International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

# A Novel Machine Learning Approach for Estimation of Software Defects

**Ms.M.ANITHA[1], Ms. L.MEGHANA [2]**

**#1** Assistant professor in the Master of Computer Applications in the SRK Institute of Technology, Enikepadu, Vijayawada, NTR District

#2 MCA student in the Master of Computer Applications at SRK Institute of Technology, Enikepadu, Vijayawada, NTR District

**Abstract_** One of the most important components of software is its quality. Software designs are becoming more sophisticated as demand grows, increasing the likelihood of software failures. By repairing flaws, testers help to increase the quality of software. As a result, defect analysis increases software quality dramatically. The project's resources and the effort of the software developers can be allocated more efficiently for system development and quality assurance operations thanks to effective system defect prediction on the front line of the project life cycle. The main goal of this research is to compare seven machine learning algorithms in the context of four NASA datasets collected from the public PROMISE repository [12] in order to evaluate their competence in software defect prediction and determine the best category. Overall, the results of the ensemble learners category in defect prediction, which includes Random Forests (RF) and Bagging, are very similar to their counterparts.

## 1.INTRODUCTION

The software industry is rapidly evolving as a result of rising demand and technological advancements. Defects will eventually arise because the majority of software development is done by people. Defects are undesired or unacceptable deviations in software documentation, programmes, and data in general [1]. Defects may arise in requirements analysis as a result of the product manager's misinterpretation of the customer's needs, and this error will then be carried over to the system design phase. Inexperienced developers might potentially cause defects in the code. Defects have a substantial impact on software quality, resulting in higher software maintenance costs, particularly in the healthcare industry, and aircraft software defects can have fatal implications. If a bug is discovered after the product has been deployed, the

development team will have to re-design some software components, which will increase development expenses. Defects are a nightmare for reputable businesses. Because of client unhappiness, their reputation suffers, and their market share suffers as a result. As a result, software testing has emerged as one of the most important areas of industry research [2]. As software development and complexity have increased, the number of faults has climbed to the point where traditional manual procedures have become inefficient and time-consuming. Automatic fault categorization has become a research hotspot thanks to the rise of machine learning. In this study, we first go through software flaws in depth and the numerous categories that have been offered in the literature, before moving on to the manual classification methods proposed by various scholars. Finally, we discuss the current state of machine learning methods for autonomous software detection.

## 2.LITERATURE SURVEY

**2.1 S. Parnerkar, A. V. Jain, and C. Birchha, ‘‘An approach to efficient software bug prediction using regression analysis and neural networks,’’ Int. J. Innov. Res. Comput. Commun. Eng., vol. 3, no. 10, Oct. 2015.**

Machine Learning approaches are good in solving problems that have less information. In most cases, the software domain problems characterize as a process of learning that depend on the various circumstances and changes accordingly. A predictive model is constructed by using machine learning approaches and classified them into defective and non-defective modules. Machine learning techniques help developers to retrieve useful information after the classification and enable them to analyse data from different perspectives. Machine learning techniques are proven to be useful in terms of software bug prediction. This study used public available data sets of software modules and provides comparative performance analysis of different machine learning techniques for software bug prediction. Results showed most of the machine learning methods performed well on software bug datasets. The advancement in software technology causes an increase in the number of software products, and their maintenance has become a challenging task. More than half of the life cycle cost for a software system includes maintenance activities.

**2.2 B. Liu, H. Qin, Y. Gong, W. Ge, M. Xia, and L. Shi, ‘‘EERA-ASR: An energy-efficient reconfigurable**

**architecture for automatic speech recognition with hybrid DNN and approximate computing,'' IEEE Access, vol. 6, pp. 52227–52237, 2018.**

This paper proposes a hybrid deep neural network (DNN) for automatic speech recognition and an energy-efficient reconfigurable architecture with approximate computing for accelerating the DNN. To accelerate the hybrid DNN and reduce the energy consumption, we propose a digital–analog mixed reconfigurable architecture with approximate computing units, including a binary weight network accelerator with analog multi-chain delay-addition units for bit-wise approximate computing and a recurrent neural network accelerator with approximate multiplication units for different calculation accuracy requirements. Implemented under TSMC 28nm HPC+ process technology, the proposed architecture can achieve the energy efficiency of 163.8TOPS/W for 20 keywords recognition and 3.3TOPS/W for common speech recognition. Deep Neural Networks (DNNs) that have many hidden layers have been proven to outperform traditional models (i.e., Markov models, Gaussian mixture models) on a variety of speech recognition benchmarks by a large margin [1], [2].

