IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Special Issue-15, May 2019 E-ISSN: 2347-2693

An Innovative Approachto Perform Software Defect Prediction

Prakash Behera[1*], Chimaya Dash[2], R Chandramma[3], Prakash Behera[3], Piyush Kumar Pareek[4], Aditya Pai H[5]

[1,2]Dept. of CSE, Claret College, Jalahalli, Bengaluru and Research Scholar, JJTU, Rajasthan, India
[3]Dept. of Computer Science and Engineering, VKIT, Bengaluru and Research Scholar - EWIT, VTU Belgaum, India
[4]Department of Computer Science and Engineering, EWIT, Bengaluru, India
[5]Dept. of Computer Science and Engineering, K.S. Institute of Technology, Bengaluru and Research Scholar, VTU Belgaum, Bengaluru, India

DOI: https://doi.org/10.26438/ijcse/v7si15.296303 | Available online at: www.ijcseonline.org

***Abstract*—**identifying defective substances from existing software frameworks is an issue of extraordinary significance for expanding both software quality and the proficiency of software testing related exercises. We present in this paper a novel methodology for anticipating software defects utilizing fuzzy decision trees. Through the fuzzy methodology we plan to all the more likely adapt to clamor and loose data. A fuzzy decision tree will be prepared to recognize whether a software module is defective or not. Two open source software frameworks are utilized for tentatively assessing our methodology. The acquired outcomes feature that the fuzzy decision tree approach beats the non-fuzzy one on practically all contextual investigations utilized for assessment. Contrasted with the methodologies utilized in the writing, the fuzzy decision tree classifier is appeared to be more effective than the greater part of the other machine learning-based classifiers.

***Keywords*—**Software defect prediction,Machine learning,Decision tree, Fuzzy theory.

I. INTRODUCTION

Software quality assurance is a noteworthy issue in the software designing field and is utilized to guarantee the software quality. So as to expand the adequacy of quality assurance and software testing, defect prediction is utilized to recognize defective modules in an up and coming variant of a software framework and is helpful for relegating more exertion for testing and dissecting those modules [1].

The majority of the machine learning based classifiers existing in the defect prediction writing are administered. From this point of view, the issue of precisely foreseeing the defective modules is a hard one, as a result of the imbalanced idea of the preparation information (the quantity of non-defects in the preparation information is a lot higher than the quantity of defects). Along these lines, it is difficult to prepare a classifier to perceive the defects, when a little number of defective models were given amid preparing. A real test in defect prediction is to build the number of effectively distinguished defects and to limit the quantity of misclassified defects. Considerably more, it is difficult to distinguish the applicable software measurements which would most likely separate among defects and non-defects.

So as to manage the previously mentioned issues, we are presenting in this paper a managed machine learning strategy dependent on fuzzy decision trees for distinguishing defects in existing software frameworks. Supposedly, our methodology is novel in the defect prediction writing. The exploratory assessment of the fuzzy decision tree is performed on two open source software frameworks and demonstrates that our proposition gives preferable outcomes over most comparative existing ones.

The remainder of the paper is sorted out as pursues. The significance of the software defect prediction issue, just as the inspiration driving our methodology are displayed in Section II. The essentials of fuzzy decision trees and a related work on software defect prediction are given in Section III. Our approach for identifying software defects utilizing fuzzy decision trees is presented in Section IV. Area V gives a test assessment of the fuzzy decision trees on two open-source software frameworks. An investigation of the got results and a correlation with comparable existing work is displayed in Section VI. Area VII blueprints the finishes of the paper also, gives a few headings for future research.

II. MOTIVATION

Software defect location speaks to the movement through which software modules which contain mistakes are distinguished. Surely, the disclosure of such defective modules plays a significant job in guaranteeing the quality of the software improvement process. A movement which is likewise associated with keeping up the software quality is the code audit. Checking on the current code is tedious and exorbitant and it is every now and again utilized in the

coordinated software improvement. Software defect location can be useful in the code audit procedure to call attention to parts of the source code where it is probably going to recognize issues.

From a managed learning point of view, the issue of recognizing defective software elements is an unpredictable and troublesome one, primarily on the grounds that the preparation information is very imbalanced. Clearly, a software framework contains few defective elements, contrasted with the quantity of non-defective ones. In this way, a managed classifier for defect recognition will be prepared with a lot of defective precedents which is much littler than the arrangement of non-defective ones. Thusly, the classifier would be helpless to figure out how to dole out the larger part class, in particular the non-defective class. That is the reason, the field of software defect prediction is a functioning exploration region, being a constant enthusiasm for creating performance classifiers which can deal with the imbalanced idea of the software defect information.

