IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-2, Feb 2019 E-ISSN: 2347-2693

A Novel Software Reliability Prediction Algorithm Using Fuzzy Attribute Clustering and Naïve Bayesian Classification

[1*]Neeta Rastogi, [2]Shishir Rastogi, [3]Manuj Darbari

[1]Department of Computer Science & Engineering, BBD National Institute of Technology & Management, Lucknow, India
[2]School of Computer Applications, BBD University, Lucknow, India
[3]Department of Computer Science & Engineering, School of Engineering, BBD University Lucknow, India

[*]*Corresponding Author: neeta.rastogi@yahoo.com Tel.: +91-9889200100*

DOI: https://doi.org/10.26438/ijcse/v7i2.7382 | Available online at: www.ijcseonline.org

Accepted: 18/Feb/2019, Published: 28/Feb/2019

Abstract: In current times, ubiquitous computing has given massive rise to research work in artificial intelligence, machine learning, software engineering and to research development in telecommunication, medicine, and image / audio / video processing. Due to the vastness of software being developed, software fault prediction is a very pertinent area for ensuring software quality and has so much scope to work. Machine learning now a days is one of the most promising way to deal with software fault prediction problems. The assumptions considered in a testing case need to be different from those in other testing cases because of the varying complexity of software testing. Although, there are software fault prediction models who can effectively assess software reliability in specific testing scenarios, no single model can accurately predict the fault numbers in a software in all testing conditions due to the fact that the modern software being developed are bigger and complex in both size and functions and thus, assessing the software reliability is a daunting task. Some popular approaches of Software fault prediction models use General Bayesian network and Augmented Naive Bayes classifiers, which do not impose any restriction on network architecture and are able to learn appropriate network architecture. An algorithm combining Fuzzy Attribute Clustering with Naive Bayes Classification has been worked out in this paper. The proposed Fuzzy Attribute Cluster Net Bayes (FACNB) algorithm is a machine learning-based prediction algorithm for software reliability prediction (using soft computing methods). It focusses on all data types in the area of software analytics. The prediction accuracy of the proposed algorithm shows improvement over other such algorithms.

Keywords- FACNB, Fuzzy Attribute Clustering, Software reliability model, Software reliability prediction, Bayes classifier, Machine learning algorithm.

I. INRODUCTION

Software fault prediction studies focused on to create prediction models that detect software components with more likelihood of having faults. Software metrics data and fault information from previous software releases are used to train the classification model, and this model is then used to predict the fault-proneness of the modules in new releases. Several change metrics are extracted from software repositories, and models are built using machine learning algorithms, namely Decision Tree, K-Nearest Neighbour, and Random Forest **[1]**. Among these attributes, the reliability is one of the most important quality factors, and it is the only common factor of different quality models i.e. McCall's, Boehm's, FURPS and ISO 25010 that replaced the ISO 9126 **[2]**. Reliability measurement covers different elements like reliability techniques, models, and metrics. To meet the demands of high quality and reliable software-based systems - the significant effort is devoted to software V&V (Verification & Validation). Intensive V&V helps software development organizations to ensure correct functionality and reliability of software systems; at the same time, it is also a resource intensive activity accounting for up to 50% of total software development costs and even more for software systems that are safety critical. Thus, having a good testing strategy is crucial for any industry with high software development costs. Since, finding and fixing defects is the major activity within software validation and overall the most expensive activity in embedded software development, predicting an expected defect inflow and/or a total defect count at early project stages allows to manage the limited test resources effectively, assess the time required to release the high-quality product and to avoid releasing software with critical defects. The regression-based quantitative models are limited in their applicability in early stages of a software project when the data of the defect inflow is not available,

where, such predictions can be most useful for planning and allocation of test resources. A remedy to using these regression-based models is to use Bayesian statistical estimation models. These Bayesian models combine the usage of data (parameters) from previous projects with the use of limited data sets of the current project at its early stage by predicting the total expected defects in the software lifecycle **[3]**. In recent years, traditional software development methods have been replaced with agile software development approaches. The dataset that includes collected code metrics for a finalized project is employed for both the testing stage and the training stage while employing a machine learning based method **[4]**. Fault-prone modules should be determined during development of the current version to ensure that possible faults can be located before the version comes into service by considering an iterative Software Fault Prediction (SFP) methodology based on the development of software projects **[5]**.

Rest of the paper is organized as follows, Section I contains the introduction of several software fault prediction techniques and approaches, Section II contains the related work of software fault prediction techniques using ANN and machine learning done so far leading to the objective of our research, Section III contain the research methodology along with mathematical modeling and experimental setup of proposed FACNB algorithm, Section IV describes results and discussions, Section V describes conclusions drawn.