**2.3 N. Cummins, S. Amiriparian, G. Hagerer, A. Batliner, S. Steidl, and B. W. Schuller, ''An image-based deep spectrum feature representation for the recognition of emotional speech,'' in Proc. 25th ACM Multimedia Conf. (MM), 2017, pp. 478–484.**

The outputs of the higher layers of deep pre-trained convolutional neural networks (CNNs) have consistently been shown to provide a rich representation of an image for use in recognition tasks. This study explores the suitability of such an approach for speech-based emotion recognition tasks. First, we detail a new acoustic feature representation, denoted as deep spectrum features, derived from feeding spectrograms through a very deep image classification CNN and forming a feature vector from the activations of the last fully connected layer. We then compare the performance of our novel features with standardised brute-force and bag-of-audio-words (BoAW) acoustic feature representations for 2- and 5-class speech-based emotion recognition in clean, noisy and denoised conditions. The presented results show that image-based approaches are a promising avenue of research for speech-based recognition tasks. Key results indicate that deep-spectrum features are comparable in

performance with the other tested acoustic feature representations in matched for noise type train-test conditions; however, the BoAW paradigm is better suited to cross-noise-type train-test conditions. Convolutional neural networks (CNNs) have become increasingly popular in machine learning research.

## 4.PROPOSED SYSTEM

The author of this study compares the performance of various machine learning algorithms to find faults or defects in software components, including SVM, Bagging, Nave Bayes, Multinomial Nave Bayes, RBF, Random Forest, and Multilayer Perceptron Algorithms. Defects in software components will emerge as a result of bad coding, which will raise software development and maintenance costs, as well as customer dissatisfaction. Various techniques have been developed to detect faults in software components, but machine learning algorithms are currently getting a lot of traction due to their superior performance. As a result, the author of this research use machine learning methods to detect faults in software modules. The datasets CM1 and KC1 are used in this paper by the author, and they are from NASA Software components. I'm also evaluating the above-mentioned algorithms' performance using the same datasets.

### 4.1 ALGORITHM DETAILS

**SVM Algorithm**: Machine learning involves predicting and classifying data and to do so we employ various machine learning algorithms according to the dataset. SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: The algorithm creates a line or a hyper plane which separates the data into classes. In machine learning, the radial basis function kernel, or RBF kernel, is a popular kernel function used in various kernelized learning algorithms. In particular, it is commonly used in support vector machine classification. As a simple example, for a classification task with only two features (like the image above), you can think of a hyper plane as a line that linearly separates and classifies a set of data. Intuitively, the further from the hyper plane our data points lie, the more confident we are that they have been correctly classified. We therefore want our data points to be as far away from the hyper plane as possible, while still being on the correct side of it.

So when new testing data is added, whatever side of the hyper plane it lands will decide the class that we assign to it.

**Random Forest Algorithm:** it's an ensemble algorithm which means internally it will use multiple classifier algorithms to build accurate classifier model. Internally this algorithm will use decision tree algorithm to generate it train model for classification.

**Bagging**: This algorithms work similar to learning tree the only difference is voting concept where each class will get majority of votes based on values close to it and that class will form a branch. If new values arrived then that new value will applied on entire tree to get close matching class.

**Naive Bayes**: Naive Bayes which is one of the most commonly used algorithms for classifying problems is simple probabilistic classifier and is based on Bayes Theorem. It determines the probability of each features occurring in each class and returns the outcome with the highest probability.

**Multinomial Naive Bayes:** Multinomial Naive Bayes classifier is obtained by enlarging Naive Bayes classifier. Differently from the Naive Bayes classifier, a multinomial distribution is used for each features.

**Multilayer Perceptron**: Multilayer Perceptron which is one of the types of Neural Networks comprises of one input layer, one output layer and at least one or more hidden layers. This algorithm transfers the data from the input layer to the output layer, which is called feed forward. For training, the back propagation technique is used. One hidden layer with (attributes + classes) / 2 units are used for this experiment. Each dataset has 22 attributes and 2 classes which are false and true. We determined the learning rate as 0.3 and momentum as 0.2 for each dataset.

.