A few investigations that have been performed in the defect prediction writing [2] have demonstrated that defect information removed from change logs and bug reports might be boisterous and uncertain [3]. Our past research in the defect prediction field (like [4]) strengthened that it is extremely elusive a fresh partition between the defective and non-defective substances, by and large defective substances appear to be fundamentally the same as to non-defective ones. One self-sorting out guide utilized in [4] uncovered that the defect information contains some dubious zones (covering zones among defects and non-defects) that can lead fresh classifiers to wrong predictions. That is the reason we think about that the fuzzy methodologies would be a decent decision for attempting to ease the recently referenced issues.

III. FUZZY DECISION TREES

Fuzzy decision trees [5] have been examined in the delicate registering writing as a hybridization between the traditional decision trees [6] and the fuzzy rationale. The established calculations for structure decision trees (ID3, C4.5) were stretched out toward a fuzzy setting [7] by thinking about parts of fluffiness and vulnerability. At each inside hub of the fuzzy tree, all examples from the informational index are utilized, yet each case has a specific participation degree related. At the root hub, all examples have the participation degree 1. Each inward hub contains a quality (chose utilizing Information Gain - Formula (4)) and has one tyke hub for each fuzzy capacity related to the chose trait. Every one of these youngster hubs will contain all cases, yet the participation level of each case from the parent hub will be increased by the estimation of the fuzzy capacity for the given case. A leaf hub from the fuzzy decision tree, rather than containing a solitary class (target esteem) as in the traditional methodology, contains the extent of the total enrolment esteems as for the aggregate enrolment for every one of the classes.

A fuzzy decision tree is utilized diversely when another example must be grouped (tried) than a customary one. The test example will be considered to have a place with all branches of the fuzzy decision tree with various degrees given by the stretching fuzzy capacity. A last fuzzy enrolment esteem will be gotten, along these lines, for each leaf hub in the tree. All the enrolments for the leaf hubs are summed for each objective class. The class having the most extreme related participation esteem will be considered as the last grouping for the testing occasion.

Normally, the fuzzy decision tree approach thoughtfully joins the fresh methodology when the participation degrees of the fuzzy sets utilized in the process portray fresh enrolments. The exemplary decision tree is accordingly a subclass of the fuzzy decision tree and the execution of each fuzzy variation will be in any event in the same class as the fresh reporter.

Be that as it may, the issue of defect prediction is testing because of the imbalanced idea of the preparation informational indexes. Normally the defective substances inside a software venture are fundamentally scarcer than the non-defective ones and along these lines the arrangement utilizing decision trees isn't a simple undertaking to be tackled, in light of the fact that the Entropy and the Information Gain measures, which have a major impact in the decision process, are firmly subject to the parity in size between the objective classes utilized in preparing.

Another issue happens when the likelihood appropriations of the characteristics inside the informational index are registered independently on every one of the objective classes. It would have been liked that the traits display a typical Gaussian pattern as this will help the decision procedure, yet the likelihood examination of the informational indexes immediately uncovered that the greater part of the traits fall under a lognormal conveyance with numerous qualities swarmed towards 0. For this situation it is profoundly hard for any type of decision tree to separate appropriately between the two classes as the occasions in the two gatherings will in general have a similar conduct what's more, are impeccably divergent all through the area making a clear gathering delimitation a genuine test. Indeed, even in the fuzzy viewpoint it is extremely hard to settle on one class and the different as the defective versus non-defective gatherings cover altogether enough to regard a considerable lot of the occasions to be characterized dubious.

IV. RELATED WORKS

Software defect location is a well-examined issue, there are a wide range of methodologies displayed in the writing that endeavor to distinguish the defective elements in a software framework. A writing study distributed in 2011, [8], found that 208 papers were distributed regarding this matter somewhere in the range of 2000 and 2010 and from that point forward the quantity of papers has expanded. A large portion of these approaches are administered, implying that they require a few preparing information so as to manufacture the model. There are a few transparently accessible informational indexes that can be utilized for preparing, and in this area we are going to introduce a few methodologies from the writing that utilization for the trial assessment the equivalent informational collections that we have utilized: JEdit and Ant. Okutan and Yildiz present in [9] a methodology that employments Bayesian Networks and the K2 calculation for defect discovery.

Other than the effectively existing software measurements they include two new measurements to the informational collection: absence of coding quality (LOCQ) and number of engineers (NOD). For the exploratory assessment, they utilize 9 openly accessible informational indexes (counting JEdit and Ant) also, the usage of the K2 calculation from Weka [10].

In light of the produced Bayesian Networks they examine the adequacy of various software metric sets for defect discovery, and presume that the LOC-RFC, RFC-LOCQ, RFCWMC sets are the best.

