^{M} \bar{f}_k^{\mu}} \quad (25)$$

$$\frac{\partial p}{\partial \underline{f}_k^{v}} = \frac{\partial p}{\partial \bar{f}_k^{v}} = \beta \left[\frac{p_k^{v}}{\sum_{k=1}^{M} \underline{f}_k^{v} + \sum_{k=1}^{M} \bar{f}_k^{v}} - \frac{p^{v}}{\sum_{k=1}^{M} \underline{f}_k^{v} + \sum_{k=1}^{M} \bar{f}_k^{v}} \right] \quad (26)$$

where:

$$p^{v} = \frac{\sum_{k=1}^{M} \left(\underline{f}_k^{v} + \bar{f}_k^{v} \right) * p_k^{v}}{\sum_{k=1}^{M} \underline{f}_k^{v} + \sum_{k=1}^{M} \bar{f}_k^{v}} \quad (27)$$

$$\frac{\partial \underline{\mu}_{ik}(x_i)}{\partial c_{ik}} = \left(1 - \pi_{c,ik}\right) * \left(x_i - c_{ik}\right) * \exp\left(- \frac{(x_i - c_{ik})^2}{2\sigma_{1,ik}^2} \right) / \sigma_{1,ik}^2 \quad (28)$$

$$\frac{\partial \bar{\mu}_{ik}(x_i)}{\partial c_{ik}} = \left(1 - \pi_{c,ik}\right) * \left(x_i - c_{ik}\right) * \exp\left(- \frac{(x_i - c_{ik})^2}{2\sigma_{2,ik}^2} \right) / \sigma_{2,ik}^2 \quad (29)$$

DOI: 10.31695/IJASRE.2021.1.34014

$$\frac{\partial \bar{\mu}_{ik}(x_i)}{\partial \sigma_{1,ik}} = (1 - \pi_{c,ik}) * (x_i - c_{ik})^2 * \exp\left(-\frac{(x_i - c_{ik})^2}{2\sigma_{1,ik}^2}\right) / \sigma_{1,ik}^3 \quad (30)$$

$$\frac{\partial \bar{\mu}_{ik}(x_i)}{\partial \sigma_{2,ik}} = (1 - \pi_{c,ik}) * (x_i - c_{ik})^2 * \exp\left(-\frac{(x_i - c_{ik})^2}{2\sigma_{2,ik}^2}\right) / \sigma_{2,ik}^3 \quad (31)$$

The derivatives for the NMFs are $\frac{\partial \underline{v}_{ik}}{\partial c_{ik}} = -\frac{\partial \underline{\mu}_{ik}(x_i)}{\partial c_{ik}}$, $\frac{\partial \bar{v}_{ik}}{\partial c_{ik}} = -\frac{\partial \bar{\mu}_{ik}(x_i)}{\partial c_{ik}}$, $\frac{\partial \bar{v}_{ik}}{\partial \sigma_{1,ik}} = -\frac{\partial \bar{\mu}_{ik}(x_i)}{\partial \sigma_{1,ik}}$ and $\frac{\partial \underline{v}_{ik}}{\partial \sigma_{2,ik}} = -\frac{\partial \bar{\mu}_{ik}(x_i)}{\partial \sigma_{2,ik}}$. Then,

$\frac{\partial \underline{f}_k^{\mu}}{\partial \underline{\mu}_{ik}} = \prod_{j=1, j\neq i}^{M1} \underline{\mu}_{jk}$, $\frac{\partial \bar{f}_k^{\mu}}{\partial \bar{\mu}_{ik}} = \prod_{j=1, j\neq i}^{M1} \bar{\mu}_{jk}$, $\frac{\partial \underline{f}_k^{v}}{\partial \underline{v}_{ik}} = \prod_{j=1, j\neq i}^{M1} \underline{v}_{jk}$ and $\frac{\partial \bar{f}_k^{v}}{\partial \bar{v}_{ik}} = \prod_{j=1, j\neq i}^{M1} \bar{v}_{jk}$,

