COMPARISON OF PROPOSED NEURO EVOLUTION ALGORITHM WITH VARIOUS CLASSIFIER ALGORITHM ON SOFTWARE DEFECT PREDICTION METHOD

Mr. T. Murali Krishna¹, Dr. T. Nalini²

¹Research Scholar, Dept. of CSE, Bharath Institute of Higher Education and Research (BIHER), Chennai
²Professor, Dept. of CSE, Dr MGR Educational Research and Institute, Chennai

ABSTRACT

Software defect identification is a fundamental and active research question in software engineering. Researchers have proposed various defect identification approaches. It becomes one of the important issues in the field of software engineering research in recently. As machine learning methods like, supervised learning and unsupervised learning techniques are widely used in the prediction since of its high accuracy and simple calculation. Aim of the work is to propose a neuro evolutionary algorithm that can improve the performance of the software defect prediction. The performance of the algorithm is compared with existing algorithm such as SVM, KNN and Decision Tree algorithm. Various software defect prediction techniques are executed using neuro evolutionary algorithm and its results are compared. The result demonstrates that the neuro evolutionary algorithm optimizes the performance and obtains better accuracy than existing algorithm.

Keywords: Software defect prediction, Neuro evolutionary algorithm, SVM, KNN, Decision Tree.

I. INTRODUCTION

Software bug prediction method plays a significant role in improving software quality, dropping development time of software’s, and as well as reducing testing expenses. It becomes one of the important issues in the field of software engineering research in recently. As machine learning methods like, supervised learning and unsupervised learning techniques are widely used in the prediction since of its high accuracy and simple calculation. However, the imbalanced classification problem of data sets is common in practical applications and seriously affects the performance of learning algorithm. In this Study we propose a better optimization of performance by using hybrid neuro evolution algorithm. The goal is to constructive insights based on which one can further advance this particular research area. The Research Question 1 is to proposed software bug prediction model brings benefits in performance than the existing model. Where the Research Question 2 is to parameterise the space, it is inspiring to find the optimal configuration with a limited budget of computational cost. Therefore, the proposed optimization technique can be conducted by using three possible types of methods, where by varying the datasets the first methods, we implemented tradition software prediction using our proposed model. In the second approach cross-project bug prediction method is implemented as it can able to solve the problem of lack in dataset and in the third approach is just in time software bug prediction.

II. LITERATURE REVIEW

A. Hall, (1998), method is applied by utilizing WEKA apparatus M. Hall et al (2009). A similar element determination method has been applied by various papers. As Menzies et al. [2009] expressed, the significance of the classifier is more than that of highlight determination, so include choice method isn't the fundamental focal point of this investigation. Too, some other element choice strategies, for example, Principal Component Analysis (PCA) and Info Gain Attribute Evaluation were utilized in tests, however results from CFS were the best. McCulloch and Walter Pitts (1997) the elements of a human mind examined, proposed a model of computing element called McCulloch and Pitts which present weighted whole of the inputs to the component utilized by limit rationale activity. Mix of a portion of these figuring components was utilized to understand a few.
coherent calculations. Hebb in 1949 proposed a law turns into a crucial learning in neural system writing. The perceptron model was proposed by Rosenblatt in 1958. Windrows and Hoff proposed for figuring component an ADALINE (versatile direct component) model and LMS (least mean square) learning calculation is utilized to change the loads of an ADALINE model in 1960. Neural Network research began after the distribution of AI research paper by Marvin Minsky and Seymour in 1969. They found two key issues utilizing the computational machines that prepared Neural Networks. With numerical documentation, hardware additionally portrayed by Rosenblatt, for example, the circuit, simply after the back-spread calculation a circuit whose scientific calculation can be handled, it was proposed by Paul Werbos in (1975). Under the name connectionism the equal appropriated preparing of the mid-1980s got well known. The announcement by David E. Rumelhart and James McClelland (1986), gave a full work in PCs to reproduce neural rely upon the utilization of connectionism organize forms. In computerized reasoning we utilized the Neural Networks. During the 1990s, Neural Networks were gotten progressively mainstream in AI as a result of help vector machines and other. Restored enthusiasm for the 2000s in Neural Networks was started as a result of the appearance of profound learning. In CMOS, Computational gadgets have been made, to both neuralomorphic figuring and biophysical reproduction. In the computational gadgets had large improvement from 2006. In the examination gathering, current neural systems and profound feed forward neural systems created by Jürgen Schmidhuber Between 2009 and 2012, at the Swiss AI Lab which have won in design acknowledgment and AI eight global rivalries. CNN is a neural system with various layers and is based on the creature visual cortex. The first CNN was created by LeCun et al. (2007). Application regions of CNN incorporate chiefly picture handling and manually written character acknowledgment for example postal code translation. Thinking about the engineering, prior layers are utilized for recognizing the highlights, for example, edges what's more, the later layers are utilized for the recombination of highlights to shape elevated level characteristics of the info followed by the classification. At that point pooling will be done, which mitigates the dimensionality of the removed highlights. The subsequent stage is to perform convolution and afterward once more pooling, that is taken care of into a totally connected multilayer perceptron. Obligation of the finishing up layer called a yield layer is to perceive the highlights of the picture by utilizing backpropagation calculations. In CNN, the benefit of profound layers of preparing, convolutional layer, pooling, and a completely associated classification layer uncovers different applications for example, discourse acknowledgment, clinical applications, video acknowledgment and different errands of characteristic language handling. CNN delivers better exactness and improves the presentation of the framework because of its restrictive highlights, for example, neighbourhood network and shared loads. It works much better than some other profound learning techniques. It is the most generally utilized engineering when contrasted with others. presents a Deep GA that seriously prepares deep neural networks for testing RL errands, and an encoding strategy that empowers productive dispersed preparing and a best in class minimal network encoding. We found that the GA is quick, empowering preparing Atari in 4h on a solitary work area or 1h disseminated on 720 CPUs. We archived that GAs are shockingly serious with famous calculations for deep support learning issues, for example, DQN, A3C, and ES, particularly in the difficult Atari area. We likewise demonstrated that intriguing calculations created in the neuroevolution network can now quickly be tried with deep neural networks, by indicating that a Deep GA-fueled oddity search can fathom a tricky Atari-scale game. It will be intriguing to see future exploration research the potential and cutoff points of GAs, particularly when joined with different procedures known to improve GA execution. All the more for the most part, our outcomes proceed with the story began by backprop and reached out with ES that old, basic calculations in addition to present day measures of calculation can perform incredibly well.