Multivariate Logistic Regression is utilized by Malhotra in [11] to recognize the defective substances in the Ant framework. The creators initially recognize and expel anomalies from the information, at that point apply the Multivariate Logistic Regression utilizing 10-overlap cross approval. The assembled model incorporates two measurements from the information set: RFC and CC.

While defect location is generally considered as a twofold grouping issue, the creators in [12] think of it as a relapse issue and they endeavour to foresee the accurate number of defects in every element. They look at six changed relapse techniques, Linear Regression, Bayesian Ridge Regression, Bolster Vector Regression, Nearest Neighbours Regression, Decision Tree Regression, Gradient Boosting Regression, and presume that Decision Tree Regression gives the best outcomes regarding exactness and root mean square blunder. The creators additionally explore the contrast between inside undertaking defect prediction (when the prediction model is constructed based on past form of a similar software framework) and cross project defect prediction (when the model is based on other ventures). In contrast to other such investigations, they presume that cross project defect prediction models are practically identical to inside undertaking defect prediction models concerning prediction execution.

The creators in [13] present a cross-venture defect prediction approach too, yet they plan the issue as a multi-target enhancement issue where two extraordinary targets must be considered: the quantity of defect inclined substances identified and the expense of breaking down the anticipated defect inclined classes. Their methodology depends on a multi-objective Hereditary Algorithm, and was tried utilizing 10 unique informational indexes, counting JEdit and Ant. They infer that the multi-objective approach accomplishes preferred execution over the single-objective approach they utilized for examination.

Scanniello et al. present a methodology, where the classes from the software framework are first bunched, to recognize groups of emphatically associated classes, at that point Stepwise Linear Regression is utilized to assemble a defect location model for each bunch independently [14]. Contrasted with the methodology where all classes are utilized together to assemble a location model, this methodology can give an increasingly exact discovery of the quantity of issues for each class.

V. METHODOLOGY

In this segment we present our fuzzy decision tree based classifier for recognizing defective software elements in existing software frameworks. As we have recently presented in [4], the elements from a software framework (classes, strategies, capacities) may be spoken to as high-dimensional vectors speaking to the estimations of a few software measurements connected to the considered element. Subsequently, a software framework S is seen as a lot of substances (occurrences) S = {e1, e2, ..., en} [4]. A lot of software measurements will be utilized as the list of capabilities describing the substances from the software framework, M = {m1,m2, ...,ml}. In this way, an element ei∈ S might be envisioned as a l-dimensional vector, ei = (ei1, ei2, . . . ,eil), where eij speaks to the estimation of the software metric mj connected to the software element ei.
As in a directed learning situation, the mark (class) related for every substance is known (D=defect, N=non-defect). The initial step before applying the fuzzy decision tree based learning approach is the information pre-processing step. At that point, the pre-processed preparing information will be utilized for structure (preparing) the fuzzy decision tree based classifier. The fabricated order model will be then tried so as to assess its execution. These means will be itemized in the accompanying.

A. Information pre-processing

Amid this progression, the informational index speaking to the high dimensional software substances will be pre-processed. An element determination step will be utilized so as to distinguish a subset of software measurements that are pertinent for anticipating software defects.

The informational collections that will be utilized for the exploratory assessment of our methodology were made for open-source object oriented software frameworks. In these informational collections every substance relates to a class from the software framework and contains information to distinguish the module (name of the framework, variant of the framework, name of the class) and the estimation of 20 unique software measurements in addition to the quantity of bugs in the given element.

Table 1. Thresh hold value for software metrics

Software Metric	a	b
WMC	10	22.9
CBO	10	20
RFC	30	66

Amid the information pre-processing step, we initially change the number of bugs into a parallel property, to indicate whether the element is defective or not. The esteem 0 will be utilized for non-defective elements and 1 will be utilized for the defective ones. So as to diminish the quantity of software measurements in the information set, we have utilized the discoveries of a methodical writing audit directed on 106 papers [15], which ponders the pertinence of 19 diverse software measurements for the assignment of software issue prediction. Out of the measurements revealed by the examination as having a solid positive adequacy on software flaw prediction, three can be found in our informational indexes, so we have chosen to kill different measurements. The measurements kept after the pre-processing are: WMC, CBO and RFC.

So as to construct the fuzzy sets for the chose software measurements, we have taken motivation from crafted by Fil'oet al. [16]. They have utilized 111 software frameworks written in Java and processed the estimation of 17 distinctive software measurements for each class of the frameworks. For every metric they have recognizes edges to gather the estimation of the measurement in three ranges: Good/Common, Regular/Casual, and Bad/Uncommon.

