Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 4, April : 2023

SOFTWARE "DEFECT DETECTION" STRATEGIES IN CONTEMPORARY METHODOLOGIES USING ML

Dr. N. NEELIMA PRIYANKA., Associate Professor, Department of CSE, S.R.KINSTITUTEOFTECHNOLOGY, Enikepadu,Vijayawada–521108,, Andhra Pradesh., India.
P. KARUNA JYOTHI, M. RAJALAKSHMI, A. KETHAN, SK. BASHA RIYAZ,Student, Department of CSE, S.R.KINSTITUTEOFTECHNOLOGY, Enikepadu,Vijayawada–521108,, Andhra Pradesh., India.

Abstract
Software engineering is a subfield of computer science that emphasizes open lines of communication among all parties involved in a system and the timely and cost-effective completion of the system's development. Planning, analyzing, designing, implementing, testing, integrating, and maintaining a software system nowadays is no easy task. The job of a software engineer entails meeting strict deadlines and financial constraints while creating complex computer systems. Few faults may generate mistakes that lead to re-doing the job, increasing the development and expense of maintenance. These defects include faulty design, areas where the logic is weak, wrong data processing, etc. The decline in consumer satisfaction may be traced back to all of these factors. In this paradigm, problems are prioritized by severity, and remedial and preventative measures are implemented accordingly. We'll be putting seven different machine learning algorithms through their paces using data from NASA's public promise repositories including CM1, KC1, PC1, and KC2. In order to tackle categorization issues, the compared machine learning algorithms engage in supervised learning.
These are two examples of classifier trees: To sum up: (i)Bagging (ii)Random Forests (RF) Multilayer Perceptron (MLP) and Radial Basis Function (RBF) are two methods used in neural networks. There are two distinct Bayesian classifier methods: One discriminative classifier Support Vector Machine (SVM) (i)Naive Bayes (ii)Multinomial Naive Bayes. Our findings will help those who use this software choose the most effective algorithms for moving on with their own projects.

Keywords: *Software Quality Metrics, Software Defect Prediction, Software Fault Prediction, Machine Learning Algorithms* .

INTRODUCTION

The process of developing a software system is a laborious one that involves careful planning, analysis, design, implementation, testing, integration, and ongoing maintenance. It is the responsibility of a software engineer to construct a software system within a certain budget and on time, both of which are set during the planning phase of the project. During the course of development, there is a possibility that there may be some defects, such as faulty design, bad functional logic, improper data handling, incorrect coding, and so on. These flaws have the potential to produce mistakes, which in turn can lead to rework, increases in development and maintenance costs, and a loss in customer satisfaction.

By keeping track of these flaws, the quality of the software may be improved via the use of a strategy known as defect management. In this method, flaws are rated according to their severity, and repair and preventative activities are carried out in accordance with the level of severity assigned to each defect.

According to the findings of recent studies, "defect prevention" tactics are being adopted in the place of "defect detection" strategies in contemporary methodologies. It is an expensive endeavor to use defect prevention measures with the goal of reducing the number of faults generated throughout the process of

software development. It involves more work and ultimately results in increased expenditures for the project. Because of this, finding bugs in the software within the early stages of the project's life cycle is quite important. The use of machine learning techniques, one of which is the binary prediction model, makes it possible to identify modules in a software system that are prone to defects before a failure occurs during the process of software development. The purpose of this study is to evaluate the software defect prediction performance of seven machine learning algorithms by making use of quality metrics such as accuracy, precision, recall, and F-measure associated with defects as an independent variable. Additionally, our objective is to find the best category while comparing the software defect prediction performance of these machine learning algorithms within the context of four NASA datasets obtained from the public PROMISE repository. The chosen machine learning algorithms for this comparison are put to use for supervised learning, with the end goal of solving classification issues.

There are two techniques that are tree-structured classifiers: (i) Bagging and (ii) Random Forests (RF); two techniques that are neural networks: (i) Multilayer Perceptron (MLP) and (ii) Radial Basis Function (RBF); two techniques that are Bayesian classifiers: (i) Naive Bayes and (ii) Multinomial Naive Bayes; and one discriminative classifier called Support Vector Machine (SVM). The remaining parts of the paper are divided into the following sections: The related work is discussed in Section 2 in a general sense, while the experimental approach is described in Section 3 in more depth. The results of the experimental investigation are summarized in Section 4, along with some suggestions for new lines of inquiry that may be pursued in the future.