### III. TRADITIONAL SOFTWARE BUG PREDICTION TECHNIQUE.

Three approaches are available to evaluate the software defect prediction model. In this work all the three approaches are considered for evaluating the proposed defect prediction model. (i) With-in Project software fault prediction, (ii) Cross-Project software fault prediction, (iii) Just in time software fault prediction.

The prediction model is constructed and various SDP datasets are applied to evaluate the proposed model. Historical data of similar type of software projects are collected and applied to the proposed model therefore to train and analyse the performance of with-in project defect prediction. Software bug prediction model is constructed trained based on the historical data. The main limitations of within project software fault prediction is, it requires large amount of historical data. It is not always possible to have sufficient amount of historical data. To handle this situation, the second type of software fault prediction method is proposed namely cross project software fault prediction. In this method any trained software fault prediction model can be applied to other project. But the main limitation of this method is the anticipated project should have alike metric sets, the implications of the metric sets considered should be equal among all the considered projects. Therefore, it is
some time difficult to relate projects of unrelated dataset. The third type of software fault prediction technique deals with just in time software fault prediction.

### 3.1 Proposed Algorithm

A hybrid Neuro evolutionary algorithm is proposed as a research method. This proposed method consists of extreme learning algorithm applied with evolutionary algorithms; therefore, the algorithm can help in optimizing the performance in predicting the defects in order to obtain the best trained model the learning curves of several machine learning algorithms are analysed, and the performance of the derived models are evaluated using traditional metrics-based approaches. The proposed evolutionary algorithm consists of several components which includes selection of parent, crossover function mutation function, survival, and finally the condition for termination. An evolutionary algorithm (EA) is a subset of evolutionary computation, a conventional populace-based metaheuristic streamlining calculation. An EA utilizes components propelled by organic advancement, for example, generation, change, recombination, and choice. Up-and-comer answers for the enhancement issue assume the part of people in a populace, and the wellness work decides the nature of the arrangements. Advancement of the populace at that point happens after the rehashed use of the above administrators. [1-5]

Evolutionary algorithms regularly perform well approximating answers for a wide range of issues since they in a perfect world don't make any supposition about the basic wellness scene. Strategies from developmental calculations applied to the displaying of natural advancement are commonly restricted to investigations of microevolutionary cycles and arranging models dependent on cell measures. In most genuine utilizations of EAs, computational intricacy is a restricting factor actually, this computational unpredictability is because of wellness work assessment. Wellness guess is one of the answers for conquer this trouble. Not with standing, apparently straight forward EA can take care of frequently complex issues in this way, there might be no immediate connection between calculation unpredictability and issue intricacy. Genetic programming (GP) is a method of developing projects, beginning from a populace of ill suited (generally arbitrary) programs, fit for a specific assignment by applying activities comparable to characteristic hereditary cycles to the number of inhabitants in programs. It is basically a heuristic inquiry strategy regularly portrayed as 'slope climbing’, for example looking for an ideal or if nothing else appropriate program among the space all things considered.[6-10]