The edge between the initial two territories was figured as the 70 percentile of the information, while the second edge was considered at the 90 percentile. Since the examination displayed in [16] contains limits for just a solitary one of the software measurements that we are utilizing, we have chosen to figure our own edges.

We have taken all informational collections from the Tera-Promise vault [17] that have a place with the Defect classification and utilize the equivalent software measurements as the informational indexes utilized for the trial assessment. If there should be an occurrence of informational collections with numerous forms, we have taken the last form. Along these lines, we have fabricated an informational index containing 6082 occasions originating from an aggregate of 30 ventures. We have figured the 70 and 90 percentile for the measurements and utilized these qualities as edges for structure two trapezoidal participation capacities for the non-defect and defect classes.

The primary capacity estimates the participation of a given software metric incentive to the class of non-defective substances, while the second one quantifies the enrolment to the class of defective substances. The two fuzzy enrolment capacities for the WMC software metric are outlined on Figure 1. Formulae 1 and 2 portray the conditions used to register the participation degree of a software metric incentive to the non-defect, individually defect fuzzy sets. The definite limit esteems utilized for every one of the three measurements are exhibited in Table 1.

$$\mu_{non-defect}(x) = \begin{cases} 1, & x < a \\ \frac{b-x}{b-a} & a \le x \le b \\ 0, & x > b \end{cases} \quad (1)$$

$$\mu_{defect}(x) = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a} & a \le x \le b \\ 1, & x > b \end{cases} \quad (2)$$

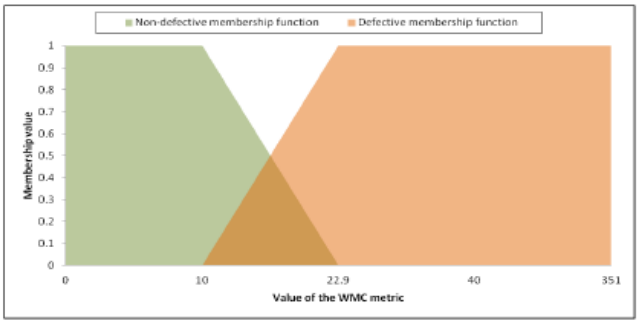

Figure 1. Fuzzy functional membership for software metric

B. Training

Amid the preparation procedure the fuzzy decision tree is worked from the informational collection that was pre-processed as exhibited in the past area. Defect identification informational collections are for the most part imbalanced, which can impact the preparation procedure and lead to a fuzzy decision tree where each leaf hub predicts that the given occasion is non-defective. So as to decrease the unevenness of the informational index, we improve it before

the preparation process, by including additional defective occasions from other information sets. These occasions are utilized just for the preparation procedure, they are not considered amid the testing.

1) Building the fuzzy decision tree (FuzzyDT): The structure process for the fuzzy decision tree proposed in the current paper looks like the one for a fresh variation of a decision tree with a few adjustments to adapt to vulnerability and information irregularity. Both of these angles have a significant impact on the proposed fuzzy decision tree variant molding it into a custom variation customized to take care of the given issue as precisely as could be expected under the circumstances. So as to deal with the vulnerability of software defect prediction the defective and non-defective ideas should have been formalized as fuzzy sets as for every one of the traits inside the informational collection. The technique for fuzzy set development for both of the objective classes was displayed in the past area, in any case, it must be included that a concentrated choice procedure was important to feature the properties that may have a helpful sway on the fuzzy decision process the same number of qualities in the informational collection were normally not fit to help any type of arrangement. This is a viewpoint that adds to the trouble of the defective/non-defective order task. It must be referenced that all together for the fuzzy way to deal with work, the fuzzy enrolment capacities utilized in the decision procedure should be dealt with in all respects carefully ideally their devise being the product of a cooperation with software building specialists. On the off chance that the fuzzy enrolment capacities don't delineate onto genuine settings, the entire decisional procedure will be influenced.

When the fuzzy enrollment capacities are developed for each quality, independently on each objective class, the genuine fuzzy decision tree development may initiate. Now, the other serious issue examined in the presentation happens: information irregularity. Because of the scarceness of defective examples, the defective target class will be plainly imbalanced with deference to the non-defective target class on the contemplated traits. In the exemplary fuzzy decision tree approach, the fuzzy entropy and fuzzy data gain measures are extremely one-sided with deference to information lop-sidedness and this effects the decision procedure in the sense that there is an unmistakable tendency towards marking cases as non-defective just in light of the fact that the preparation set contains an essentially expanded number of non-defective occurrences. This is a significant issue with profound ramifications in the decisional process and in this way figuring out how to manage information lop-sidedness was imperiously vital.

