JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Code Vectorizer Based Machine Learning Models for Software Defect Prediction

Mr. Vatti Srikanth[*1], Mr. Ch. Srinivasa Reddy[*2]

[1]MCA Student, Department of Master of Computer Applications,
Vignan's Institute of Information Technology (A), Beside VSEZ, Duvvada, Vadlapudi Post,Gajuwaka, Visakhapatnam-530049.

[2]Assistant Professor, Department of Information Technology,
Vignan's Institute of Information Technology(A), Beside VSEZ,Duvvada,Vadlapudi Post,
Gajuwaka, Visakhapatnam-530049.

Abstract

Even with meticulous planning, thorough documentation, and suitable procedures, some faults in software are unavoidable. These software flaws could result in a drop in quality, which could be the root cause of failure. Different techniques have been developed by researchers for efficient software defect prediction. A software module's vulnerability to flaws or defects can be predicted, which can speed up testing by allowing developers and testers to focus their efforts and resources on those modules. In this project, we provide Code Vectorizer, a revolutionary technique for defect prediction that extracts information from the language of the software source code itself. Despite of great planning, well documentation and proper process during software development, occurrences of certain defects are inevitable. These software defects may lead to degradation of the quality which might be the underlying cause of failure. Researchers have devised various methods that can be used for effective software defect prediction. The prediction of the presence of defects or bugs in a software module can facilitate the testing process as it would enable developers and testers to allocate their time and resources on modules that are prone to defects.

Keywords: Code Vectorizer,Prediction,Software Modules,Vulnerability,Software Development.

1. INTRODUCTION

The efficiency of the testing phase of a Software Development Life Cycle (SDLC) can be increased using software defect prediction (SDP). In the past, researchers have consolidated data from numerous information technology organizations and applied statistical and machine learning approaches. For each research project, computer scientists have used different data with each dataset having a multitude of data related issues. The PROMISE Software Engineering repository offers a wide range of datasets consisting of software metrics data from NASA projects and it is one of the most used datasets for SDP. In software engineering, researchers have adopted different software complexity measurement metrics such as Halstead and McCabe metrics. Erturk and Sezer used datasets in the PROMISE data repository and focused on McCabe software metrics.

It was reported that LM algorithm-based neural network had a higher accuracy than the general polynomial function based neural network. In, Fenton and Neil explored the application of Bayesian Belief Networks (BBN) for SDP. They also highlighted the key issues in software engineering research such as gauging a relationship between software metrics and

defects, problems with multivariate statistical techniques and quality of the data being used. In, Malhotra discussed six SDP research questions ranging from the choice of SDP datasets, choice of machine learning algorithms, performance metrics and the strengths and weaknesses of application of machine learning in SDP.

The performance of machine learning algorithms has a direct correlation with the quality of data. Due to the nature of the problem, SDP datasets tend to have a high level of class imbalance i.e., the occurrences of non-defective data points are much higher than defective datapoints because of which machine learning models tend to have a bias towards the non-defective class.

2.LITERATURE SURVEY

[1] E. Erturk and E. A. Sezer, "A comparison of some soft computing methods for software fault prediction," Expert Systems with Applications, vol. 42, no. 4, 2015, pp. 1872–1879.

The main expectation from reliable software is the minimization of the number of failures that occur when the program runs. Determining whether software modules are prone to fault is important because doing so assists in identifying modules that require refactoring or detailed testing. Software fault prediction is a discipline that predicts the fault proneness of future modules by using essential prediction metrics and historical fault data. This study presents the first application of the Adaptive Neuro Fuzzy Inference System (ANFIS) for the software fault prediction problem. Moreover, Artificial Neural Network (ANN) and Support Vector Machine (SVM) methods, which were experienced previously, are built to discuss the performance of ANFIS. Data used in this study are collected from the PROMISE Software Engineering Repository, and McCabe metrics are selected because they comprehensively address the programming effort. ROC-AUC is used as a performance measure. The results achieved were 0.7795, 0.8685, and 0.8573 for the SVM, ANN and ANFIS methods, respectively.

[2] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," Applied Soft Computing vol. 27, 2015, pp. 504–518.