## 4.RESULTS AND DISCUSSION

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

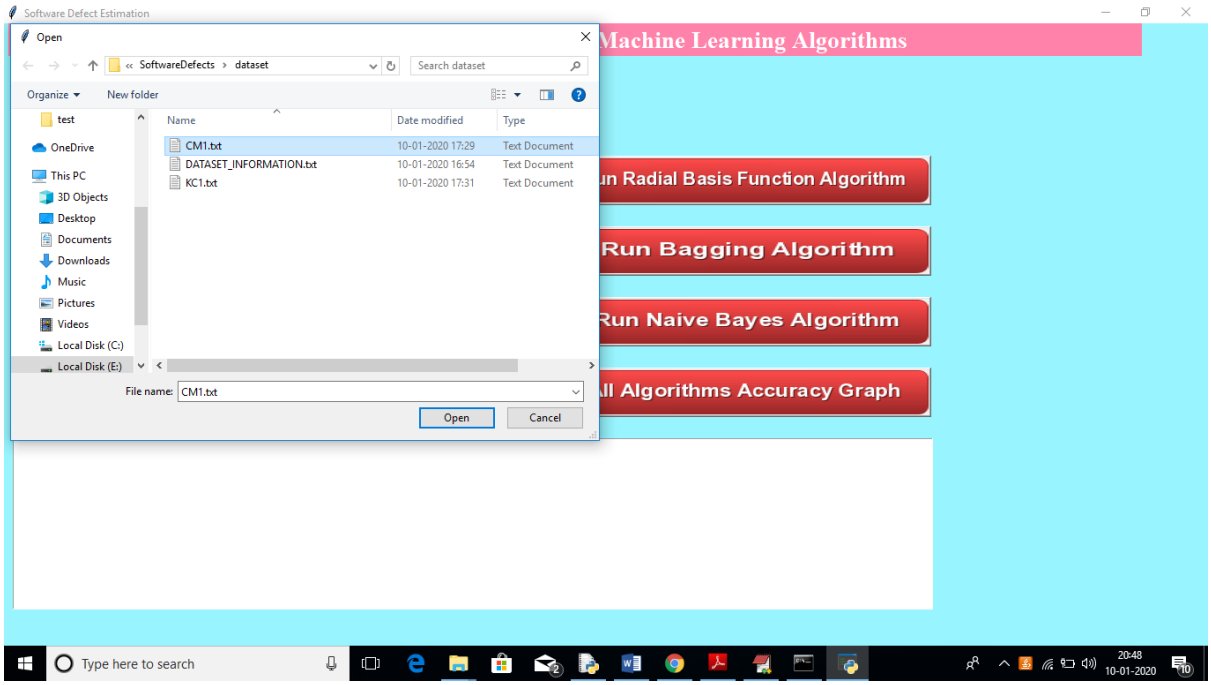


**Fig 4.1 In above screen uploading 'CM1.txt' dataset and information of this dataset you can read from internet of 'DATASET_INFORMATION' file from above screen. After uploading dataset will get below screen**