II. RELATED WORKS

In literature, it is found that different models have different predictive capabilities and no single suitable model exists for reliability prediction which can be applied under all circumstances in a realistic environment. Some of the previously done work in this area is discussed in this section. Software Reliability Engineering (SRE) provides a means for the software practitioners to predict, estimate, and measure the rate of failure occurrences in the software **[5].** ***J. Wang et al., (2018)*** proposed a deep NN model which has better stability, robustness, and accuracy for software reliability prediction. It was limited by types and quantities of fault data sets.The proposed technique used only the fault detection data as training data for deep NN **[6]**. ***N. P. Padhy et al., (2018)*** developed an Aging- And Survivability-Aware (Sensitive) Reusability Optimization Model (ASROM) for object-oriented software systems.All the algorithms developed so far for object-oriented reusability metrics have been developed based on the Object-Oriented Chidamber And Kemerer (OO-CK) metrics suite. Until now, the researchers had considered only the theoretical aspects, but this paper addresses the development of reusability algorithms, models for reusability estimation and related techniques. This study demonstrated that it is feasible to derive an efficient and robust reusability prediction model for web-service products using OO-CK metrics. It was also found that OO-CK metrics, particularly complexity, cohesion and coupling-related metrics can be helpful in predicting reusability in web-service software products **[7]**. ***P. Roy et al., (2015)*** proposed a multi-layer feed-forward ANN based LGCM for software reliability estimation and prediction. This model can be applied in decision making situations like the software release time problem. This proposed ANN-based LGCM can be compared with various types of traditional statistical NHPP based SRGMs and more real software failure data sets can be used to train the ANN to further validate the model **[8]**. ***F. Febrero et al, (2016)*** presented the results of a Systematic Literature Reviews (SLR) on software reliability assessment based on representative International Standard proposals. The main outcome of this work is the confirmation that Standard Based-Software Reliability Modeling (SB-SRM) is receiving limited attention from the academic community in addition to having little impact on the industry, or at least industry is not reporting on its application **[9]**. ***M. Zhu et al., (2015)*** investigated the impact of software development Environmental Factors (EFs) on software reliability assessment. Statistical analysis method, such as principal component analysis, relative weighted method, Tukey method, backward elimination, and correlation analysis is applied to analyze the environmental factors. The study listed the most significant factors based on the general ranking and the principal components **[10]**. ***Jazdi et al., (2016)*** presented a method to calculate the reliability of industrial automation systems. Since, the software reliability is the crucial aspect that can influence the entire system reliability, it deserves special attention. For this reason, this paper discussed the influencing factors of the software reliability as well, for example, the software development process was analyzed to identify the most important influencing factors. The mentioned factors have been considered to create a Neuro-fuzzy-based concept, which characterizes and consolidates them to realize an estimation of the software quality **[11]**. ***D. Srdjana et al., (2017)*** developed a BN model for effort prediction in agile software development projects. This model is relatively small and simple, and all the input data are easily elicited, so that the impact on agility is minimal. The model predicts task effort, and it is independent of agile methods used. It is also suitable for use in the early project phase. The model is validated using a database of 160 tasks from real agile projects. The prediction accuracy is measured by the percentage of correct overall predictions. The model results in very good accuracy: only one misclassified value. Pred. (m=25) Equals 100% – all predictions are classified within 25% tolerance. The MMRE values show that there are no occasional large estimation errors **[12]**. ***O. F. Arar et al, (2017)*** proposed a modified classifier called the Feature Dependent Naïve Bayes (FDNB) method. The software defect prediction problem was selected as the application domain for this modified classifier, and eight publicly

available and widely used NASA data sets were utilized in the experiments. It was also shown that the improper design of a learner model can give misleading results. The obtained results indicated an outperformance of the proposed method over standard NB with feature subset selection pre-processing and other NB variations of weighted-feature characteristics **[13]**. ***I. Lakshmanan et al, (2015)*** proposed a new feed forward neural network based dynamic weighted combination model using a back-propagation algorithm as a Software Reliability Growth Models (SRGM) to improve the software reliability estimation accuracy. The performance comparison using practical software failure data sets show that the proposed model estimation accuracy is better than that of traditional SRGMs and already proposed neural network combination model. All the four SRGMs were evaluated using two practical software failure data sets with varying characteristics **[14]**.

III. RESEARCH METHODOLOGY

Machine learning algorithms have been widely used in various fields of research. We assume that collected data from sensors becomes faulty in the presence of faults. Therefore, system's normal and faulty behavior can be modeled from collected data using the machine learning algorithms during the training phase and recognized at runtime **[15]**. There are numerous machine learning algorithms for fault detection. In this paper, a novel Machine Learning is proposed to identify the accurate and stable features of software faults through a FACNB (Fuzzy Attribute Cluster Net Bayes) algorithm. A globally optimal solution can be achieved through FACNB algorithm, and it automatically learns the better feature characteristics of the software faults than traditional methods.

III.I Naïve Bayes Classifier

It is a classifier based on Bayes theorem used in software fault prediction **[16]**. It resolves the several difficulties like spam classification (to predict whether an email is a spam or not), medical diagnosis (given a list of symptoms, predict whether the patient has cancer or not) and so on. This method can be used to predict faulty and non-faulty modules. It is considered to provide better accuracy in comparison with other classifiers. It provides computational efficiency and is easy to construct, as no learning phase is required. An advantage of the naive Bayes classifier is that it only requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification. Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix **[17]**.

III.II Proposed FACNB algorithm

FACNB is used to examine the software data adequately and providing the better future results of software reliability level prediction. The proposed algorithm is as follows:

Step 1: Initiate the dataset collection
Dataset= $= \sum_{k=0}^{n} \text{Number of instances}$

Step 2: Select the input using Fuzzy attributes.

Step 3: Do the fuzzy conversion process
begin
Initialization i=0
NR=Number of records from FA (2)
MI=Number Attributes from datasets
SI=0 (or) 1 (o) r between 0 and 1
While (I <=NR)
IF (MI==1)
FC-> true
Else if (M1==0)
FC->false
Else if (M1>0 && M1<=1)
FC->between true and false
End

Step 4: Group the nodes using fuzzy clustering technique

Step 5: Create the local Bayesian networks for each cluster

Step 6: Compute the conditional probability of Naïve Bayes classifier using,

$$P(C = c_i \,|\, w) = P(C = c_i \,|\, \Lambda_{j=1}^{n} A_j = a_{i_j}^{(j)})$$

for all c_i and then predicts the class c_i for which this probability is highest.