An answer for the awkwardness issue was proposed in [18]. Rather than following other rather short-sighted approaches that straightforwardly influence the informational index, for example, oversampling or then again under-inspecting, which as we would like to think are definitely not fit for the present issue in light of the fact that the inconsistency between defective and non-defective examples is excessively high, the creators propose a method for adapting to the awkwardness by changing the entropy and data gain measures. Along these lines, from a constructional perspective, the main adjustment will be changing the entropy and data gain formulae to a structure that considers the unevenness and incorporates it in the calculation, in this manner weakening its effect. Give us a chance to consider, in the accompanying, that the defective class is the positive one also, the non-defective class is the negative one. As we have referenced in Section III-An, each inner hub from the tree stores every one of the cases from the preparation informational index D, however each case has a specific enrolment degree. The entropy measure at a hub from the fuzzy tree is processed as in Equation (3) and sums up the entropy calculation from the fresh case.

$$Entropy(node) = -\frac{m_+}{mm} \cdot \log \frac{m_+}{mm} - \frac{m_-}{mm} \cdot \log \frac{m_-}{mm} \quad (3)$$

Where m+ speaks to the aggregate of the enrolment degrees for the cases from D having a place with the positive class, m− wholes the enrolment degrees for the occasions from D having a place to the negative class and mm is the whole of m+ and m−. For registering the data addition of a characteristic a with regard to the arrangement of occurrences put away at an inward hub from the fuzzy tree, a sort of disarray lattice at that hub is registered. We mean by Fa+ also, Fa− the fuzzy capacities related to credit and to the positive and negative class, separately. By TPFuzzy, FPFuzzy and FNFuzzy we express the qualities which sum up (for the fuzzy case) the segments of the perplexity lattice for the fresh case. More precisely, these qualities are processed as pursues:

- TPFuzzy entireties the enrollment degrees for the examples I having a place with the positive class increased with the aftereffect of applying the capacity Fa+ on the estimation of trait an in occurrence I.
- FNFuzzy totals the enrollment degrees for the examples I having a place with the positive class duplicated with the aftereffect of applying the capacity Fa− on the estimation of characteristic an in occasion I.
- TNFuzzy wholes the enrollment degrees for the cases I having a place with the negative class increased with the aftereffect of applying the

capacity Fa− on the estimation of characteristic an in occasion I.

- FPFuzzy wholes the enrollment degrees for the cases I having a place with the negative class increased with the aftereffect of applying the capacity Fa+ on the estimation of characteristic an in occasion I.

We use the following notations:

- $m = TP^{Fuzzy} + TN^{Fuzzy} + FP^{Fuzzy} + FN^{Fuzzy}$.
- $p = TP^{Fuzzy} + FN^{Fuzzy}$.
- $pp = TP^{Fuzzy} + FP^{Fuzzy}$.

Using the previous notations, the new formula for the *information gain* measure is presented in Formula (4).

$$IG(node) = Entropy(node) - \frac{pp}{m} \cdot E_1 - \frac{m-pp}{m} \cdot E_2 \quad (4)$$

where

$$E_1 = -\frac{TP^{Fuzzy}}{pp} \cdot \log \frac{TP^{Fuzzy}}{pp} - \frac{FP^{Fuzzy}}{pp} \cdot \log \frac{FP^{Fuzzy}}{pp}$$

and

$$E_2 = -\frac{TN^{Fuzzy}}{m-pp} \cdot \log \frac{TN^{Fuzzy}}{m-pp} - \frac{FN^{Fuzzy}}{m-pp} \cdot \log \frac{FN^{Fuzzy}}{m-pp}.$$

C. Testing

After the fuzzy decision tree was prepared (as portrayed in Segment IV-B1), another case will be delegated appeared Segment III-A. For assessing the general execution of the FuzzyDT model, a forget one cross-approval is utilized [19]. In the forget one (LOO) cross-approval on an informational index with n software elements, the FuzzyDT model is prepared on n-1 substances and afterward the acquired model is tried on the occurrence which was forgotten. This is rehashed n times, for every element from the informational collection. Amid the cross-approval process, the perplexity grid [20] for the two potential results (non-defect and defect) is processed. We are thinking about that the defective class is the positive one and the non-defective class is the negative one. The disarray grid contains four qualities, the quantity of True Positives (TP), True Negatives (TN), False Positives (FP) what's more, False Negatives (FN). For figuring the qualities from the disarray grid, we are utilizing the known names (classes) for the preparation cases.