Numerous research have been conducted to create and use statistical and machine learning based models for defect prediction in software systems. These models have been employed in a wide range of contexts. Basili have utilized logistic regression in order to investigate the influence that the suite of object-oriented design metrics has on the prediction of fault-prone classes. This was done in order to find out more information. Khosh gofta has evaluated the performance of a neural network in comparison to a non-parametric discriminant model in order to determine whether or not the modules of big telecommunication systems are prone to faults. The findings of their research indicated that the prediction accuracy of the neural network model had a superior outcome when compared to the non-parametric discriminant model. They conducted a case study by using regression trees in order to categorize fault-prone modules of huge telecommunication systems. Bayesian Belief Network has been used by Fenton in order to locate software flaws. However, this technique for machine learning suffers from a wide variety of drawbacks, many of which were brought to light by Weaver (2003).

LITERATURE SURVEY

The work that has been done up to this point makes use of 17 distinct data sets for the various algorithms that have been employed, such as SVM, KNN, and CART. These data sets are used to train the models that will be used in the future. However, despite the fact that these methods are used throughout the process of model development, it was found that the results that were generated by the models were incorrect. One of the methods of supervised learning that is used on a regular basis is called the Support Vector Machine, which is more often known to by its acronym, SVM. When implemented in the suitable manner, it may be used to solve issues relating to regression as well as classification. The objective of the Support Vector Machine (SVM) approach is to find the optimal line or decision boundary that may divide an n-dimensional space into different classes. This may be achieved by sectioning off several areas within the overall room. As a result of this, in the future we will be able to effortlessly place any newly obtained data points in the appropriate category.KNN is a straightforward method of supervised machine learning (ML) that may be put to use in the process of working on

classification or regression. Additionally, it is often used in the process of missing value imputation. The KNN algorithm is used in a variety of different kinds of programs. The KNN algorithm is a great illustration of an unsupervised machine learning (ML) method.It is also known as a lazy learner algorithm because, rather than learning immediately from the training set, it saves the dataset, and then, when the time comes for classification, it executes an action on the dataset. This is why it is known as a lazy learner algorithm. Because of this, it is also sometimes referred to as the lazy learner algorithm.A technique for making predictions that goes by the name of the CART algorithm is often used in the study of machine learning. It presents an explanation of how the values of the target variable may be expected based on other circumstances, and it does so by giving the explanation in the form of a model. This gives an explanation of how the values of the target variable can be anticipated based on other circumstances. All of the information that relates to the product has been stored. It acts as a gateway through which the management of orders, catalogs, and customers may be accomplished. It prepares the information that the user will see on the site, including the data on the items, the information on the categories, and the information on the site itself.It is an algorithm that is essential to machine learning, and it gives users a wide variety of application options from which to choose. Leo Breiman, a statistician, came up with the word "Decision Tree algorithms" to define Decision Tree algorithms, which may be utilized for classification or regression predictive modeling challenges. Leo Breiman came up with the name "Decision Tree algorithms." Breiman was also responsible for the development of the phrase "Decision Tree algorithms."

PROPOSED SYSTEM

The problem of class misbalancing, which leads to a drop in the performance of defect prediction, is addressed by the recommended system, which is comprised of SVM, Multilayer Perceptron, Run Bagging algorithm, Naive Bayes algorithm, Random Forest algorithm, Multinomial NB, and Radial Basis Functions. This system also addresses the problem of class misbalancing, which leads to a drop in the performance of defect prediction. This approach also solves the issue of class misbalancing, which is a contributing factor to the decline in the performance of defect prediction. This loss in performance may be directly attributed to the fact that the class does not have enough balance. These procedures have been included into the system in the hope that they may one day be useful in resolving the issue. The dataset was trained and then spat out in accordance with the restrictions, and the accuracies were provided in order to facilitate an investigation into the extent to which various algorithms are able to predict the amount of errors. This was done in order to determine, out of the myriad of different methods, which one produced the most accurate findings. proposed Software components will always have errors as a result of poor coding, which may lead to an increase in the cost of software development and maintenance. This is a problem since software components are essential to the operation of a software system. In addition to that, one direct consequence of this problem is that there will be unhappy consumers as a result of the service that they get. However, machine learning algorithms are currently experiencing a popularity spike that is currently experiencing a parabolic spike as a direct consequence of their improved performance. There have been several methods developed as a way of detecting defects in software components; however, machine learning algorithms are the method that is currently experiencing the popularity spike. This is a direct consequence of the fact that machine learning algorithms perform better than their human-based counterparts. The increased performance that is supplied by machine learning algorithms is the key reason for their rising appeal, which is why they are becoming more popular. This gain in popularity is a direct result of this enhanced performance that is offered by machine learning algorithms.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 4, April : 2023

The expected model is used in order to carry out the purpose of carrying out the assessment of the performance metrics, and this is done via the usage of the anticipated model.