Step 7: Discover the class probability

Step 8: Calculate the class probabilities for classes 0 or 1

Step 9: Calculate the conditional probabilities

$$P(A|B) = P(B|A) * P(A) / P(B)$$

Step 10: End the process.

The data of PC1 (NASA) is input to the system. FACNB requires pre-processing of data to minimize time consumption in order to cluster the faulty data. The output of C-Means clustering is stored for comparison. The output of C-Means is fed to adaptive Neuro-Fuzzy C-Means clustering algorithm. The algorithm tuned by Neural Network trains the data and improve performance index. The output of the algorithm is compared with an output of Fuzzy C-Means in terms of accuracy, reliability, RMSE, and MAE.

III.III NASA MDP

The NASA data sets can be obtained easily from the NASA MDP (Metrics Data Program) repository. This study takes only twelve datasets from the NASA facility website for pre-processing. The set of available static code features includes LOC, Halstead, and McCabe complexity metrics and it is related to the number of linearly independent paths through a program.

III.IV Mathematical modeling of FACNB

Fuzzy Naive Bayes method computes conditional class probabilities and then predict the most probable class of a vector of training data $X = \{X_1, X_2, ..., X_n\}$, according to sample data D.

The functions in Naïve Bayes are computed as follows:

$$P(w_i/X) = [P(X/w_i)P(w_i)\mu_i(X)]/P(X) \quad (1)$$

Where, P is the probability distribution, X be the no. of input training data sets and w be the weightage of the classifiers.

This Naive Bayes method assumes conditionally independent among the events in X. It modifies the equation (1) to:

$$P(w_i/X) = (1/S)P(w_i)\prod_{k=1}^{n}[P(X_k/w_i)\mu_i(X)] \quad (2)$$

Then, the classification rule for Fuzzy Naive Bayes ***[29]*** is done by:

$$X \in w_i if : P(w_i/X) > P(w_j/X), \quad for \quad all\, i \neq j \quad and\, i, j \in \Omega \quad (3)$$

Where, $P(w_i/X) = P(w_i/X_l = A_{li}, ..., X_n = A_{ni})$ by equation (2) and A be the attributes and i & j be the instances.

As proposed, it consists of a hybrid classifier bringing together Fuzzy Set Attributes and cluster Theory and a Naive Bayes classifier, termed as Fuzzy Attribute Cluster Naive Bayes classifier,

$$FNBclassify(a) = \arg\max_{c \in C} P(c) \sum_{x_1 \in X_1} P(x_1|c)\mu_{x_1} ... \sum_{x_n \in X_n} P(x_n|c)\mu_{x_n} \quad (4)$$

Where $\mu_{x_i} \in [0,1]$ denotes a membership function or degree of truth of attribute $x_i \in X_i$ in a new example a. To be conservative, it is required that all degrees of truth are normalized in the current variable assignation, in this case $\sum_{x_i \in X_i} \mu_{x_i} = 1$ ***[23]***.

The probabilities for equation (4) can be calculated as below,

$$P(C = c) = \frac{(\sum_{e \in L} \mu_c^e) + 1}{|L| + |D(C)|} \quad (5)$$

$$P(X_i \in x_i) = \frac{(\sum_{e \in L} \mu_{x_i}^e) + 1}{|L| + |D(X_i)|} \quad (6)$$

$$P(X_i = x_i | C = c) = \frac{(\sum_{e \in L} \mu_{x_i}^e \mu_c^e) + 1}{(\sum_{e \in L} \mu_c^e) + |D(X_i)|} \quad (7)$$

where L is the training set consisting of all examples $e = \{X_1 = x_1, ..., X_n = x_n, C = c\}$,

$\mu_c^e \in [0,1]$ denotes the degree of truth of c ∈ C in an example e ∈ L, and

$\mu_{x_i}^e \in [0,1]$ is the membership of attribute $x_i \in X_i$ in such example.

All degrees of truth must be normalized such that $\sum_{c \in C} \mu_c^e = 1$ and $\sum_{x_i \in X_i} \mu_{x_i}^e = 1$.

Laplace-correction is applied to compute the probabilities.

III.V Experimental Setup

In this method the original versions of the data sets from the NASA MDP Repository have been used.

III.V.I Initial pre-processing

In this method binary classification is done which involves the binarization of class variable and removal of module identifier and extra error data attributes. Since the 'unique module identifier' attribute is not having any information to assess the defectiveness of a module it is removed. Any other error-based attributes are also required to be removed to ensure correct classification.

III.V.II Removal of constant attributes

The database prepared for learning should be based on attributes which show variance at different instances. The numeric attributes which have a constant/fixed value throughout all instances will have a variance of zero. Since these attributes contain no information with which to discern modules apart, will only be a waste of classifier resources. During this process there may be removal of data that may be correct, but in the context of machine learning it is of no use and is therefore discarded. This data may be valid, but after the data divide into training and testing set, it may be that the training data contains a constant attribute.

III.V.III Removal of repeated attributes

To check overrepresentation of any attribute repeated attributes i.e two or more attributes having identical values

for each instance also need to be removed as such attributes are fully correlated. Among the NASA data sets there are two repeated attributes (post stage 1), namely the 'number of lines' and 'LOC total' attributes in data set KC4. The difference between these two metrics is poorly defined at the NASA MDP Repository. During this process again, there may be removal of data that may be correct because it is done so because it can be problematic when data mining.