Table 2. Description of the Data sets for evaluation

Data set	Defects	Non-defects	Difficulty
JEdit	48	319	0.6667
Ant	166	579	0.5723

Since the software defect prediction information are exceptionally imbalanced (the quantity of defects is a lot littler than the number of non-defects) the fundamental test in software defect prediction is to get a substantial genuine positive rate and a little false negative rate. For defect indicators, the precision of the classifier (for example number of testing occasions which were accurately arranged - Formula (5)), is anything but a significant assessment measure, since the imbalanced idea of the information.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

A progressively pertinent assessment measure for the execution of the software defect classifiers is the Area Under the ROC Bend (AUC) measure [21] (bigger AUC esteems show better defect indicators). The AUC measure is generally utilized if there should be an occurrence of approaches that yield a solitary esteem which is changed into a class mark utilizing an edge. For such methodologies, altering the estimation of the edge can prompt various estimations of the Likelihood of recognition (Formula (6)) and the Probability of false caution (Formula (7)) measures. For every limit, the point (Pf, Pd) is spoken to on a plot, and AUC measures the territory under this bend. [11]

$$Pd = \frac{TP}{TP + FN} \quad (6)$$

$$Pf = \frac{FP}{FP + TN} \quad (7)$$

If there should arise an occurrence of methodologies where the yield is legitimately the class mark, there is just one (Pf, Pd) point, which can be connected to the (0,0) and (1,1) points, and the zone under this bend can be processed utilizing Formula (8).

$$AUC = (1 - Pf) * Pd + \frac{Pf * Pd}{2} + \frac{(1 - Pf) * (1 - Pd)}{2} \quad (8)$$

VI. EXPERIMENTAL EVALUATION

FuzzyDT model (portrayed in Section IV) on two open source software frameworks which were recently utilized in the software defect prediction writing. We notice that we have utilized our own execution for FuzzyDT, without utilizing any outsider libraries. [24] [25]

A. Contextual investigations

For the trial assessment of the FuzzyDT model we have utilized two transparently accessible informational collections, made for two software frameworks written in Java: JEdit (adaptation 4.2)1 and Ant (form 1.7)2. The two informational indexes are accessible at [17]. Subtleties about these two informational indexes can be found in Table II.

The last section of Table II contains the trouble of the informational indexes. This measure was presented by Boetticher in [22] and is registered as the level of elements for which the closest neighbor (disregarding the name of the substance when registering the separations) has an alternate name. Since our information sets are imbalanced, when registering the trouble of the information sets we considered just the level of defective elements for which the closest neighbor is non-defective. For every datum set that is utilized for the test assessment, we will perform two investigations. In the main trial we are going to utilize the informational index with no change, while in the second investigation we are going to improve it by adding to the informational index defective substances taken from an alternate software framework. We are adding additional defective elements to lessen the unevenness in the informational collection. In the writing, two choices are normally displayed for including increasingly defective elements: oversampling (at the point when some defective elements are copied) and Destroyed (when new minority-class substances are made utilizing the current ones) [23]. We trust that utilizing genuine defective substances from an alternate task is superior to making manufactured substances. For the two informational collections, we have included as additional defective substances, all the defective elements from the Tomcat informational index, which is likewise accessible at the Tera-Promise archive [17]. Therefore, we have added 77 defective substances to the two information sets, expanding the level of defective substances from 0.131 to 0.282 (for JEdit) and from 0.223 to 0.30 (for Ant). [9] [10]

Table 3. Experimental Evaluation Results

Data set	TN	TP	FN	FP	AUC	Accuracy
JEdit original	305	18	30	14	0.666	0.88
JEdit enhanced	289	27	21	30	0.734	0.86
Ant original	539	60	106	40	0.646	0.80
Ant enhanced	526	84	82	53	0.707	0.82

B. Results

Table 3 contains the aftereffects of the trial assessment. As introduced in the past segment, for every datum set we have run the FuzzyDT model both for the first informational index and the informational index improved with the defective substances taken from the Tomcat framework. We notice that these defective elements were utilized just for the preparation of the model, the testing was performed just on the elements from the JEdit and Ant frameworks. Other than the AUC execution measure - registered with the Formula (8) - we have chosen to add to Table 3 the whole disarray network to permit the calculation of any execution measures for our methodology, to encourage the examination of our results to different methodologies. While we contended that exactness is certifiably not a decent exhibition measure if there should be an occurrence of imbalanced informational collections, we have chosen to add it to Table III to demonstrate how diverse the estimations of this measure are contrasted with AUC. [13]

VII. CONCLUSION AND FUTURE WORK

A fuzzy decision tree model has been presented for foreseeing, in an administered way, those elements from software frameworks which are probably going to be defective. The test assessment which was performed on two open-source software frameworks gave results superior to anything the greater part of the comparable existing approaches and featured a generally amazing exhibition of the proposed methodology. Considerably more, the fuzzy decision tree approach demonstrated to outflank, for the considered contextual investigations, the fresh DT approach.