The parameter, β, is adjusted as follows:

$$\frac{\partial E}{\partial \beta} = \frac{\partial E}{\partial p}\frac{\partial p}{\partial \beta} = (p^a - p)(p^v - p^{\mu}) \quad (32)$$

5. EXPERIMENTAL SETUP

The dataset used to implement our proposed intuitionistic model is gotten from the work of Fenton et al. [53], a highly referenced dataset in the literature as depicted in Table 1 and contains the size of different projects measured in thousands (K) of lines of code (KLOC) and the number of defects in each of the twenty three software projects as gathered from the electronics consumers industry. The inputs to the system include - requirement phase metrics (Requirement Complexity (RC), Requirement Stability (RS) and Experience of Requirement Team (ERT)).

Table 1 – Software projects and their number of defects

S/N	Software Project	Software Size (KLOC)	Number of Defects
1	1	6.0	148
2	2	31.0	31
3	3	53.9	209
4	5	14.0	373
5	7	21.0	204
6	8	5.8	53
7	9	2.5	17
8	10	4.8	29
9	11	4.4	71
10	12	19.0	90
11	13	49.1	129
12	14	58.0	672
13	15	154.0	1768
14	16	27.7	109
15	17	33.0	688
16	19	87.0	476
17	20	50.0	928

DOI: 10.31695/IJASRE.2021.1.34014

18	21	22.0	196
19	22	44.0	184
20	23	61.0	680
21	24	99.0	1597
22	27	52.0	412
23	29	11.0	91

These requirements metrics are as listed in [54] and include the following:

(i) Requirement Complexity (RC) - High RC means increase in the number of faults which leads to low reliability of software.

(ii) Requirement Stability (RS) - This metric quantifies the stability of requirement specification in the requirement phase of SDLC. More RS increases the number of faults which causes the reliability to be low

(iii) Experience of Requirement Team (ERT) - This metric determines the relevant and skill set of team staff involved in project execution at the requirement stage of SDLC.

However, to aid comparison with the results of IT2FLS in [55], the KLOC is also adopted as one of the inputs into the IT2IFLS. To design the system, three MFs and NMFs are utilized. The experiment is run for 10 times with 100 epochs each and average error computed. The hesitation index is set at 0.01 with the user parameter initially fixed at 0.5. The experiment is run on MATLAB© 2020a.

6. PERFORMANCE EVALUATION

To aid comparison with previous works, the same performance metrics used in Chartagee et al. [30] and Umoeka et al. [55] are also adopted. These include the root mean squared error (RMSE), the normalized root mean squared error (NRMSE), the mean magnitude of relative error (MMRE), the balanced mean magnitude of relative error (BMMRE) and the coefficient of determination (R^2).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i^a - y_i)^2} \tag{33}$$

$$NRMSE = \frac{1}{n}\sum_{i=1}^{n}(y_i^a - y_i)^2 / \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{y_i} * \frac{1}{n}\sum_{i=1}^{n}y_i^a \tag{34}$$

$$MMRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i^a - y_i|}{y_i} \tag{35}$$

$$BMMRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i^a - y_i|}{Min(y_i^a, y_i)} \tag{36}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i^a - y_i)^2}{\sum_{i=1}^{n}(y_i^a - \bar{y})^2} \tag{37}$$

where $\bar{y}$ is the mean of the actual values, y_i^a.

The IT2IFLS inputs consist of RC, RS and ERT with KLOC. For efficient learning, the KLOC is normalized to lie in closed interval of 0 and 1. After prediction, the final results are de-normalized to obtain the predicted values in their original forms. The predicted results using IT2IFLS is compared with existing works in the literature as shown in Table 2.