III.V.IV Replacement of missing values

Depending on the classification method used missing values may or may not be needed to be corrected. Simplest way of dealing with missing values within the NASA data sets is that all instances which contained missing values within the NASA data sets are discarded. It is always better to cleanse data than to remove it in order to maximize the quantity of possible information to learn from. This stage adds data via the replacement of missing values, because they are problematic for many learning techniques. If a learning method that is resilient to missing values (as in the case of naive Bayes) some researchers may not wish to carry out this stage.

III.V.V Enforce integrity with domain-specific expertise

Varied quantities of attributes derived from simple equations of other attributes, are contained in NASA data sets which are useful for checking data integrity. It is also possible to use domain-specific expertise to validate data integrity just by searching for theoretically impossible occurrences.

IV. RESULTS AND DISCUSSIONS

A Simulation software for the proposed FACNB algorithm is developed using JAVA/J2EE.

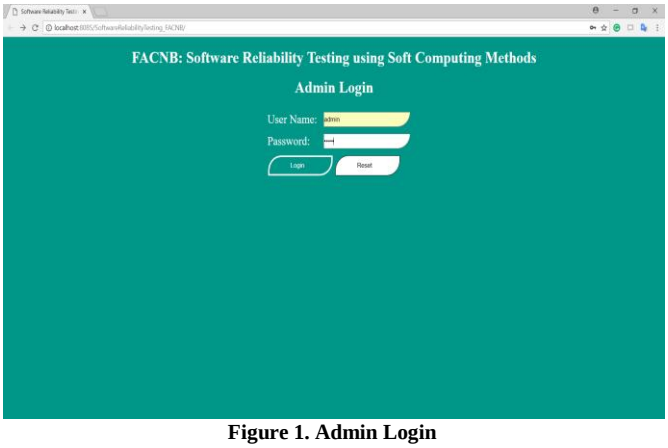

Figure 1. Admin Login

Figure 1 shows the admin login page. In the background, this page values will validate using adminlogin.java. If valid it redirects to the admin page. Otherwise, it shows error then redirected to the login page.

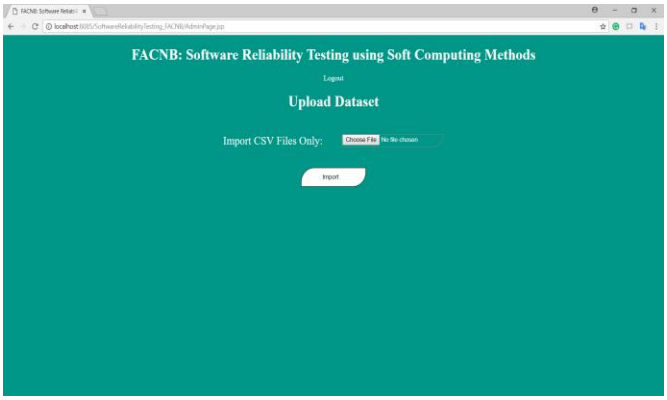

Figure 2(a) Upload dataset screen

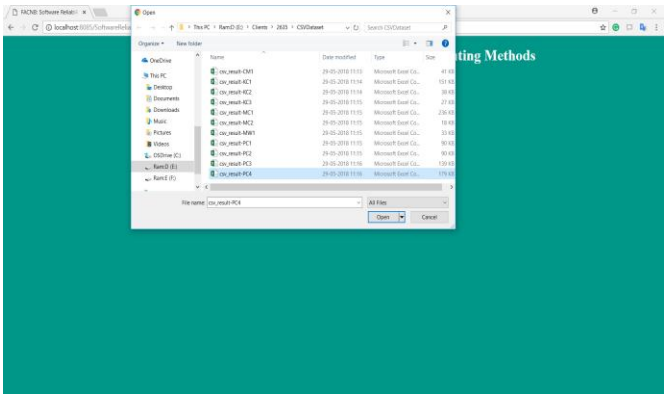

Figure 2(b) Upload dataset: File selection

Figure 2(a) and 2 (b) illustrates the upload benchmark dataset which was collected from 12 NASA datasets to demonstrate the effectiveness of the software reliability.

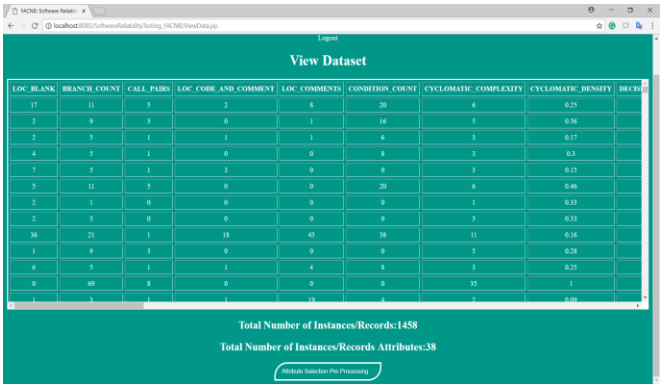

Figure 3. View dataset

Figure 3 displays the pre-processing step of the obtained dataset. Each observation in the dataset consists of a unique ID, error count and various static code features. This source file is input to the parser for computing the metrics. From this, the source code is verified for the lexical phase. The above-mentioned static code features were mined from the user source code by automated methods.