Further work will be done so as to expand the test assessment of the fuzzy decision tree approach proposed in this paper. We likewise mean to research a hybridization between the fuzzy DT model and social affiliation rules [26], since we are sure that relations between the values for various software measurements would be pertinent in separating among defective and non-defective software elements.

REFERENCES

[1] R. hua Chang, X. Mu, and L. Zhang, *"Software defect prediction using non-negative matrix factorization."* JSW, vol. **6**, no. **11**, pp. **2114–2120**, **2011**.

[2] J. Aranda and G. Venolia, *"The secret life of bugs: Going past the errors and omissions in software repositories,"* in Proceedings of the 31st International Conference on Software Engineering, ser. ICSE '09. Washington, DC, USA: IEEE Computer Society, pp. **298–308**, **2009**.

[3] S. Kim, H. Zhang, R. Wu, and L. Gong, *"Dealing with noise in defect prediction,"* in Proceedings of the 33rd International Conference on Software Engineering, ser. ICSE '11. New York, NY, USA: ACM, 2011, pp. **481–490, 2011**.

[4] Z. Marian, G. Czibula, I.-G. Czibula, and S. Sotoc, "Software defect detection using self-organizing maps," Studia Universitatis Babes-Bolyai, Informatica, vol. **LX**, no**. 2**, pp. **55–69**, **2015**.

[5] M. Umanol, H. Okamoto, I. Hatono, H. Tamura, F. Kawachi, S. Umedzu, and J. Kinoshita, *"Fuzzy decision trees by fuzzy id3 algorithm and its application to diagnosis systems,"* in Proceedings of the Third IEEE Conference on Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence.,,, pp. **2113–2118** vol.**3, 1994.**

[6] T. M. Mitchell, Machine learning. McGraw-Hill, Inc. New York, USA, **1997.**

[7] C. Z. Janikow, "Fuzzy decision trees: issues and methods," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. **28**, no. **1**, pp. **1–14**, **1998**.

[8] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, *"A systematic literature review on fault prediction performance in software engineering,"* IEEE Transactions on Software Engineering, vol. **38**, no. **6**, pp. **1276–1304**, **2011**.

[9] A. Okutan and O. T. Yildiz, *"Software defect prediction using Bayesian networks,"* Empirical Software Engineering, vol. **19**, no. **1**, pp. **154–181**, **2014**.

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, *"The weka data mining software: An update,"* SIGKDD Explorations, vol. **11,** no. **1,2009.**

[11] R. Malhotra, *"A defect prediction model for open source software,"* in Proceedings of the World Congress on Engineering, vol. **II**, July **2012.**
[12] M. Chen and Y. Ma, *"An empirical study on predicting defect numbers,"* in Proceedings of the 27th International Conference on Software Engineering and Knowledge Engineering, pp. **397–402, 2015.**
[13] G. Canfora, A. D. Lucia, M. D. Penta, R. Oliveto, A. Panichella, and S. Panichella, *"Multi-objective cross-project defect prediction,"* in Proceedings of the 6th International Conference on Software Testing, Verification and Validation, pp. **252–261, 2013.**
[14] G. Scanniello, C. Gravino, A. Marcus, and T. Menzies, *"Class level fault prediction using software clustering,"* in Proceedings of the 28th IEEE/ACM International Conference on Automated Software Testing, pp. **640–645, 2013**.
[15] D. Radjenovi´c, M. Heri˘cko, R. Torkar, and A. ˇ Zivkovi˘c, *"Software fault prediction metrics: A systematic literature review,"* Information and Software Technology, vol. **55**, no. **8**, pp. **1397–1418**, **2013**.
[16] T. G. S.Fil´o, M. A. S. Bigonha, and K. A. M. Ferreira, *"A catalogue of thresholds for object-oriented software metrics,"* in First International Conference on Advances and Trends in Software Engineering, **2015**.
[17] "Tera-promise repository," http://openscience.us/repo/.
[18] W. Liu, S. Chawla, D. A. Cieslak, and N. V. Chawla, *"N.: A robust decision tree algorithms for imbalanced data sets,"* in In: Proceedings of the Tenth SIAM International Conference on Data Mining, pp.**766–777, 2010**.
[19] G. Wahba, Y. Lin, and H. Zhang, *"GACV for support vector machines, or, another way to look at margin-like quantities,"* Advances in Large Margin classifiers, pp. **297–309**, **2000**.
[20] D. M. W. Powers, "Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation," Journal of Machine Learning Technologies, vol. **2**, no. **1**, pp. **37–63**, **2011**.
[21] T. Fawcett, *"An introduction to ROC analysis,"* Pattern Recogn. Lett., vol. **27**, no. **8**, pp. **861–874**, **2006**.
[22] G. D. Boetticher, *"Advances in Machine Learning Applications in Software Engineering"*. IGI Global, ch. Improving the Credibility of Machine Learner Models in Software Engineering, **2007.**
[23] N. V.Chawla, "Data Mining and Knowledge Discovery Handbook", Springer US, ch. Data Mining for imbalanced datasets: an overview, **2010/**
[24] "Orange data mining," http://orange.biolab.si/.
[25] N. V. Chawla, K.W. Bowyer, L. O. Hall, andW. P. Kegelmeyer, *"Smote: Synthetic minority over-sampling technique,"* J. Artif. Int. Res., vol. **16,** no. **1**, pp. **321–357**, **2002.**
[26] G. Serban, A. C˘ampan, and I. G. Czibula, *"A programming interface for finding relational association rules,"* International Journal of Computers, Communications & Control, vol. **I**, no. **S**., pp. **439–444**, June **2006**.