DOI: 10.31695/IJASRE.2021.1.34014

Table 2: Comparison of IT2IFLS outputs with other prediction models

SN	Project No [53]	Actual Values	Predicted values				
			[53]	[54]	[30]	[55]	Proposed model
1	1	148	75	55.94	85	146.6	148
2	2	31	52	5.48	37	32.8	30.8
3	3	209	254	210.61	-	228.6	212.1
4	5	373	349	-	-	365.1	373
5	7	204	262	113.43	139	206.5	204.2
6	8	53	48	53.81	-	53.0	49.7
7	9	17	57	52	41	23.4	17
8	10	29	209	26.17	64	155.1	29
9	11	71	51	40.61	-	69.2	71
10	12	90	347	176.25	219	91.4	90.1
11	13	129	516	336.59	-	129.7	129.1
12	14	672	674	697.48	-	680.2	671.9
13	15	1768	1526	1650.89	1946	1768.1	1767.8
14	16	109	145	127.94	-	100.0	109.6
15	17	688	444	135.74	371	573.6	688
16	19	476	581	573.44	-	470.0	475.9
17	20	928	986	869.23	-	927.2	927.9
18	21	196	259	105.51	-	195.4	196
19	22	184	501	291.02	-	191.1	184
20	23	680	722	690.17	-	682.2	680
21	24	1597	1514	-	-	1597.8	1597
22	27	412	430	400.17	-	392.2	412
23	29	91	116	110.06	82	91.4	91

DOI: 10.31695/IJASRE.2021.1.34014

Shown in Table 2 is the actual and predicted outputs of IT2IFLS with other existing models in the literature. From Table 2, IT2IFLS with MFs and NMFs predicted outputs are closer to the actual outputs compared to previous works in the literature. It is also demonstrated based on the prediction results that the fuzzy logic models namely: IT2FLS in [55] and the proposed IT2IFLS with optimized parameters do outperform the IT2FLS in [30] where the parameters of the model are not tuned.

Table 3: Performance comparison of the proposed model

Models	***RMSE***	***NRMSE***	***MMRE***	***BMMRE***	R^2
Chatterjee et al., [30]	132.8230	0.0758	0.6280	0.7247	0.9398
Pandey and Goyal [52]	195.7192	0.1118	0.6744	1.5648	0.8692
Fenton et al., [51]	158.3891	0.0904	1.4922	1.5696	0.9144
Umoeka et al [53]	102.65	0.0543	0.1204	0.2484	0.9402
Proposed model	**59.25**	**0.0211**	**0.0376**	**0.1012**	**0.9948**

Table 3 shows the accuracy of the different models based on the performance metrics. As shown in Table 3, IT2IFLS provides the lowest prediction error in terms of RMSE, NRMSE, MMRE and BMMRE. The prediction accuracy of IT2IFLS is very significant with coefficient of determination (R^2) of 99.48. Figure 3 shows the actual and predicted outputs of software faults using the classical IT2FLS [55] and the proposed IT2IFLS. As shown in Figure 3, the predicted outputs of IT2IFLS are in close agreement with the actual outputs than the classical IT2FLS. Figure 4 is the convergence graph of IT2IFLS showing a smooth convergence. Shown in Figures 5 and 6 are the graphical views of the different prediction errors with IT2IFLS having the least errors.