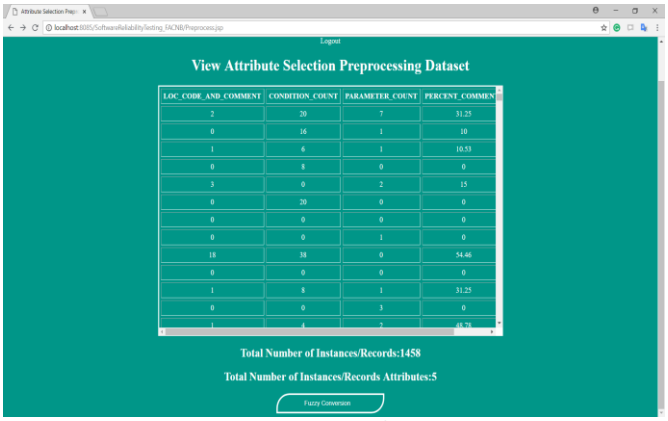

Figure 4. Attribute selection for pre-processing

Figure 4 demonstrates the attribute selection process. After the process of mining, the subsequent file is stored in an attribute relationship as a file format. This type of file is read as the test set file for each source code. Here, 38 attributes are crisp as 5 attributes to consume the buffer memory to proceed further.

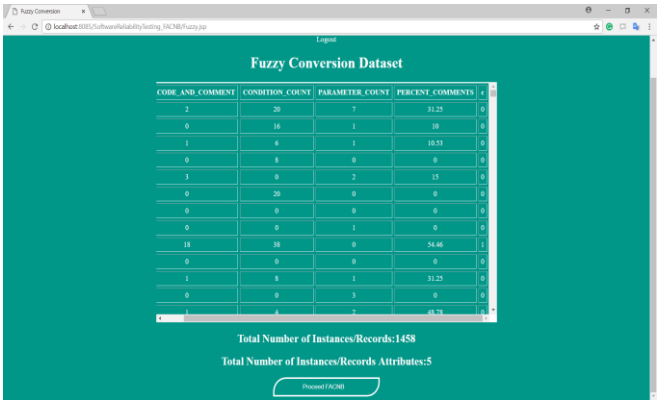

Figure 5. Fuzzy Conversion

Figure 5 shows the fuzzy conversion of the dataset to identify the software faults accurately. This fuzzy system improves the accuracy by training the dataset using NN.

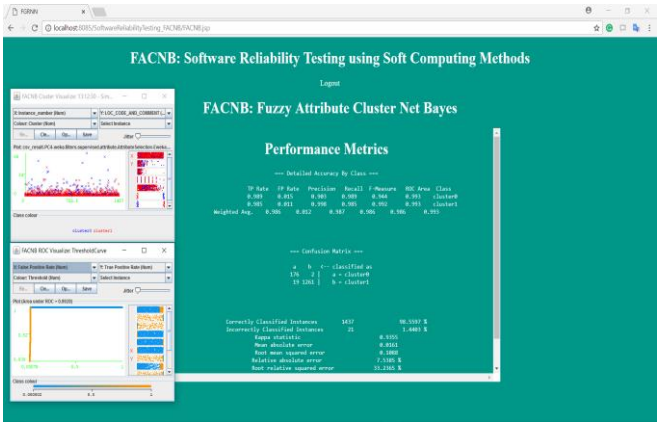

Figure 6. Fuzzy Attribute Cluster Net Bayes

Figure 6 explains the FACNB algorithm techniques to identify the detailed accuracy results.

IV.I Performance metrics

The main aim of most classifiers is to perform binary classification, i.e., Faulty or Non-Faulty. The performance metrics used are accuracy, confusion matrix, Area under the ROC curve.

Accuracy

The accuracy of the classifier means to correctly predict the class label of new or unseen data. Accuracy is percentages of the testing set example correctly classified into class.

The area under ROC curve (AUC)

For measure the area under the curve Receiver operating characteristics (ROC) is the plot. Receiver operating characteristics (ROC) curve is a graphical representation of the performance of the binary classifier. The curve is created by true positive rate against the false positive rate at the different threshold value. AUC is given a better result for software defect detection.

Confusion matrix

The confusion matrix is a specific table that is used to measure the performance of machine learning algorithm. Table 1 shows an example of a generic confusion matrix. Each row of the matrix represents the instances in an actual class, while each column represents the instance in a predicted class or vice versa. Confusion matrix used for measure the performance of the classifier. Confusion matrix has for basic categories which are True positive, True Negative, False Positive, False Negative.

Table 1. Confusion matrix

		Actual class	
		Fault	Non-fault
Predicted results	Fault	176	2
	Non-fault	19	1261

The Performance measures are evaluated by the following equations,

$$\text{Recall} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \quad (1)$$

$$\text{False Positive Ratio (FPR)} = \frac{\text{false positive}}{\text{false positive} + \text{true negative}} \quad (2)$$

$$\text{Precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \quad (3)$$

$$\text{True positive rate (TPR)} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \quad (4)$$

$$\text{F1 measure} = 2.\frac{\text{precision.recall}}{\text{precision}+\text{recall}} \quad (5)$$

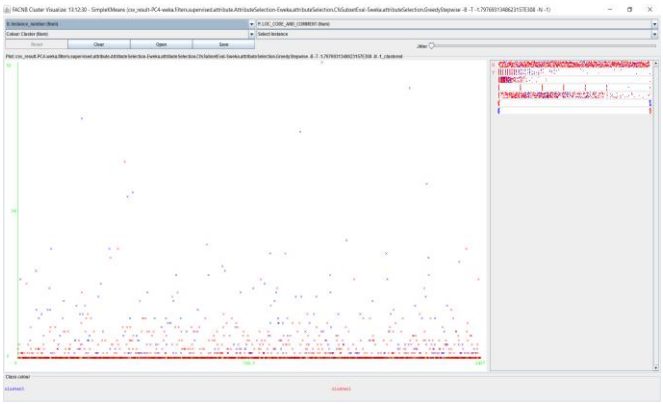

Figure 7. Cluster visualization graph

Figure 7 shows the clustering process in the datasets. It is done by the novel FACNB algorithm.