In this study we perform a systematic review of studies from January 1991 to October 2013 in the literature that use the machine learning techniques for software fault prediction. We assess the performance capability of the machine learning techniques in existing research for software fault prediction. We also compare the performance of the machine learning techniques with the statistical techniques and other machine learning techniques. Further the strengths and weaknesses of machine learning techniques are summarized.

3.EXISTING SYSTEM

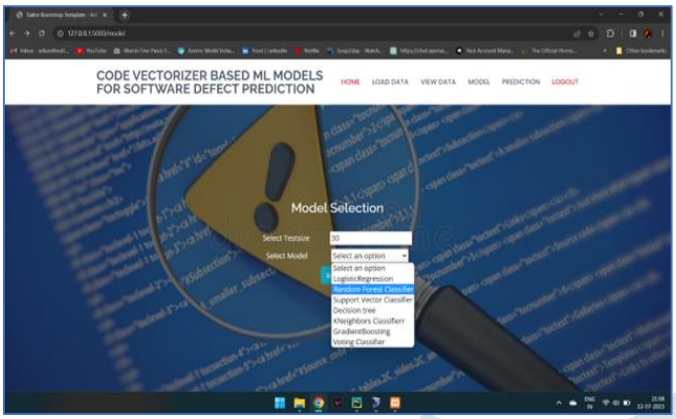
In previous research on this field, many machine learning algorithms are used. Algorithms like Random Forest, SVM, Gradient boosting are even proven to be better models than many deep learning models involving neural networks in multiple layers.

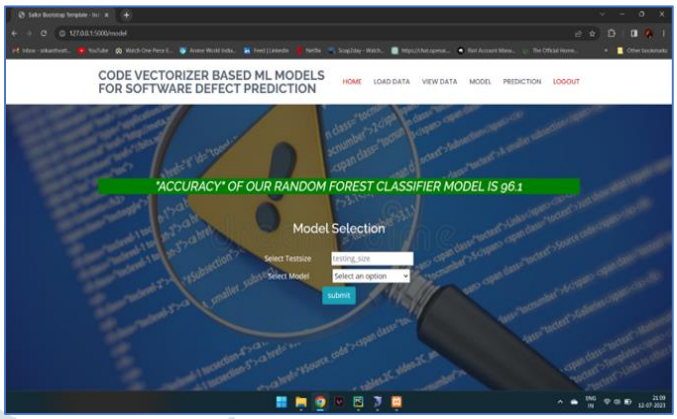
PROBLEMS IN EXISTING SYSTEM:

- Low accuracy.
- Difficult to handle.
- Low reliability.
- Time Consuming.

4. PROPOSED SYSTEM

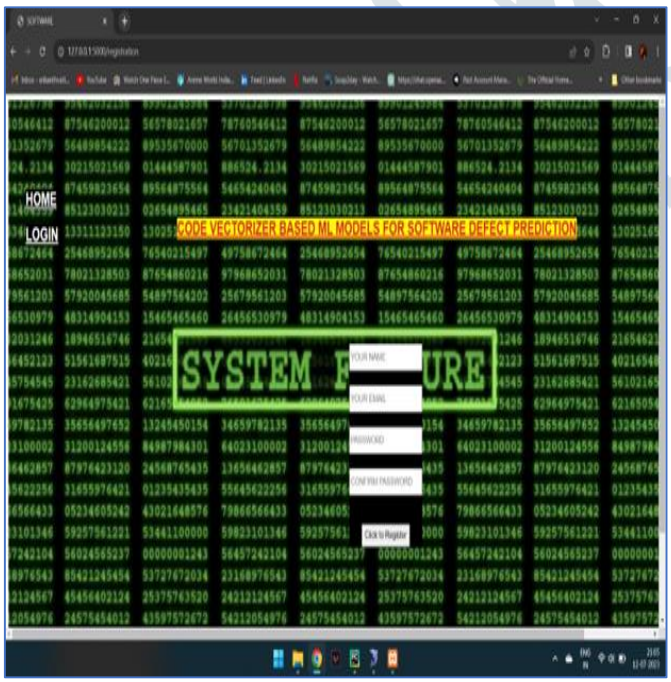
In the proposed system, we aimed to improve estimation accuracy by using relevant features of a large dataset. We used Ensemble techniques to improve accuracy, precision, and recall scores. We are using Machine learning algorithms such as Support Vector Machines (SVM), gradient boosting, Random Forest, Decision Trees, Logistic Regression and combines