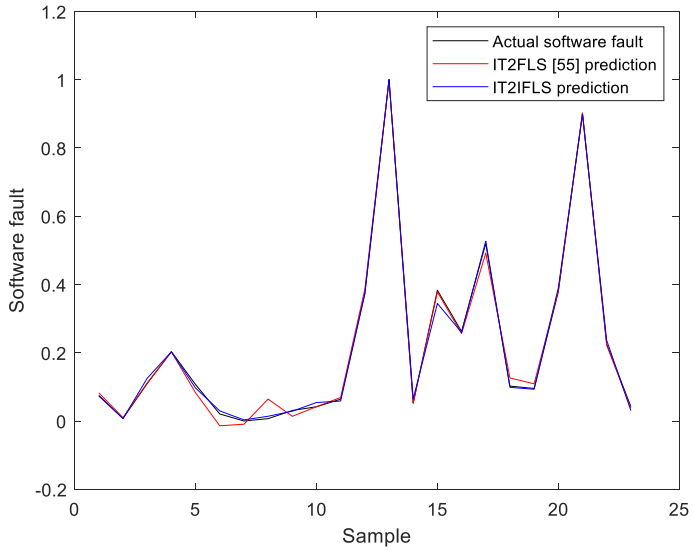

Figure 3: Comparison of classical IT2FLS [55] with IT2IFLS prediction

DOI: 10.31695/IJASRE.2021.1.34014

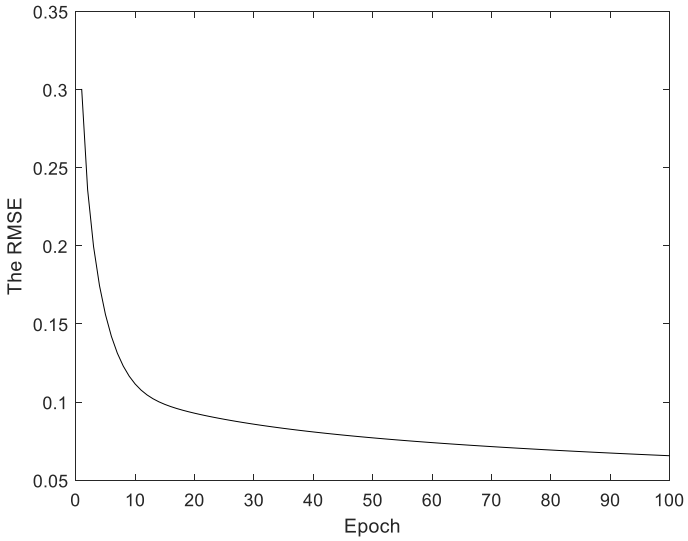

Figure 4: Convergence graph of IT2IFLS

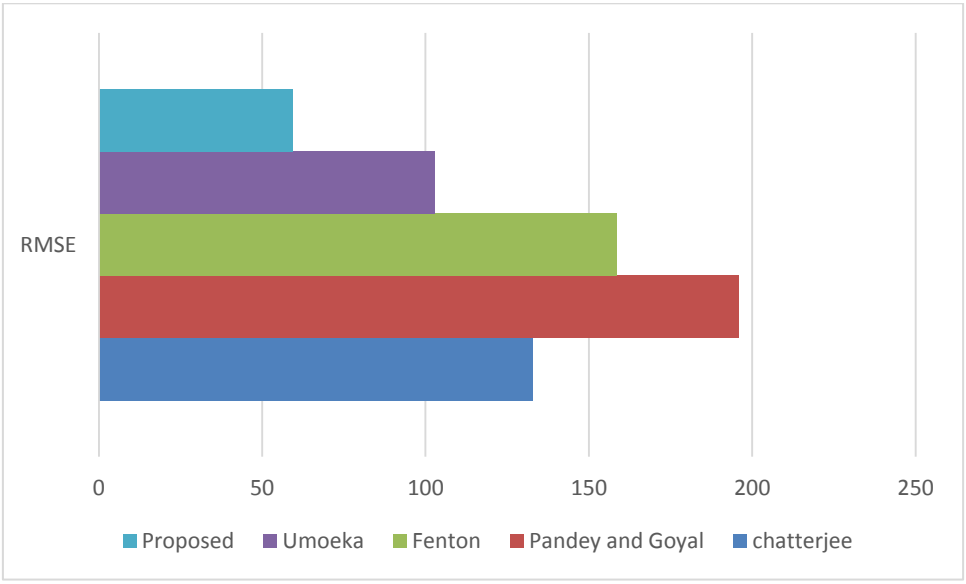

Figure 5: RMSE of Software fault prediction

DOI: 10.31695/IJASRE.2021.1.34014

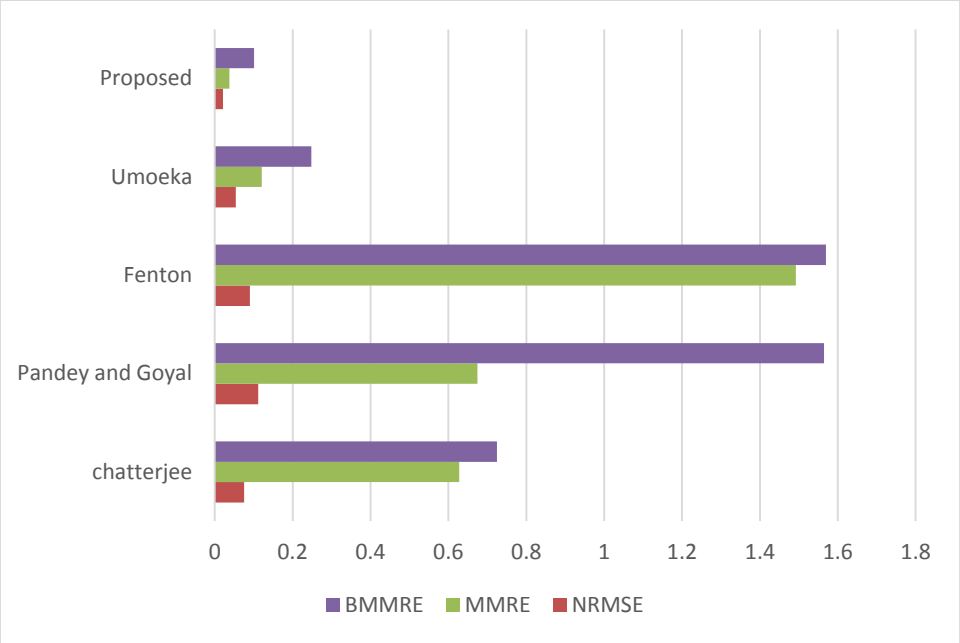

Figure 6: Software fault prediction errors (BMMRE, MMRE and NRMSE)

7. CONCLUSION

This work presents a novel application of IT2IFLS for the prediction of software fault in the requirement phase of Software Development Life Cycle. IT2IFLS unlike ordinary IT2FLS is defined using separate MF and NMF degrees with some form of hesitation in the fuzzy set definition. The main advantage of using IT2IFLS in evaluating software faults is that the proposed method can express indifference in expert's opinion in the form of hesitation which makes evaluation closer to human reasoning as possible. As demonstrated through experimental analysis, the prediction using IT2IFLS follows closely with the actual software fault outputs than IT2FLS, an indication of good performance. As shown in Table 3, IT2IFLS provides least errors in terms of RMSE, NRMSE, MMRE and BMMRE compared to existing works in the literature.