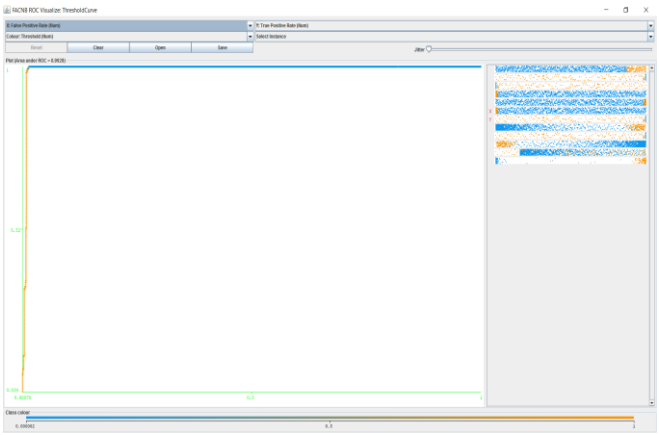

Figure 8. ROC curve graph

Figure 8 displays the region of curve presented in the cluster dataset.

Finally, the correct and incorrect instances were calculated by using the following formulae

$$\text{Relative Error}(\%) - \text{RE}_i = \left(\frac{|(P_i - A_i)|}{A_i}\right) * 100 \quad (6)$$

$$\text{Average Relative Error}(\%) = 1/n \sum^n RE_i \quad (7)$$

$$\text{Mean Absolute Error} = [\sum^n |P_i - A_i|]/n \quad (8)$$

Where, P_i =Predicted Value, A_i =Actual Value and n=total no of observations/patterns

Table 2: Performance Metrics

	TP rate	FP rate	Precision	Recall	F-measure	ROC area	Class
	0.989	0.015	0.903	0.989	0.944	0.993	Cluster 0
	0.985	0.011	0.998	0.985	0.992	0.993	Cluster 1
Weighted Average	0.986	0.012	0.987	0.986	0.986	0.993	

The overall calculation results are shown in Table 3 below, which includes both correct and incorrect instances. There are totally 1458 instances is taken in the attributes.

Table 3. Calculation results

Correctly classified instances (1437)	98.56 %
Incorrectly classified instances (21)	1.44 %
Kappa statistic	93.55 %
Mean absolute error	0.16 %
Root mean square error	1.089 %
Relative absolute error	7.53 %
Root relative squared error	33.23 %

The comparison of the various clustering algorithm is shown in table 4 below.

Table 4. Comparison table

Evaluation parameter	Fuzzy C-means clustering [18, 19]	Adaptive Neuro-fuzzy technique [20]	Novel FACNB algorithm
Accuracy	75	87	99.3
Net reliability	60.07	47.20	35.23
Mean Absolute Error (MAE)	0.25	0.13	0.16
Root Mean Squared Error (RMSE)	0.0833	0.0194	1.089

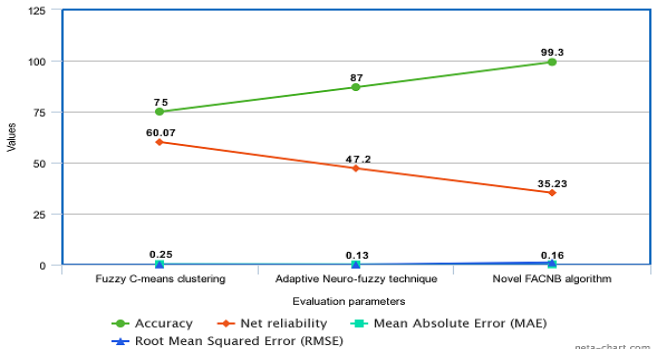

Figure 9. Comparison graph of evaluated parameters

Table 4 and Figure 9 shows the comparison between the existing and proposed method to validate the performance of the proposed FACNB algorithm. The parameters such as accuracy, net reliability, MAE, RMSE is compared with the projected values. This method gives a higher accuracy of 99.3 % as compared to other methods.

Table 5. Comparison of ROC curve

Algorithm	ROC curve value
Naïve Bayes [21]	0.750
J48 [21]	0.745
Random Tree [22]	0.5842
Classification and Regression Tree (CART) [22]	0.6048
Bayesian Logistic Regression [22]	0.5119
Proposed FACNB	0.9928

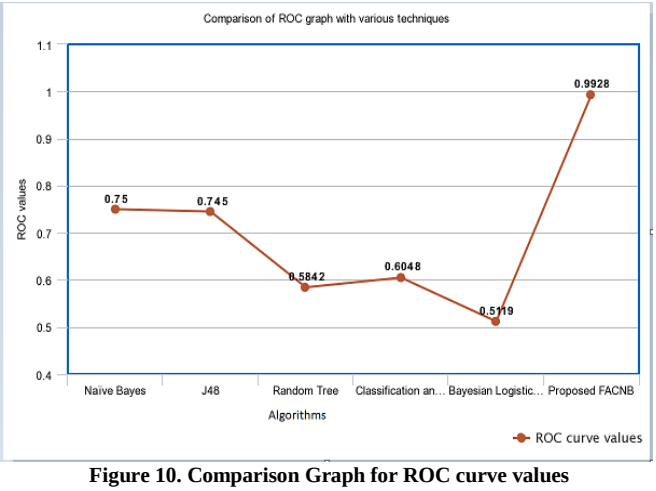

Figure 10. Comparison Graph for ROC curve values

Table 5 and Figure 10 represents the comparison of ROC curve values with various existing algorithms. Here, the ROC curve values for the proposed FACNB algorithm gives higher value as 0.9928. The minimum value present in the curve is 0.75 which comes under Naïve Bayes algorithm, for Random Tree Algorithms it is 0.584.So,it is evident that the proposed method provides higher accuracy than other methods.

