2021 2nd International Conference on Intelligent Engineering and Management (ICIEM) | 978-1-6654-1450-0/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICIEM51511.2021.9445350

Multi-Layer Perceptron Neural Network with Feature Selection for Software Defect Prediction

J. Mary Catherine
Department of Computer Science
Chevalier T Thomas Elizabeth College for Women
Chennai, India
catherine_vinod@yahoo.co.in

S Djodilatchoumy
Department of Computer Science and Applications
Pachaiyappa's College
Chennai, India
djodilatchoumy@hotmail.com

Abstract— **Software is continuously evolving and hence it is essential for the production of quality and stable software by every software provider. Recently there is a paradigm shift in how software is designed. One of the biggest challenges of software engineering is predicting defects in software modules, to save quality testing time. As software development challenges and constraints rise, unexpected effects such as failure and errors decrease the consistency of software and user loyalty, rendering error-free software more complex and frustrating. In this paper, we analyze the use of Multi-Layer Perceptron Neural Network [5] (MLP-NN) for the efficient prediction of defects. We have also executed the MLP-NN with a subset of features selected using popular feature selection methods. The model was evaluated on 5 datasets from the AEEEM dataset. The results were compared with other common classifiers like Logistic Regression, MLP-NN, and Random Tree. The findings indicate that feature selection has a major role in increasing the accuracy of prediction. Our model had higher accuracy in few cases while at par with others in some.**

Keywords - defect prediction, bug metrics, defects, multilayer perceptron, neural networks, feature selection, regression, random tree.

I. Introduction

The Software testing industry is currently undergoing a plethora of changes as the software keeps evolving continuously. Software is becoming more and more complex every day. Software testing, which was once a distinct phase, has now become an inevitable part of the Software Development Life Cycle. The earlier the faults are discovered, the lesser is the cost of repairing them. Many researchers in the past have attributed the failure of the software to the lack of adequate testing. A better relationship between testing and requirements was laid down as a prerequisite for better-quality software [1], leading to the establishment of test methods [2].

The method of identifying defective modules in software is Software Defect Prediction (SDP). Common methods for detecting faults include code analysis, device checking, integration testing, and machine testing. Early identification of defective components of the software helps verification practitioners to focus on the problematic areas in the software framework being built. According to [3], the defects are classified into five categories based on the phase in which they are detected. Customer needs which are described vaguely, prerequisites mentioned partially and misinterpreted requirements are some of the reasons for bugs in the requirements phase. Design phase bugs include sub-optimal design, missing design artifacts, etc. Lack of technical expertise, incomplete code review, and coding errors contribute to a majority of defects. Apart from these, inadequate testing and duration slippages also contribute to defective software.

Many researchers have used classification methods for predicting defects in software. Some of the most widely used techniques include Naïve Bayes, J48, C4.5, KNN, Multiple Linear Regression, Support Vector Machines, etc [4].
Multilayer Perceptron Neural Network [5] (MLP-NN) is one of the most powerful models in this area. In learning databases, multiple learning approaches suffer from over-matching. In this paper, an MLP-NN is used along with feature selection methods in predicting software vulnerabilities. It strengthens the learning algorithm's capability to generalise.

Computer applications have grown more and more sophisticated with the advancement of computing technologies, and there are many limitations in the life cycle development of software due to insufficient human ability. Relevant shortcomings are unavoidable, pending precise preparation, acceptable documentation, and product creation management systems are a few. A bug, fault, problem, mistake, or malfunction in a computer program is a software flaw. In this way, an erroneous or unintended outcome can be induced by the device. The identification of the mistakes that exist inside the codes is critical because it directly leads to cost savings and enhancement of software quality. Studies reveal that only in certain program modules the problem is identified. Such defects can lead to failure of the program, diminished customer satisfaction, or increased maintenance costs. Models for error prediction have been a common tool for early coding error detection.