Authors Profile

Mr. Prakash Beherais currently working as assistant professor in St. Claret College. Also pursuing his PhD degree in Computer Science and Engineering in JJTU University in Rajasthan. His Area of Specialization is Software Engineering.

Mr. Chinmaya Dashis currently working as assistant professor in St. Claret College. Also pursuing his PhD degree in Computer Science and Engineering in JJTU University in Rajasthan. His Area of Specialization is Software Engineering.

*Mrs.*Chandrammahas received Master's degree in Computer science & Engineering from VTU, Belgavi in 2006, B.E in CSE from Bangalore University in 1999.Author has over 18 years of Academic experience. She has published one book in Lambert Publications, Germany and published several papers in International Journals and also published several papers in International or national conferences. She is currently serving in Department of Computer Science & Engineering at Vivekananda Institute of Technology Bengaluru and pursuing her Ph.D in Computer science & Engineering, VTU Belagavi. Her Interested areas of Research are Natural Language Processing, Data mining & Machine Learning.

*Dr.*Piyush Kumar Pareek is a Ph.D (2016)in Computer Science Engineering from Jain University , Bengaluru , M. Tech (2012) from Dayanand Sagar College of Engineering , Bengaluru & B.E (2010) from Basaveshwar Engineering College , Bagalkote , Author has over 8+ Years of Academic Experience , He is author or co-author of over 53 international journal papers or conference proceedings, He has Four Books published in Lambert Publications , Germany. He is BOS Member – Alumni – ISE - of Basaveshwar Engineering College, Bagalkote, He is PhD Alumni Convener of Jain University, Bengaluru. He is Recognized Research Supervisor in Computer Science & Engineering (VTU0817401), Visvesvaraya Technological University, Balagavi . He is currently serving in department of Computer Science & Engineering at East West Group of Institutions, Bengaluru. His Interested areas of Research are Software Engineering, Computer Networks & Machine Learning. He is Editorial Board Member & Reviewer of International Journal of Advanced Research in Computer Science & Technology (IJARCST), Brittany, FRANCE. Reviewer of International journal of Engineering Research and Technology ESRSA Publications (IJERT), INDIA , International journal of Computer Science Research and Technology (IJCSRT), INDIA, Inter Science Research Network (IRNET), Bhubaneswar, INDIA, International Journal of Computer Science and Informatics (IJCSI), Bhubaneswar, INDIA, International Journal of Engineering, Sciences and Management (IJESM), U.P, INDIA., International Journal of Scientific & Technology Research (IJSTR), Paris, FRANCE, Journal of Theoretical and Applied Computer Science (JTACS), Szczecin, Poland, International Journal of Advanced Research in Computer Science & Technology (IJARCST), INDIA, Journal of Harmonized Research in Engineering, INDIA.,IASTER's International Research Journals , INDIA, IJMAS Journals , Bhopal , INDIA, RR Journals ,INDIA , Reviewer of IRD INDIA & STEM Journals , INDIA

Mr. Aditya Pai H received the BE degree in Computer Science and Engineering in VTU, Belgaum in 2009 and pursued his masters MS degree in Information Technology and Management Manipal University, Manipal in 2012. He is currently pursuing PhD degree in Computer Science and Engineering under VTU, Belgaum at East West Institute of Technology Research Centre, Bengaluru. He has 7 years of teaching experience. Currently working as Assistant Professor in Department of Computer Science and Engineering, K.S. Institute of Technology, Bengaluru. He has two books being published in Lambert Publishing, Germany. He is also the member of ISTE. His research area is Software Engineering.