In the future, we intend to explore other prediction approaches such as extreme learning machine, support vector machine, generalized interval type-2 intuitionistic fuzzy logic and hybrid methods in the software fault prediction.

REFERENCES

[1] Ali, A., Jawawi, D., Isa M., Babar, M., "Technique for Early Reliability Prediction of Software Components Using Behaviour Models". PLoS ONE 11(9): e0163346, 2016, doi:10.1371/journal.pone.0163346.

[2] Kaur, R. and Sharma, E., "Various Techniques to Detect and Predict Faults in Software System: Survey". International Journal on Future Revolution in Computer Science and Communication Engineering, 2018, 4(2), 330 – 336.

[3] Singh, P., Pal, N., Verma, S. and Vyas, O., "Fuzzy Rule-Based Approach for Software Fault Prediction". In proceedings of IEEE Transactions on Systems, MAN and Cybernetics: Systems, 2016, 47(5), 826 – 837.

[4] Ranjan, P., Kumar, S. and Kumar, U., "Software Fault Prediction Using Computational Intelligence Techniques: A Survey". Indian Journal of Science and Technology, 2017, 10(8), 1 – 9.

[5] Auju, A. J. and Judith, J. E., "Software Defect Prediction using Efficient Classification Algorithm". International Journal of Recent Technology and Engineering, 2019, 8(3), 301 – 304.

[6] Fan, G., Diao, X., Yu, H., Yang, K. and Chen, L., "Software Defect Prediction via Attention-Based Recurrent Neural Network". Hindawi Scientific Programming, 2019.