The main focus of this paper is to evaluate the existing classifiers namely Random tree, Logistic Regression, and Multilayer Perceptron in their efficiency in predicting software defects. Our main objectives include:

- Minimizing the number of selected features for the classifier, and
- Maximizing the performance of the SDP model

We propose a model which combines feature selection with MLP-NN (MLP+FS). We evaluate the above-mentioned classifiers and the proposed model using open-source defect datasets obtained from AEEEM.

The rest of the paper contains five sections. Section II presents the work of other researchers in this area. Section III and IV describes the approach adopted in this paper and the experimental setup. Section V presents the results of the experiment and future work in this area.

II. RELATED WORK

A modern approach is recommended to overcome the current challenges by designing new learning approaches based on the concepts of the help vector machine by using evolutionary methods. The approach suggested avoids distortion of the topic and raises the margin of classification. Within 3 NASA libraries, the performance of the proposed algorithm was tested against 11 machine learning models and statistical methods [5].

Among product quality testers and practitioners, forecasting inaccurate software modules is of great concern. As a result, numerous attempts have been made using multiple approaches to predict machine errors. The paper uses hierarchical clustering strategy, k-NN, and Neural Network to identify program components as dysfunctional or vulnerable [19]. In the neural network approach, efficiency has been found to be higher compared to the clustering-based approach. A comparison of approaches for detecting faulty modules was done to find the best suitable model for predicting software defects. The study says that the choice of the right algorithm for data mining depends on multiple variables, such as the problem area, the dataset type, the scope of the project, and the dataset instability [6].

The use of approaches likes classification, prediction, clustering, and association rules to forecast sensitive blocks in the program is another approach to defect prediction. Researchers have recently begun investigating machine learning strategies to forecast the consistency of applications in addition to the data mining techniques mentioned above. Methods for integrating machine learning and mathematical methods have also been explored by researchers. In reality, machine learning approaches have been illustrated for poorly understood problem domains with evolving circumstances dependent on various principles and regulations. Since the problems in software can be classified into the learning and categorizing according to the defect characteristics, to evaluate the defects, traditional machine learning approaches are valid. Machine learning methods are based on neural networks, non-linear advanced simulation techniques that can simulate functions that are complex. The main purpose of this work was to explore the use of Multilayer Perceptron Neural Network [5] (MLP-NN) to model the error-proneness of the software systems, using the datasets from NASA MDP. Different algorithms were compared based on the RMSE and other accuracy values [7].

A review of the general machine error prediction paradigm that facilitates unbiased and rigorous analysis between competing forecasting systems is presented in this paper. The findings indicate that for different datasets, multiple learning programs need to be chosen (i.e., no program is dominated). Since most errors are found in certain modules, it is important to ask about modules that are badly affected relative to other modules, and proper maintenance, especially for sensitive applications, should be conducted promptly. Comparative software is a statistical research application that is used to compare more than 22 taxonomies from the NASA Matrix database across 10 public domain datasets. Using metric-based grouping, high predictive precision is observed in the STP experiment. Artificial immune system developed for defect prognosis dependent on the human immune system [5].

The suggested grouping follows the action of antigens and antibodies in the human biological system as pathogens attack. The immune system's evolution of new threats is intended to overcome the issue of the prediction of software defects in four separate real-time network vulnerability datasets, multiple software vulnerability prediction models. The findings suggest that, compared to other models with a consistency-based subgroup evaluation strategy, the combination of 1R and event-based learning provides comparatively improved consistency in precision prediction. One of the most sought-after data mining problems in the database world lately is classification. Coding classification rules were diagnosed using neural networks by the researcher [8].

Machine learning models to predict defective software modules were used by several researchers in the past. Some related studies are discussed here. In, Deep Neural Network model, the researchers proposed a hybrid model of genetic algorithm for effective defect prediction in software. The purpose of the hybrid genetic algorithm is to select the optimal features and to predict that the deep neural network modules are defective and flawless. Databases from the PROMISE repository are used for experiments. The results shows the high efficiency of the proposed model as compared to others. In this paper, the researchers elaborated on the importance of the feature selection process in the software defect prediction process [9].