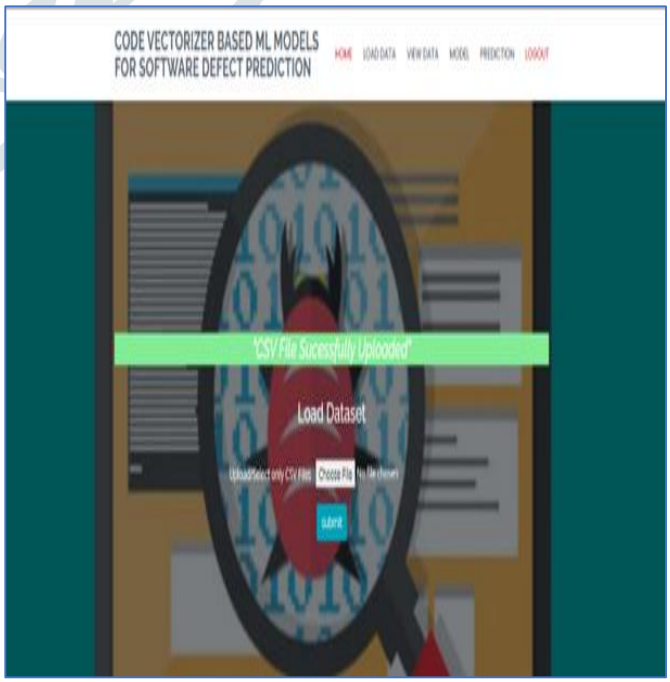




all the models then Ensembling into other model for the final output.

5. EXPERIMENTAL RESULTS





6. CONCLUSION

In this application, we have successfully created a ML modelsfor the prediction of software defects. This is developed in a Jupyter Notebook. We used some machine learning algorithm which helps us to predict the flaws in the software, which are noticed as Ensembling Random Forest algorithm, Decision Tree Algorithm, Logistic Regression, Support Vector Machines, Gradient Boosting and KNN algorithms models will give better results when compared to the individual model results. It also over comes the discrepancy created by the previous models/projects. This project maintains full-fledged record of the process and the user who have performed the task as well.

7. REFERENCES

[1] Wang, K., Makond, B., Chen, K., & Wang, K. A hybrid classifier combining SMOTE with PSO to estimate 5-year survivability of breast cancer patients, Applied Soft Computing, 20 (07), 15-24, 2014.

[2] S. Shirabad, J. and T.J. Menzies, “The PROMISE Repository of Software Engineering Databases,” School of Information Technology and Engineering, University of Ottawa, Canada, 2005 Available: http://promise.site.uottawa.ca/SERepository

[3] E. Erturk and E. A. Sezer, “A comparison of some soft computing methods for software fault prediction,” Expert Systems with Applications, vol. 42, no. 4, 2015, pp. 1872–1879.

[4] M. Singh and D. S. Singh, "Software Defect Prediction Tool based on Neural Network,” International Journal of Computer Applications, vol. 70, 2013.

[5] N.V. Chawla, K.W. Bowyer, L.O. Hall and W.P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” Journal of Artificial Intelligence Research, vol. 16, 2002, pp. 321-357.

[6] V. Phan, M. L. Nguyen, and L. T. Bui, “Convolutional Neural Networks over Control Flow Graphs for Software Defect Prediction,” 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI), pp. 45–52, 2017.

[7] S. Ruder, M. E. Peters, S. Swayamdipta, and T. Wolf, “Transfer learning in natural language processing,” NAACL-HLT, pp. 15–18, 2019.

[8] U. Alon, M. Zilberstein, O. Levy, and E. Yahav, “code2vec: Learning distributed representations of code,” Proceedings of the 46th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, vol. 3. ACM, pp. 40:1–40:29, 2019.