[7] Adak, M., Software Defect Prediction using Data Mining Fuzzy Logic. In Proceeding of 6th International Conference on Digital Information, Networking and Wireless Communication, Beirut, Lebanon, 2018, 65 – 69.

[8] Jayanthi, R., Lilly, F. and Arya, Arti. "A review of Software Defect Prediction Techniques using Product Metrics". International Journal of Database Theory and Applications, 2017, 10(1), 163 – 174.

[9] Malhotra, R and Jain, A., "Fault Prediction Using Statistical and Machine Learning Methods for Improving Software Quality". Journal of Information Processing Systems, 2012, 8(2), 241-262.

DOI: 10.31695/IJASRE.2021.1.34014

[10] Nair, A., Arya, A., Shrivastava, A. and Shrivastava, V., "Software Fault Prediction using Intelligence Techniques". International Journal of Advanced Research in Computer Science, 2013, 4(11), 166 – 169.

[11] Moeyersoms, J., deFortumy, E., Dejaeger, K. and Beasens, B., "Comprehensible Software Fault and Effort Prediction: A Data Mining Approach". Journal of Systems and Software, 2015, 100, 80 – 90.

[12] Arasteh, B., "Software Fault Prediction using Combination of Neural Network and naïve Bayes Algorithm". Journal of Networking Technology, 2018, 9(3), 94 -101.

[13] Kakkar, M., Jain, S., Bansal, A. and Grover, P., "Fuzzy Logic-Based Model to Predict Per phase Software Defect". International Journal of Innovative Technology and Exploring Engineering, 2019, 8(9), 36 – 41.

[14] Rathore. S. and Kumar, S., "A Study on Software Fault Prediction Techniques". Artificial Intelligence Review, 2017, 51, 255 – 327.

[15] Catal, C. and Diri, B., "Investigating the Effect of Dataset Size, Metrics Sets and Feature Selection Techniques on Software Fault Prediction Problem". Elsevier, 2009, pp. 1040-1058.

[16] Basili, V., Briand, L and Melo, W., "A validation of Object-Oriented Design Metrics as Quality Indicators". IEEE Transactions on Software Engineering, 22(10), 1996, pp. 751–761.

[17] Khoshgoftaar, T., Allen, E., Hudepohl, J. and Aud, S., "Application of Neural Networks to Software Quality Modeling of a very large Telecommunications System", IEEE Transactions on Neural Networks, 1997, 8(4), 902–909.

[18] Khoshgoftaar, T., Allen, E., and Deng, J., "Using Regression Trees to Classify Fault-Prone Software Modules". IEEE Transactions on Reliability 51 (4), 2002, 455–462.

[19] Parvinder, S., Sunil, K., Satpreet, S., Simranjit, K., Manpreet, K. and S. Gurvinder, S., "A Study on Early Prediction of Fault Proneness in Software Modules using Genetic Algorithm". World Academy of Science, Engineering and Technology, International Journal of Computer and Information Engineering, 2010, 4(12), 1891 – 1896.

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

[21] Reshi, J. and Singh, S., "Predicting Software Defects through SVM: An Empirical Approach". International Journal of Scientific Research and Development, 2017, 5(5), 1835 – 1838.

[22] Jin, C., Jin, S. and Ye, J., "Artificial Neural Network-Based Metric Selection for Software Fault-Prone Prediction Model". IET Software, 2012, 6(6), 479–487.

[23] Kumar, L. and Rath S., "Neuro-Genetic Approach for Predicting Maintainability Using Chidamber and Kemerer Software Metrics Suite". In: Unger H., Meesad P., Boonkrong S. (eds) Recent Advances in Information and Communication Technology. Advances in Intelligent Systems and Computing, vol 361. Springer, Cham, 2015, 31 – 40.

[24] Erturk, E. and Sezer, E., "Iterative Software Fault Prediction with a Hybrid Approach", Applied Soft Computing, 2016, vol. 49, 1020-1033.

[25] Shepperd, M., Kadoda, G., "Comparing Software Prediction Techniques using Simulation". IEEE Transaction in. Software. Engineering, 2001, 27 (11), 1014–1022.