V. CONCLUSION

Various models using clustering technique, machine learning etc have earlier been proposed for several software fault prediction techniques but their accuracy is limited. The existing fuzzy C_means clustering method and Adaptive Neuro-fuzzy techniques are best for modeling fault proneness prediction in a software system as shown in table 4. In this study, a novel machine learning algorithm called Fuzzy Attribute Cluster Net Bayes (FACNB) is proposed. The results from the experimental setup lead to the conclusion that the proposed FACNB model improves the accuracy in predicting reliability of software as compared to other methods on various parameters as shown in table 4 and 5.

REFERENCES

[1] G. R. Choudhary, S. Kumar, K. Kumar, A. Mishr & C. Catal., *"Empirical analysis of change metrics for software fault prediction"* Journal of Computers and Electrical Engineering, Vol. 67, pp. 15-24, 2018.

[2] D. Amara & L. B. A. Rabai, *"Towards a New Framework of Software Reliability Measurement Based on Software Metrics"* Procedia Computer Science Vol. 109C, pp. 725–730, 2017

[3] R. Rana, M. Staron, C. Berger, J. Hansson, M. Nilsson & W. Meding, *"Analyzing Defect Inflow Distribution and Applying Bayesian Inference Method for Software Defect Prediction in Large Software Projects"*, Journal of Systems & Software, pp. 1-29, 2016

[4] E. Erturk & E. A. Sezer, *"Iterative software fault prediction with a hybrid approach"* Journal of Applied Soft Computing Vol. 49, pp. 1020–1033, 2016.

[5] P. Kumar & Y. Singh, *"A study on software reliability prediction models using soft computing techniques"*, Journal of Information and Communication Technology, Vol. 5(2), pp. 187-204, 2013

[6] J. Wang & C. Zhang, *"Software Reliability Prediction Using a Deep Learning Model based on the RNN Encoder-Decoder"*, Journal of Reliability Engineering and System Safety, pp. 1-17, 2018

[7] N. P. Padhy, R. P. Singh & S. C. Satapathy, *"Software reusability metrics estimation: Algorithms, models and optimization techniques"*, Journal of Computers and Electrical Engineering, pp.1-16, 2018

[8] P. Roy, G. S. Mahapatra and K. N. Dey, *"Neuro-genetic approach on logistic model-based software reliability Prediction"*, Journal of Expert Systems with Applications, pp. 1-10, 2015

[9] F. Febrero, C. Caler and M. Ángeles Moraga, *"Software reliability modeling based on ISO/IEC SQuaRE"*, Journal of Information and Software Technology, Vol. 70, pp. 18-29, 2016

[10] M. Zhu, X. Zhang & H. Pham, *"A comparison analysis of environmental factors affecting software reliability"*, Journal of Systems and Software, Vol. 109, pp. 150-160, 2015.