This test was performed using NASA databases including PC1, CM1, KC1, and KC3. Experimental results of Logistic regression (LR), k-nearest neighbor (k-NN), end trees, Multilayer Perceptron (MLP), Bayesian networks, radial base function (RBF), Random Forest (RF), and now Naïve Base, (NB) [10]. The results reflected that the performance of the SVM surpassed some other classification methods. In this paper, the researchers predicted software defects using six classification methods: Discrimination Analysis, Primary Equipment Analysis (PCA), Logistic Regression (LR), Logical Classification, Holographic Networks, and Layered Neural Networks [10]. To train the ANN model, a back-propagation algorithm was used. The performance was evaluated using verification cost, forecast validity, the quality achieved, and incorrect classification ratio. According to the results, none of the classification techniques used was done with 100% accuracy [11].

The framework proposed in this paper used two dimensions: feature selection and without feature selection. 12 publicly available NASA MDP databases are used to implement the proposed framework. Performance is evaluated using a variety of measures including accuracy, recall, f-measurement, accuracy, MCC, and ROC. The results are comparable to well-known and widely used supervised machine learning techniques, such as: "Naïve Bayes (NB), Multi-Layer Perceptron (MLP), Radial Basis Function (RBF), Support Vector Machine (SVM), K Nearest Neighbor (kNN), k-Star (K*), One Rule (OneR), PART, Decision Tree (DT), Random Tree (RT) and Random Forest (RF)" [10]. The results showed that the proposed structure works better than other classification techniques in some datasets. Twelve publicly available NASA MDP databases are used for this test and performance is evaluated based on accuracy, recall, F-measurement, accuracy, MCC and ROC area [12].

III. BACKGROUND

A. *Software Metrics*

Software metrics play a major role in analyzing the performance of a software. Metrics for analyzing software can include basics like the total lines of code, number of packages in the code, number of classes and interfaces, number of overridden methods, count of parameters and static fields, etc [12]. The complexity of the code can be measured using metrics like McCabe's cyclomatic complexity, level of nesting, count of lines of code in methods, etc. While testing object-oriented software, the role of class-level object-oriented metrics plays a vital role in uncovering defects. Chidamber and Kemerer, six metrics, namely WMC, DIT, NOC, RFC, CBO, and LCOM [13]. These metrics measure the number of weighted methods per class, the depth of the inheritance tree, the number of children. They also consider the degree of coupling and lack of cohesion between objects and the response set of a class [13]. The AEEEM dataset has a total of 61 metrics from five Eclipse-based projects namely EQ, JDT, Lucene, Mylyn, and PDE [18].

B. *Multi-layer Perceptron Neural Network*

A Multi-layer neural network [5] is an artificial neural network with more than one hidden layer. As the number of the hidden layer is increased, the accuracy of the classification will be benefitted. Each layer has a different number of neurons, different activation functions, and so on. By varying the parameters, better classification accuracy can be achieved. Neural networks are capable of capturing data relationships in a precise manner, even those skipped by humans.

C. *Feature Selection Methods*

Feature Selection [10] is used to reduce the number of inputs to a classifier. Several feature selection methods are available, based on the kind of learning used. Wrapper methods are used to evaluate multiple models to add or delete predictors in order to maximize model performance. Filter-based methods are used to evaluate the relevance of the predictors based on some criteria. A Correlation-based feature selection method (CFS) uses a correlation index to choose an attribute over the other. Attributes that are highly correlated are not selected, as their contribution to the output is more or less the same [10].

IV. PROPOSED METHOD

This research presents a framework for the classification of software modules as defective or clean using a Multilayer neural network combined with feature selection methods. There are four stages in the prediction model. The different phases are a selection of dataset, pre-processing the data, implementation of various classification methods analysis of the results.