[26] Pandey, A., and Goyal, N., "Fault Prediction Model by Fuzzy Profile Development of Reliability Relevant Software Metrics". International Journal of Computer Applications, 2010, 11, 34–41.

[27] Erturk, E. and Sezer, E., "Software Fault Prediction using Fuzzy Inference System and Object-Oriented Metrics". In proceedings of the IASTED International Conference on Software Engineering, Innsbruck, Austra, 2014, 101 – 108.

[28] Chatterjee, S., and Maji, B., "A New Fuzzy Rule-Based Algorithm for Estimating Software Faults in Early Phase of Development". Soft Computing, 2016, 20, 4023–4035.

[29] Jaikumar, M. and Ramani, A., "Software Defect Prediction using Fuzzy Logic System". International Journal of Innovations and Advancement in Computer Science, 2017, 6(3), 118 – 124.

[30] Chatterjee, S., Maji, B. and Pham, H., "A Fuzzy Rule-Based Generation Algorithm in Interval Type-2 Fuzzy Logic System for Fault Prediction in the Early Phase of Software Development". Journal of Experimental and Theoretical Artificial Intelligence, 2018, 31(3), 369 – 391.

DOI: 10.31695/IJASRE.2021.1.34014

[31] Eyoh, I., John, R. and De Macre, G., "Interval Type-2 Intuitionistic Fuzzy Logic System for Non-linear System Prediction". In Proceedings of IEEE International Conference on Systems, Man and Cybernetics, Budapest, Hungary, 2016, 1063 – 1068.

[32] Son, L., Pritam, N., Khari, M., Kumar, R., Phuong, P. and Thong, P., "Empirical Study of Software Defect Prediction: A Systematic Mapping". Symmetry, 2019, 11(2), 212, **https://doi.org/10.3390/sym11020212**

[33] Atanassov, K. T., "Intuitionistic fuzzy sets," Fuzzy sets and Systems, vol. 20, no. 1, pp. 87–96, 1986.

[34] Eyoh, I. J., Umoh, U. A., Inyang, U. G., and Eyoh, J. E., "Derivative-Based Learning of Interval Type-2 Intuitionistic Fuzzy Logic Systems for Noisy Regression Problems". International Journal of Fuzzy Systems, 2020, 1-13.

[35] Imo Eyoh, Jeremiah Eyoh, and Ini Umoeka., "Interval Type-2 Intuitionistic Fuzzy Logic System for Forecasting the Electricity Load". International Journal of Advances in Scientific Research and Engineering, IJASRE (ISSN: 2454 - 8006), 6(10), (2020). 38-51. https://doi.org/10.31695/IJASRE. 2020, 33903

[36] Eyoh, I., Eyoh, J. and kalawsky, R., "Interval Type-2 Intuitionistic Fuzzy Logic for Time Series and Identification Problems: A Comprehensive Study". International Journal of Fuzzy Logic Systems, 2020, 10(1), 1 – 17.

[37] Luo, C., Tan, C., Wang, X., and Zheng, Y., "An evolving recurrent interval type-2 intuitionistic fuzzy neural network for online learning and time series prediction". Applied Soft Computing, 2019, 78, 150-163.

[38] Eyoh, I., John, R., De Maere, G., "Time series forecasting with interval type-2 intuitionistic fuzzy logic systems," In: 2017 IEEE international conference on fuzzy systems (FUZZ-IEEE). IEEE. 2017a.

[39] Imo Eyoh, Jeremiah Eyoh and Roy Kalawsky, "Interval Type-2 Intuitionistic Fuzzy Logic System for Time Series and Identification Problems - A Comparative Study", International Journal of Fuzzy Logic Systems (IJFLS), 2020, Vol.10, No.1, DOI: 10.5121/ijfls.2020.10101.

[40] Eyoh, I., John, R., De Maere, G., "Extended Kalman filter-based learning of interval type-2 intuitionistic fuzzy logic system. In: 2017 IEEE international conference on systems, man, and cybernetics (SMC). IEEE, 2017b.