Our personnel has access to a wide variety of data sets from which they may make their selection. Our investigation, on the other hand, takes use of datasets that were put together by NASA. These sets of data have been made accessible to us.

The amount of potential problems that may be caused by software bugs is used to determine the severity rating that is assigned to each bug. It is possible to protect oneself by taking preventative measures with reference to the algorithm that is used. Provides Better Outcomes. Finding faults in the project's early stages will pay off in the form of enhanced client loyalty in the project's later stages.

SAMPLE RESULTS

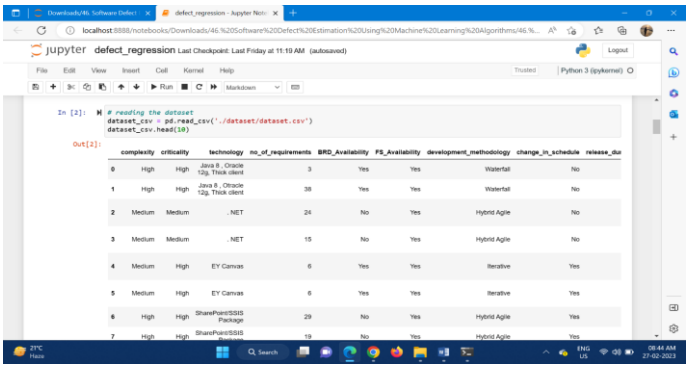

Figure 1:Loading dataset

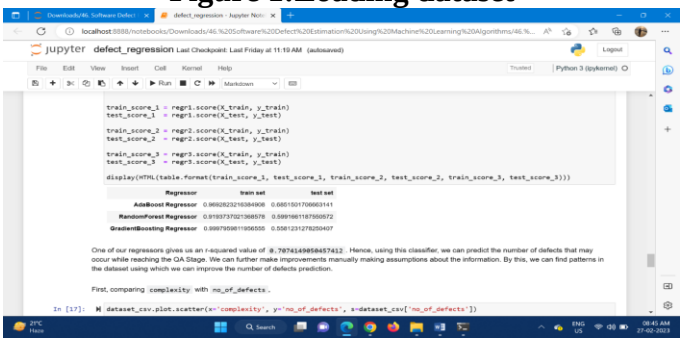

Figure 2:Regressor Score

CONCLUSION

In this experimental study, we use seven machine learning algorithms to predict software system defects before they are released into the real environment and/or delivered to customers, and we compare these algorithms using software quality metrics like accuracy, precision, recall, and F-measure to identify the best category that has the most capability to predict the software defects. In this experiment, we use the PC1, CM1, KC1, and KC2 datasets from NASA. These data sets are accessed via the open PROMISE database. This experiment shows that tree-structured classifiers, or ensemble learners like Random Forests and Bagging, perform better than their competitors when it comes to predicting the presence of defects. In particular, Bagging's capacity to foresee software defects is superior. Bagging's overall accuracy, precision, recall, and FMeasure fall within the ranges of 83.7–94.1%, 81.3–93.1%, 83.7–94.1%, and 82.4–92.8% across all datasets.When compared to other machine learning methods on the PC1 dataset, Bagging achieves the best results across the board. However, for the CM1 dataset, Bagging exceeds Naive Bayes in accuracy and recall and Naive

Bayes in precision and F-Measure. For the KC1 dataset, Random Forests provides the highest quality across all quality criteria. Finally, MLP performs better than any other machine learning approach on all KC2 dataset quality measures. The findings show that tree-structured classifiers are superior at predicting software defects. In addition, owing to its performance, tree-structured classifiers are suggested for use by software businesses in software fault prediction. Software testing and maintenance budgets may be reduced by using these methods to spot problems early on and implement fixes before they spiral out of control. One potential avenue for future research would be to conduct more experimental experiments utilizing other data sets. Datasets like this might be acquired from software developers or public repositories. An experimental investigation utilizing deep learning algorithms in addition to these machine learning techniques is the second potential path for future research. Another potential path for future research is the creation of new qualities via the combination of existing traits. In conclusion, a case study using separate software quality datasets acquired from actual projects of software businesses of varying sizes would be a sensible course of action.

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