The target /output class is the dependent attribute and the rest of the attributes are called independent attributes [10]. The bias characteristic is predicted based on the independent properties. Independent attributes are quality measurements of software systems. The target class in the datasets used has one of the following values: "clean" or "buggy". An output value of "clean" means that the block of code is not defective and "buggy" means defective.
The selection of data is the second phase of the proposed framework, which includes feature selection and class balancing operations [14].

The proposed architecture operates in two dimensions, with the first dimension having only the pre-processing stage feature selection function. However, in the second dimension, along with the feature selection function, a class balancing technique is also included [15].

The class balance technique allows us to analyze the effects of unbalanced datasets on the performance of the proposed classification structure. The feature selection function aims to select the optimal features so that classification results can be achieved with greater accuracy. It has been suggested by many researchers that only a few independent features in many datasets can be used to predict the target class effectively [10]. The remaining features which do not participate, will reduce the effectiveness of the model if not eliminated [16].

In this research, we have incorporated a collection-based multi-filter feature selection technique in which CFS is used as an attribute evaluator, and the four most widely used search methods are Best Fit and Greedy Search. For each dataset used, the feature is selected with all four of these search methods. In this process, if any particular feature is selected with any search method, that feature is given a score of 1, and the same process is repeated with the second search method and so on. After activating all search methods, the scores of each feature are collected in all search methods, and only those features that have at least 1 accumulated score are selected (this feature is selected by at least one search method) [17].

V. EXPERIMENTAL SETUP

The proposed model is obtained by slightly modifying the existing MLP-NN model. Our model has two hidden layers, with 20 neurons in each layer. We have tried different activation functions in each iteration and choose the one which gave highest accuracy. Also, the learning rate was vari ed in order to achieve maximum possible accuracy. A total of 43 metrics were shortlisted for the proposed MLP-NN by applying CFS feature selection algorithm available in WEKA 3.9.5. Two search strategies were implemented, namely, best fit and Greedy search.

The dataset was evaluated using classifiers namely Logistic Regression (LR) [10], Random Tree (RT) [10], Multi-Layer Perceptron (MLP-NN) [5] and our proposed model (MLP+FS). The classifier accuracy in each case was tabulated.

V. RESULTS AND CONCLUSION

Tableable 1: Percentage Of Correctly Classified Data

Classifier	EQ	JDT	Lucene	Mylyn	PDE
RT [10]	65.98	76.92	84.54	83.36	80.18
LR [10]	69.07	84.61	85.99	86.40	87.08
MLP-NN	70.10	82.00	89.85	87.11	86.19
MLP+FS	80.00	82.00	89.00	87.00	86.00

-

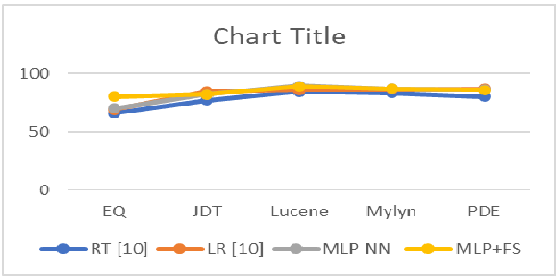

Fig 1: Graph of the classifier accuracy values

Table I shows the accuracy of various classifiers for the 5 datasets. Figure 1 represents the graph of the data. Figures 2 to 6 shows the confusion matrix for the classification of the 5 datasets. The yellow line shows the accuracy of the proposed model. The blue and yellow colors show correctly classified cases i.e., True Positive (TP) and True Negative (TN) values. The purple color shows False Positive (FP) and False Negative (FN) values.

From the above results, it is evident that the MLP-NN was able to classify the data more accurately. Also, our proposed model which combines feature selection with MLP-NN (MLP+FS) was able to achieve a similar accuracy rate. It clearly shows that selecting a subset of features based on correlation will not only reduce the dimensions of the input but also improve the accuracy of the classification.