[11] N. Jazdi, N. Oppenlaender & M. Weyrich, *"Quantification of the quality characteristics for the calculation of software reliability"*, IFAC-PapersOnLine, Vol. 49-30, pp. 001–005, 2016
[12] D. Srdjana, C. Stipe & T. Mili, *"Bayesian Network Model for Task Effort Estimation in Agile Software Development",* Journal of Systems and Software, pp. 1-16, 2017
[13] O. F. Arar & K. Ayan, *"A Feature Dependent Naive Bayes Approach and its Application to the Software Defect Prediction Problem"*, Journal of Applied Soft Computing, pp. 1-39, 2017
[14] I. Lakshmanan & S. Ramasamy, *"An artificial neural network approach to software reliability growth Modeling"*, Procedia Computer Science, Vol. 57, pp. 695 – 702, 2015.
[15] E. U. Warriach, K. Tei, *"A comparative analysis of machine learning algorithms for fault's detection in wireless sensor networks"*, International Journal of Sensor Networks, Vol. 24(1), pp. 1-13, 2017
[16] J. Nagpal & A. Khuteta, *"Software Fault Estimation using Fuzzy C-Means and Neuro-Fuzzy Classification"*, Journal of Digital Application & Contemporary Research, Vol. 2(10), pp. 1-8, 2014
[17] Sankar, Kannan & Jennifer, *"Prediction of Code Fault Using Naive Bayes and SVM Classifiers"*, Journal of Scientific Research, Vol. 20 (1), pp. 108-113, 2014
[18] Pushpavathi, Suma & Ramaswamy, *"Analysis of Software Fault and Defect Prediction by Fuzzy C-Means Clustering and Adaptive Neuro-Fuzzy C-Means Clustering",* Journal of Scientific & Engineering Research, Vol. 5(9), pp. 292-297, 2014
[19] M. Patel, *"A Survey on Software Fault Prediction Technique based on Clustering Algorithm and Artificial Intelligence"* International Journal of Innovative Research in Technology, Vol. 2 (7), pp. 623-627, 2015.
[20] S. S. Maddipati, Pradeepini, & Yesubabu, *"Software Defect Prediction using Adaptive Neuro Fuzzy Interference System"*, International Journal of Applied Engineering Research, 13 (1), 394-397, 2018
[21] Y. Jiang, B. Cukic, & Y. Ma, *"Techniques for evaluating fault prediction models"*, Empirical Software Engineering, Vol. 13(5), pp. 561–595, 2008
[22] R. Sehgal., & D. Mehrotra, *"Analysis of Software Fault Prediction Metrics"*, World Applied Sciences Journal, Vol. 32(3), pp. 368-378, 2014
[23] S. K. Pandey, R. B. Mishra, A. K. Tripathi, *"Software Bug Prediction Prototype using Bayesian Network Classifier: A Comprehensive Model"* Procedia Computer Science, 2018
[24] C. Bustamante, L. Garrido, R. Soto, *"Fuzzy Naïve Bayesian Classification in RoboSoccer 3D: A Hybrid Approach to decision making"*, Lecture notes in Computer Science, Vol. 4434, pp. 507-515, Springer Verlag, 2007.
[25] A. Hammouri, M. Hammad, M. Alnabhan, F. Alsarayrah, *"Software Bug prediction using Machine Learning approach"* International Journal of Advanced Computer Science and Applications, Vol. 9(2), pp. 78-83, 2018
[26] D. Karel, T. Verbraken & B. Baesene, *"Towards comprehensible Software Fault Prediction Models using Bayesian Network Classifiers"*, IEEE Transactions on Software Engineering, Vol 39(2), pp 237-257, 2013
[27] A. M. Mansour Mansour, *"Decision tree-based expert system for adverse drug reaction detection using Fuzzy Logic and Genetic algorithm"*, International Journal of Advanced Computer Research, Vol. 8(36), pp. 110-128, 2018
[28] K. Gusmanov, *"On the adoption of neural networks in modeling software reliability"*, Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering – ESEC/SIGSOFT FSE, Lake Buena Vista, Florida, USA, pp. 962-964, 2018
[29] R. Marcos de Moraes, L. S. Machado, *"Simultaneous assessment of teams in collaborative virtual environments using Fuzzy Naïve Bayes"* Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS 2013), Edmonton Canada, pp. 1343-1348, 2013
[30] R. Peng, Y. Li, Y. Liu, *"Software Fault Detection and Correction: Modeling and Applications"*, Springer Nature America Inc., Springer Verlag, 2018
[31] A. Azadeh, P. Pourreza, M. Saberi, O. Khadeer Hussain, E. Chang, *"An integrated fuzzy cognitive map-Bayesian network model for improving HSEE in energy sector"*, IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2017), pp. 1-7, 2017
[32] I. M. D. Aguila, *"Requirement Risk level forecast using Bayesian Network Classifiers"*, International Journal of Software Engineering and Knowledge Engineering, 2011
[33] S. Kumar, S. S. Rathore, *"Software Fault Prediction: A roadmap",* Springer Singapore, 2018
[34] Neeta Rastogi, Shishir Rastogi, Manuj Darbari, *"Survey on Software Reliability Prediction using Soft Computing"* International Journal of Computer Engineering and Technology, Vol. 9(4), pp. 212-216, 2018
[35] N. K. Rao, R. M. Reddy, B. K. Rao, *"Defect prediction in software entities classified in terms of level dependencies"*, International Journal of Scientific Research in Computer Sciences and Engineering, Vol. 1(1), pp. 20-25, 2013
[36] T. Senthilselvi, R. Parimala, *"Improving clustering accuracy using Feature Extraction Method"*, International Journal of Scientific Research in Computer Sciences and Engineering, Vol. 6(2), pp. 15-19, 2018

Authors' Profile

Mrs. Neeta Rastogi has done her B. E. in Computer Engineering from North Maharashtra University in 1997.Currently she is pursuing Ph.D from BBD University, Lucknow (India). She has 3 years of industry experience and 19 years of academic experience. She is presently a senior faculty member in Department of Computer Science and Engineering, BBD National Institute of Technology and Management. Her area of interest are Software Engineering, Software Project and Quality Management, Distributed Systems, Mobile Computing and Wireless Networks.

Mr. Shishir Rastogi has done his B. E. in Computer Engineering from North Maharashtra University in 1996 .Currently he is pursuing Ph.D from BBD University, Lucknow (India). He has a total of 23 years of industry and academic experience. He is presently a senior faculty member in School of Computer Applications, BBD Univeristy. His area of interest are Data Communication and Computer Networks, Pervasive Computing, Distributed Systems, Mobile Computing, Wireless Sensor Networks, Ad-hoc Networks, Software Project and Quality Management, Microprocessors and Computer organization.

Dr. Manuj Darbari is currently Professor in Department of Computer Science and Engineering, School of Engineering, BBD University having around 22 years of industrial and academic experience. He has done his B. Tech. in Electronics from Amrawati University in 1993, M. Tech. in Computer Science and Engineering from MNNIT, Allahabad in 1995 and Ph. D from BIT, Mesra (Ranchi) in 2011. He is an eminent researcher and has guided many research scholars so far. He is a senior member of IEEE, IETE, IEI, CSI and AIMA. He is also on the Board of Reviewers for many of the reputed international journals of Computer Science & Engineeing as well as Management. His major areas of interest include Software Engineering, Agile Software Development, Software Agents, Software Quality and Reliability, Software Project Management, Information Architecture, Ubiquitous Computing, Wireless Communications and Wireless Sensor Networks etc.