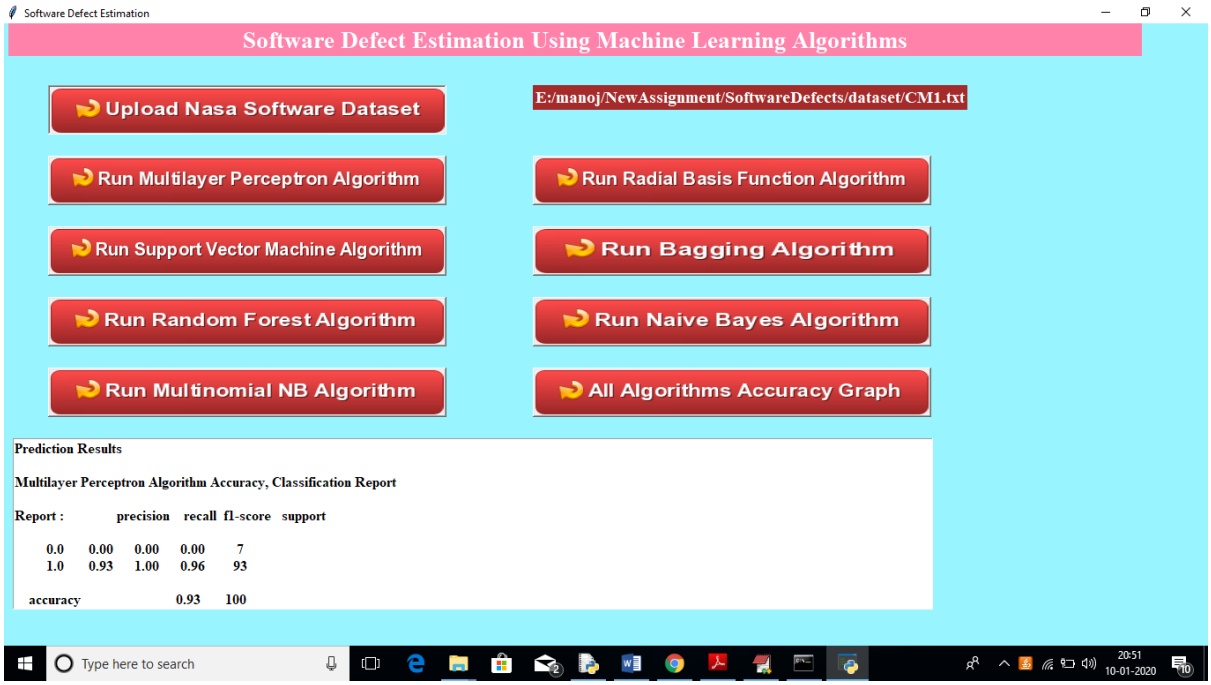


**Fig 4.2 In above screen we can see multilayer perceptron fmeasure, recall and accuracy values and scroll down in text area to see all details.**

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362


**Fig 4.3 In above screen we can see multilayer perceptron accuracy is 93%. Similarly you click on all other algorithms button to see their accuracies and then click on 'All Algorithms Accuracy Graph' button to see all algorithms accuracy in graph to understand which algorithm is giving high accuracy.**

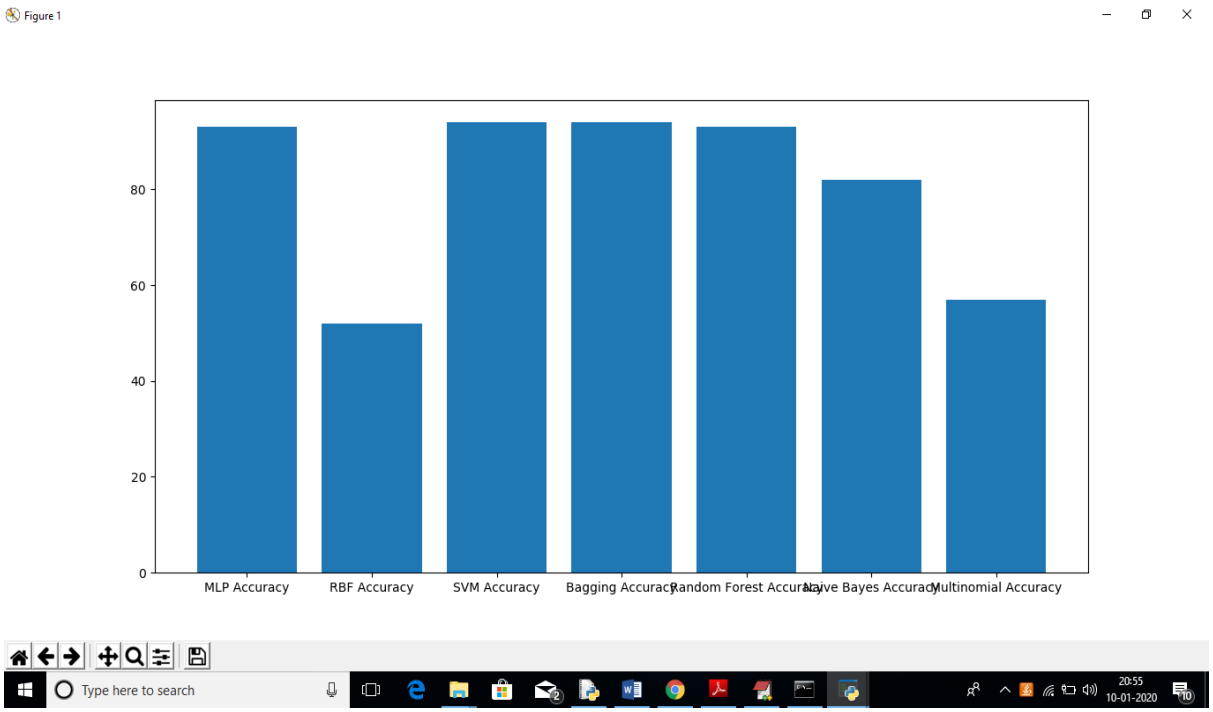


**Fig 4.4 In above graph x-axis represents algorithm name and y-axis represents accuracy of those algorithms. In all algorithms we can see MLP, Bagging is giving better accuracy.**