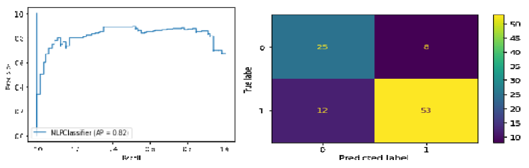

Fig 2: Precision-Recall Curve and Confusion Matrix for our Proposed method (EQ)

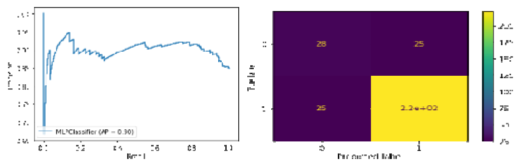

Fig 3: Precision-Recall Curve and Confusion Matrix our Proposed method (JDT)

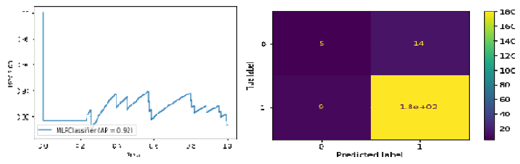

Fig 4: Precision-Recall Curve and Confusion Matrix for our Proposed method (Lucene)

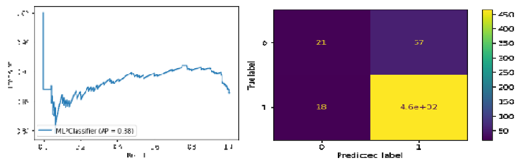

Fig 5: Precision-Recall Curve and Confusion Matrix for our Proposed method (Mylyn)

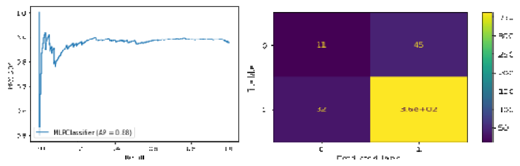

Fig 6: Precision-Recall Curve and Confusion Matrix for our Proposed method (PDE)

However, the proposed method for better testing in terms of performance consistency should be evaluated along with other data sets. In conclusion, the MLP model combined with feature selection offers a better solution for predicting errors in software modules. This research provided a multi-filter feature selection-based classification framework for software defect prediction. For defect prediction, the structure uses a synthetic neural network (MLP). Oversampling technique is also used in the structure to analyze the effect of the class inequality problem on classification performance. For testing, 5 datasets from AEEEM namely EQ, JDT, Lucene, Mylyn and PDE was used. We need to further why the class balancing technique has reduced accuracy while other measures have been significantly improved in most databases. Our future work will include evaluating our model with more datasets like JIRA, PROMISE and NASA MDP. We also aim at maximizing the accuracy of the prediction done by our model.

VI. References

[1] Suresh Jat., Pradeep Sharma., "*Analysis of Different Software Testing Techniques*". International Journal of Scientific Research in Computer Science and Engineering Vol.5, Issue.2, pp.77-80, April 2017.

[2] Chandraprakash Patidar., "*A Report on Latest Software Testing Techniques and Tools*". International Journal of Scientific Research in Computer Science and Engineering Vol.1, Issue.4, pp.50-52, Dec 2016.

[3] Sakthi Kumaresh and Baskaran Ramachandran, “Defect Prevention based on 5 Dimensions of Defect Origin”, International Journal of Software Engineering and Application, Vol 3, 87-98.

[4] Ning Li, Martin Shepperd, Yuchen Guo, “A Systematic Review of Unsupervised Learning Techniques for Software Defect Prediction”,Information and Software Technology.

[5] M. Gayathri and A. Sudha, “Software Defect Prediction System using Multilayer Perceptron Neural Network with Data Mining,” *Int. J. Recent Technol. Eng.*, no. 32, pp. 2277–3878, 2014.