The activities are: determination of the fittest projects for propagation (hybrid) and change as per a predefined wellness measure, typically capability at the ideal assignment. The hybrid activity includes trading irregular pieces of chosen sets (guardians) to deliver new and diverse posterity that become aspect of the new age of projects. Transformation includes replacement of some irregular aspect of a program with some other arbitrary aspect of a program. A few projects not chose for multiplication are duplicated from the current age to the new age. At that point the choice and different tasks are recursively applied to the new age of projects.[16-18]

### IV. EXPERIMENT SETUP

In this experiment we implemented an Neuro evolutionary algorithm where extreme learning algorithm are applied with evolutionary algorithms, therefore the algorithm can help in optimizing the performance in predicting the defects in order to obtain the best trained mode the aims the learning curves of several machine learning algorithms are analysed, and the performance of the derived models are evaluated versus three traditional metrics-based approaches with five existing algorithms like SVM, NB, c4.5,KNN, and LR[11-15]

Traditional methods are (i) With in Project software fault prediction, (ii) Cross-Project software fault prediction, (iii) Just in time software fault prediction.

Existing Classification are SVM, NB, c4.5,KNN, and LR and compare with proposed method Neuro evolutionary algorithm.

Measure metrics are Recall, F-measure, G-mean, precision and accuracy

### V. RESULTS

In this table1 shown the accuracy of five classifier with the proposed model of first traditional method of within Project software fault prediction.
Fig 1 Comparison of accuracy of various algorithm for Traditional Software Bug Prediction Technique.

The above figure 1 shows that the accuracy of the proposed method is significantly higher and it demonstrates the better performance while comparing other machine learning algorithms such as SVM, Decision tree algorithm, C4.5, NB and KNN, while applied to Traditional Software bug Prediction technique. Datasets are collected from various sources and we have applied various datasets of same time and the experiments proves that the proposed model optimized and obtained better accuracy rate.
The above figure 2 shows that the accuracy of the proposed method is significantly higher for all types of dataset and it also demonstrates the better performance while comparing other machine learning algorithms applied to Cross-Project Software bug Prediction technique. Datasets are collected from various sources and we have applied various datasets of different types, therefore to avoid lack in dataset availability and the experiments proves that the proposed model optimised and obtained better accuracy rate.

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Proposed Model</th>
<th>SVM</th>
<th>NB</th>
<th>C4.5</th>
<th>LR</th>
<th>kNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnkiDroid</td>
<td>0.88</td>
<td>0.79</td>
<td>0.88</td>
<td>0.85</td>
<td>0.79</td>
<td>0.85</td>
</tr>
<tr>
<td>BankDroid</td>
<td>0.84</td>
<td>0.64</td>
<td>0.66</td>
<td>0.65</td>
<td>0.60</td>
<td>0.64</td>
</tr>
<tr>
<td>BoardGameGeek</td>
<td>0.77</td>
<td>0.70</td>
<td>0.76</td>
<td>0.73</td>
<td>0.69</td>
<td>0.73</td>
</tr>
<tr>
<td>Chess</td>
<td>0.81</td>
<td>0.75</td>
<td>0.82</td>
<td>0.77</td>
<td>0.70</td>
<td>0.77</td>
</tr>
<tr>
<td>ConnectBot</td>
<td>0.88</td>
<td>0.69</td>
<td>0.73</td>
<td>0.72</td>
<td>0.66</td>
<td>0.71</td>
</tr>
<tr>
<td>Andlytics</td>
<td>0.81</td>
<td>0.80</td>
<td>0.87</td>
<td>0.85</td>
<td>0.79</td>
<td>0.84</td>
</tr>
<tr>
<td>FBreader</td>
<td>0.89</td>
<td>0.87</td>
<td>0.87</td>
<td>0.85</td>
<td>0.78</td>
<td>0.84</td>
</tr>
<tr>
<td>K9Mail</td>
<td>0.67</td>
<td>0.57</td>
<td>0.63</td>
<td>0.59</td>
<td>0.56</td>
<td>0.58</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>0.83</td>
<td>0.77</td>
<td>0.82</td>
<td>0.80</td>
<td>0.75</td>
<td>0.79</td>
</tr>
<tr>
<td>Yaic</td>
<td>0.89</td>
<td>0.74</td>
<td>0.82</td>
<td>0.79</td>
<td>0.74</td>
<td>0.79</td>
</tr>
<tr>
<td>Avg.</td>
<td>0.83</td>
<td>0.73</td>
<td>0.79</td>
<td>0.76</td>
<td>0.71</td>
<td>0.75</td>
</tr>
</tbody>
</table>

Table 3: Just-in Time software fault prediction

JUST-IN TIME SOFTWARE BUG PREDICTION
The above figure 3, Comparison of accuracy of various algorithm for Just-In Time Software Bug prediction shows that the accuracy of the proposed method is significantly higher for all types of dataset and it also demonstrates the better performance while comparing other machine learning algorithms applied to Just-In Time Software bug Prediction technique. Datasets are collected from various sources, as there are many changes in software development. The software bugs data are collected dynamically and they are applied as an input which proves that the proposed model optimised and obtained better accuracy rate.