## 5.CONCLUSION

Seven machine learning algorithms are used in this experimental study to predict the defectiveness of software systems before they are released to the real world and/or delivered to customers, and the best category with the most capability to predict software defects is sought out by comparing them using software quality metrics such as accuracy, precision, recall, and F-measure. PC1, CM1, KC1, and KC2 are the four NASA datasets used in this experimental study. These datasets were collected from the PROMISE repository, which is open to the public. The results of this experiment show that tree-structured classifiers, also known as ensemble learners, such as Random Forests and Bagging, outperform their counterparts in defect prediction. Especially, the capability of Bagging in predicting software defectiveness is better. When applied to all datasets, the overall accuracy. In all algorithms we can see MLP, Bagging is giving better accuracy.

## REFERENCES

[1] Victor R Basili, Lionel C. Briand, and Walcelio L Melo. ´ A validation of object-oriented design metrics as quality indicators. IEEE Transactions on software engineering, 22(10):751–761, 1996.

[2] Evren Ceylan, F Onur Kutlubay, and Ayse B Bener. Software defect identification using machine learning techniques. In 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO'06), pages 240–247. IEEE, 2006.

[3] Karim O Elish and Mahmoud O Elish. Predicting defect-prone software modules using support vector machines. Journal of Systems and Software, 81(5):649– 660, 2008.

[4] Norman Fenton, Paul Krause, and Martin Neil. Software measurement: Uncertainty and causal modeling. IEEE software, 19(4):116–122, 2002.

[5] Lan Guo, Yan Ma, Bojan Cukic, and Harshinder Singh. Robust prediction of fault-proneness by random forests. In 15th International Symposium on Software Reliability Engineering, pages 417–428. IEEE, 2004.

[6] Taghi M Khoshgoftaar, Edward B Allen, and Jianyu Deng. Using regression trees to classify fault-prone software modules. IEEE Transactions on reliability, 51(4):455–462, 2002.

[7] Taghi M Khoshgoftaar, Edward B Allen, John P Hudepohl, and Stephen J

Aud. Application of neural networks to software quality modeling of a very large telecommunications system. IEEE Transactions on Neural Networks, 8(4):902–909, 1997.

[8] Sunghun Kim, Hongyu Zhang, Rongxin Wu, and Liang Gong. Dealing with noise in defect prediction. In 2011 33rd International Conference on Software Engineering (ICSE), pages 481–490. IEEE, 2011.

[9] Yan Ma, Lan Guo, and Bojan Cukic. A statistical framework for the prediction of fault-proneness. In Advances in Machine Learning Applications in Software Engineering, pages 237–263. IGI Global, 2007.

[10] Ruchika Malhotra. A systematic review of machine learning techniques for software fault prediction. Applied Soft Computing, 27:504–518, 2015.

[11] Jinsheng Ren, Ke Qin, Ying Ma, and Guangchun Luo. On software defect prediction using machine learning. Journal of Applied Mathematics, 2014, 2014.

[12] J. Sayyad Shirabad and T.J. Menzies. The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa, Canada, 2005.

[13] Shuo Wang and Xin Yao. Using class imbalance learning for software defect prediction. IEEE Transactions on Reliability, 62(2):434–443, 2013.

[14] Robert Andrew Weaver. The safety of software: Constructing and assuring arguments. University of York, Department of Computer Science, 2003.

**AUTHOR'S PROFILE**

**Ms. M. ANITHA** completed her Master of Computer Applications and Masters of Technology. Currently working as an Assistant professor in the Department of Masters of Computer Applications in the SRK Institute of Technology, Enikepadu, Vijayawada, NTR District. Her area of interest includes Machine Learning with Python and DBMS.

**Ms. L.MEGHANA** is an MCA student in the Department of Master Of Computer Applications at SRK Institute of Technology, Enikepadu, Vijayawada, NTR District. She has Completed Degree in B.Sc.(computers) from Sri Durga Malleswara Siddhartha Mahila Kalasala Degree college Vijayawada. Her areas of interest are DBMS, Java Script, and Machine Learning with Python.