[6] S. U. Dr. Naheed Azeem, “Analysis of Data mining based Software Defect Prediction Techniques,” Glob. J. Comput. Sci. Technol., vol. 11, no. 16, 2011, [Online]. Available: https://computerresearch.org/index.php/computer/article/view/806.

[7] M. C. M. Prasad, L. F. Florence, and A. Arya3, “A Study on Software Metrics based Software Defect Prediction using Data Mining and Machine Learning Techniques,” Int. J. Database Theory Appl., vol. 8, no. 3, pp. 179–190, 2015, doi: 10.14257/ijdta.2015.8.3.15.

[8] A. Iqbal and S. Aftab, “A classification framework for software defect prediction using multi-filter feature selection technique and MLP,” Int. J. Mod. Educ. Comput. Sci., vol. 12, no. 1, pp. 18–25, 2020, doi: 10.5815/ijmecs.2020.01.03.

[9] M. M. Askari and V. K. Bardsiri, “Software defect prediction using a high performance neural network,” Int. J. Softw. Eng. its Appl., vol. 8, no. 12, pp. 177–188, 2014, doi: 10.14257/ijseia.2014.8.12.17.

[10] Ahmed Iqbal etal., “A Feature Selection based Ensemble Classification Framework for Software Defect Prediction”, International Journal of Modern Education and Computer Science, 2019.

[11] A. Iqbal et al., “Performance analysis of machine learning techniques on software defect prediction using NASA datasets,” Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 5, pp. 300–308, 2019, doi: 10.14569/ijacsa.2019.0100538.

[12] N. Gayatri, S. Nickolas, and A. V Reddy, “Feature Selection Using Decision Tree Induction in Class level Metrics Dataset for Software Defect Predictions,” Lect. Notes Eng. Comput. Sci., vol. 2186, no. 1, pp. 124–129, 2010.

[13] Shyam R. Chidamber and Chris F. Kemerer, “A Metrics Suite for Object Oriented Design”, IEEE Transactions on Software Engineering, Vol 20, No.6, June 1994

[14] A. G. Akintola, A. Balogun, F. B. Lafenwa-Balogun, and H. A. Mojeed, “Comparative Analysis of Selected Heterogeneous Classifiers for Software Defects Prediction Using Filter-Based Feature Selection Methods,” FUOYE J. Eng. Technol., vol. 3, no. 1, pp. 133–137, 2018, doi: 10.46792/fuoyejet.v3i1.178.

[15] S. Bibi, G. Tsoumakas, I. Stamelos, and I. Vlahavas, “Software defect prediction using regression via classification,” IEEE Int. Conf. Comput. Syst. Appl. 2006, vol. 2006, pp. 330–336, 2006, doi: 10.1109/aiccsa.2006.205110.

[16] N. Ahmad Alawad and N. Ghani Rahman, “Design of (FPID) controller for Automatic Voltage Regulator using Differential Evolution Algorithm,” Int. J. Mod. Educ. Comput. Sci., vol. 11, no. 12, pp. 21–28, 2019, doi: 10.5815/ijmecs.2019.12.02.

[17] H. Wang, T. M. Khoshgoftaar, J. Van Hulse, and K. Gao, “Metric selection for software defect prediction,” Int. J. Softw. Eng. Knowl. Eng., vol. 21, no. 2, pp. 237–257, 2011, doi: 10.1142/S0218194011005256.

[18] M. D’Ambros, M. Lanza, and R. Robbes, “Evaluating defect prediction approaches: A benchmark and an extensive comparison,”Empirical Softw. Engg., vol. 17, no. 4-5, pp. 531–577, Aug. 2012.

[19] Arnau V., Marín I. (2003) A Hierarchical Clustering Strategy and Its Application to Proteomic Interaction Data. In: Perales F.J., Campilho A.J.C., de la Blanca N.P., Sanfeliu A. (eds) Pattern Recognition and Image Analysis. IbPRIA 2003. Lecture Notes in Computer Science, vol 2652. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-44871-6_8.