5.1 Comparison of Other metrics with three classifier SVM, KNN, Decision tree with Proposed method Neural Evacuation Algorithm

The experimental results of JM1 data set are shown in fig4. The data in the tables retain four significant figures. The integrated learning Metrics; "T" in the table indicates TP Rate; "F" in the table indicates FP Rae; "f" indicates F-Measure; "M" indicates MCC; "U" indicates AUC; "C" indicates PRC. It can be seen from TABLE IV and Fig. 1 that there is no significant difference in the performance of each algorithm on the JM1 data set. Software defect prediction tends to pay more attention to whether defects can be fully predicted. That is, the relative accuracy index and recall are more noticeable. Among the eight prediction algorithms implemented this time, SVM, KNN, and Decision tree and the proposed model high recall rate of 0.646, 0.643, and 0.641, respectively. The AUC value reflects the classification ability of the algorithm. The Proposed model using hybrid neuroevolutionary algorithm Deep Belief Network (DBN) algorithms perform well in terms of prediction, reaching 0.710, 0.700, and 0.696, respectively, and their PRC values are also the best performance among the eight algorithms, 0.687, 0.701, and 0.682, respectively. Considering F-Measure value, ) is 0.630 and Proposed model 0.645 algorithm has similar Prediction ability, is better than 0.609 than decision tree algorithm. The top three MCC scores in the fight against imbalance classification are SVM, Decision tree and KNN, which are 0.300, 0.295 and 0.290, respectively, indicating that they have stronger imbalanced classification data adaptability.
The above figure 4 shows that the Recall, Precision, F-score, G-Mean value applying our proposed model for Traditional Bug Prediction Approach of the proposed method is significantly higher and it demonstrates the better performance while comparing other machine learning algorithms such as SVM, Decision tree algorithm and KNN, while applied to Traditional Software bug Prediction technique. Datasets are collected from various sources and we have applied various datasets of same time and the experiments prove that the proposed model optimised and obtained better prediction metrics rate.

Fig 5 Comparison of Recall, Precision, F-score, G-Mean value applying our proposed model for Cross-Project Bug Prediction Approach
The above figure 5 shows that the Comparison of Recall, Precision, F-score, G-Mean value applying our proposed model for Cross-Project Bug Prediction Approach of the proposed method is significantly higher for all types of dataset and it also demonstrates the better performance while comparing other machine learning algorithms such as SVM, Decision tree algorithm and KNN, while applied to Cross-Project Software bug Prediction technique. Datasets are collected from various sources and we have applied various datasets of different types, therefore to avoid lack in dataset availability and the experiments proves that the proposed model optimised and obtained better rate of recall, precision, F-score and G-mean value.

![Comparison of Recall, Precision, F-score, G-Mean value applying our proposed model for Cross-Project Bug Prediction Approach](image)

The above figure 6 Comparison of Recall, Precision, F-score, G-Mean value applying our proposed model for Traditional Bug Prediction Approach shows that the prediction metrics of the proposed method is significantly higher for all types of dataset and it also demonstrates the better performance while comparing other machine learning algorithms such as SVM, Decision tree algorithm and KNN, while applied to Just-In-Time Software Bug Prediction technique. Datasets are collected from various sources, as there are many changes in software development. The software bugs data are collected dynamically and they are applied as an input which proves that the proposed model optimised and obtained better accuracy rate.

**VI. CONCLUSION**

This study of software bug prediction premise reasonable for software engineering improvement through examination and exploration of software bug prediction innovation. Through the investigation of administered learning software bug prediction calculations, strategies for tackling imbalanced Prediction are dissected. Case based prediction calculation (LWL), choice tree-based prediction calculation, and gathering based learning prediction calculation Decision tree Prediction, SVM prediction calculation KNN, SVM-based prediction calculation, calculations were utilized for recreation tests and trial results were dissected. The after effects of the investigation show that there are contrasts in the performance of eight NASA datasets (JM1, KC3, and MC1), and outfit-based learning calculation Proposed calculation has solid capacities as far as characterization abilities. choice trees and BBN calculation performed well; the benefits of KNN and Decision Tree, SVM and the Proposed Model calculation in imbalanced arrangement prediction are evident and the proposed model have great performance regarding recall rate.

**REFERENCES**


www.turkjphysiotherrehabil.org
14 Zimmermann, Thomas, et al. (2009) "Cross-project defect prediction: a large scale experiment on data vs. domain vs. process." Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering. ACM,