[41] Eyoh, I., John, R., De Maere, G., "Interval type-2 intuitionistic fuzzy logic systems—a comparative evaluation". In: International conference on information processing and management of uncertainty in knowledge-based systems. Springer, Cham (2018).

[42] Eyoh, I., Eyoh, J., Umoh, U., and Kalawsky, R., "A Sliding Mode Control Learning of Interval Type-2 Intuitionistic Fuzzy Logic for Non-Linear System Prediction". Solid State Technology, 2020, 63(6), 7793-7811.

[43] Eyoh, I., John, R., Maere, G.D., Kayacan, E., "Hybrid learning for interval type-2 intuitionistic fuzzy logic systems as applied to identification and prediction problems". IEEE Trans Fuzzy Syst, 26(5), 2672–2685, 2018.

[44] Yuan, W., and Chao, L., "Online evolving interval type-2 intuitionistic fuzzy LSTM-neural networks for regression problems". IEEE Access, 2019, 7, 35544-35555.

[45] Eyoh, I., John, R., Maere, G.D., "Interval type-2 A-intuitionistic fuzzy logic for regression problems". IEEE Trans Fuzzy Syst 2018, 26(4), 2396–2408.

[46] Ebrahimnejad, A., and Verdegay, J. L., "An efficient computational approach for solving type-2 intuitionistic fuzzy numbers-based transportation problems". International Journal of Computational Intelligence Systems, 2016, 9(6), 1154-1173.

[47] Nguyen, D. D., Ngo, L. T. and Pham, L. T., "Interval type-2 fuzzy c- means clustering using intuitionistic fuzzy sets", in *IEEE Third World Congress on Information and Communication Technologies (WICT)*, 2013, pp. 299–304.

[48] Ha´jek, P., Olej, V., "Intuitionistic fuzzy neural network: the case of credit scoring using text information", pp. 337–346. Cham, Springer, 2015.

[49] Tai, K., El-Sayed, A.-R., Biglarbegian, M., Gonzalez, C. I., Castillo, O. and Mahmud, S., "Review of recent type-2 fuzzy controller applications," Algorithms, vol. 9, no. 2, p. 39, 2016.

[50] Begian, M. B., Melek, W. W. and Mendel, J. M., "Parametric design of stable type-2 tsk fuzzy systems," in IEEE Annual Meeting of the North American Fuzzy Information Processing Society, (NAFIPS), pp. 1-6, 2008.

[51] Nie M. and Tan, W. W., "Towards an efficient type-reduction method for interval type-2 fuzzy logic Systems", in Fuzzy Systems, 2008. FUZZ-IEEE 2008. (IEEE World Congress on Computational Intelligence). IEEE International Conference on, pp. 1425-1432, IEEE, 2008.

DOI: 10.31695/IJASRE.2021.1.34014

[52] Wu D. and Tan, W. W., "Computationally efficient type-reduction strategies for a type-2 fuzzy logic Controller", in Fuzzy Systems, 2005. FUZZ'05. The 14th IEEE International Conference on, pp. 353-358, IEEE, 2005.

[53] Fenton, N., Neil, M., Marsh, W., Hearty, P., Radlinski, L. and Krause, P., "On the Effectiveness of Early Life Cycle Defect Prediction with Bayesian Nets". Empirical Software Engineering, 2008. 13(5), 499 – 537.

[54] Pandey, A. and Goyal, N., "Multistage Model for Residual Fault Prediction", Studies in Fuzziness and Soft Computing, Springer, 2013, 59–80.

[55] Ini Umoeka, Imo Eyoh, Edward Udo and Veronica Akwukwuma, "Optimization of Interval Type-2 Fuzzy Logic System for Software Reliability Prediction", International Journal of Engineering Research and Advanced Technology, 2020, Vol. 6, No. 11, 1-12. DOI: 10.31695/IJERAT, 2020, 3665.

DOI: 10.31695/IJASRE.2021.1.34014