DOI: https://doi.org/10.46632/daai/2/1/1

Data Analytics and Artificial Intelligence
Vol: 2(1), 2022
REST Publisher; ISBN: 978-81-948459-4-2
Website: http://restpublisher.com/book-series/data-analytics-and-artificial-intelligence/

A Study On Software Engineering Defect Prediction

[*]V S. Prasad, K. Sasikala
Department of Computer Science and Engineering, Vinayaka Mission's Kirupananda Variyar Engineering College, (Vinayaka Mission Research Foundation (Deemed to be University)), Salem.
[*]Corresponding author Email: prasadvs_83@yahoo.com

Abstract. The success of any software system entirely depends on the accuracy of the results of the system and whether it is without any flaws. Software defect prediction problems have an extremely beneficial research potential. Software defects are the major issue in any software industry. Software defects not only reduce the software quality, increase costing but it also suspends the development schedule. Software bugs lead to inaccurate and discrepant results. As an outcome of this, the software projects run late, are cancelled or become unreliable after deployment. Quality and reliability are the major challenges faced in a secure software development process. There are major software cost overruns when a software product with bugs in its various components is deployed at client s side. The software warehouse is commonly used as record keeping repository which is mostly required while adding new features or fixing bugs. Many data mining techniques and dataset repository are available to predict the software defects. Bug prediction technique is an important part in software engineering area for last one decade. Software bugs which detect at early stage are simple and inexpensive for rectifying the software. Software quality can be enhanced by using the bug prediction techniques and the software bug can be reduced if applied accurately. Dependent and independent variable are considered in Software bug prediction. To prevent defect based on software metrics software prediction model are used. Metrics based classification categorize component as defective and non-defective.

1. Introduction

The success of any software system is entirely depend on the accuracy of the results of the system and whether it is without any flaws, software defect prediction problems have an extremely beneficial research potential. Caper, investigated the software bugs or defects have rendered major contributions towards technical explanations for software project failure. Mining of software repositories have several research challenges to be addressed. New software bug prediction models need to be designed, effective software defect metrics need to be synthesized and provided them as inputs to various data mining techniques for extracting classified information in order to envisage the software faults in new software versions and also more developed methods are needed to reduce software cost overruns. The software metrics lie at the core of bug prediction models. The main objective of any organization is to have defect free software. In fact, earlier detection of defect would save time and cost of the system. The above statement clearly describes that we need to identify the finest machine learning model for software defect prediction for which various performance parameters are available, such as, accuracy, mean square error, and correlation and R-Squared to compare with other diverse models. Data is an important part of the system. One of the biggest challenges is to acquire the right dataset and where in to categorize the dependent and independent variables. The more the data, the more complex will the system become and more probability of the defect appearing. Hence, it is always safer to remove the insignificant variables from the dataset and reduce the independent variable using the feature selection technique. The insignificant variables have a negligible impact to detect the software bug. There are various types of feature selection techniques that are available to derive the significant and insignificant variables from the dataset. For the research, the Wrapper and Filter method used in Feature Selection technique are taken to find the imperative variable from the publicly available Promise Repository. Data Mining Data mining addresses all the techniques and processes involved in discovering interesting patterns that are hidden inside large data sets, which help in decision making process. Data mining is not only applicable to marketing data, drug designing and weather forecasting, but also has been used by the software development industries to manage their software development processes. Data Mining Techniques Data mining techniques, such as, clustering, classification, association rules and various statistical techniques are involved to extract actionable information from huge data sets. A large number of data mining practices have been developed for bug detection, prediction and prevention. To accomplish the data mining task various software tools are available to analyze enormous quantities of data and apply diverse data mining techniques Stored software engineering data contains adequate information in terms of project status, progress, and evolution. This data supports various characteristics of software development within the industrial software development process. Using firm and well known data mining techniques, researchers and specialists have started exploring their precious data in order to manage their projects in an improved and professional manner and produce high quality software systems. The early stage of software defect discovery allows managers to make appropriate decisions and plan limited project resources in a more structured and systematic manner

2. Significance of Study

Software development is facing enormous challenges in quality and reliability. There are major software cost overruns when a software product with bugs in its various components is deployed at client side. A software bug is an inaccuracy, defect, botch up, or a error in a computer program or system that triggers it to generate an improper or unanticipated result, or to perform in an unintentional manner. As a result, software projects will be delayed cancelled or become unreliable after deployment. There are varied social and technical issues that are related to software failures. Social issues such as forcing project to adhere to schedules or handling of the software project by inexperienced professionals are common in software industries. A total absence of contemporary approximating techniques and the failure in planning for requirements growth during the development stage is amongst the technical issues that contribute to the failure of the project. “Cost per defect” is harmful for the software quality. The cost-prefect-metric- is used to scrutinize the monetary assessment of the software quality. The financial significance of software quality is dependent on two major 9 factors:
1) Reduction in defect repair costs;
2) Reduction in development and maintenance costs. The first of these factors is managed in an inaccurate fashion by the cost-per-defect metric. A serious economic analysis of the software quality needs additional metrics besides cost per defect and better measurement methods as well Successful software quality control involves defect prevention, defect removal, and defect measurement activities. Defect prevention includes all activities that minimize the probability of creating an error or defect in the first place. The gauging of defect consists of various matrices of malfunctions detected during the development phase and includes the shortcomings that are pointed out by the customer after its release. The removal of defects takes care of all the activities that detect and eliminate flaws and mistakes in any type of deliverable product. Software metrics for example, product metrics and process metrics are at the core of bug prediction models. Objectives of The Study to select statistical learning and data mining techniques to be applied on historical software data for prediction of software defect.

3. Literature Review

Prasad, Florence and Arya (2015) in their study has depicted that Software quality is a field of study and practice that describes the desirable attributes of software products. Software quality metrics are a subset of software metrics that focus on the quality aspects of the product, process, and project. The software defect prediction model helps in early detection of defects and contributes to their efficient removal and producing a quality software system based on several metrics. The main objective of paper was to help developers identify defects based on existing software metrics using data mining techniques and thereby improve the software quality. In their paper, various classification techniques are revisited which are employed for software defect prediction using software metrics in the literature. Vashisht, Lal and Suresh chandar (2015) in their research paper has pointed out the fact that number of approaches has been proposed for effective and accurate prediction of software defects, yet most of these have not found widespread applicability. The main objective in this communication is to provide a framework which is expected to be more effective and acceptable for predicting the defects in multiple phases across software development lifecycle. The proposed framework is based on the use of neural networks for predicting defects in software development life cycle. Further, in order to facilitate the easy use of the framework by project managers, a software graphical user interface has been developed that allows input data (including effort and defect) to be fed easily for predicting defects. The proposed framework provides a probabilistic defect prediction approach where instead of a definite number, a defect range (minimum, maximum, and mean) is predicted. The claim of efficacy and superiority of proposed framework is established through results of a comparative study, involving the proposed framework and some well-known models for software defect prediction. Arvinder Kaur, Kamaldeep Kaur and Chopra (2016) in their research article has depicted that there are many approaches for predicting bugs in software systems. 22 A popular approach for bug prediction is using entropy of changes as proposed by Hassan (2009). Their paper uses the metrics derived using entropy of changes to compare five machine learning techniques, namely Gene Expression Programming (GEP), General Regression Neural Network, Locally Weighted Regression, Support Vector Regression (SVR) and Least Median Square Regression for predicting bugs. Four software subsystems: mozilla/layout/generic, mozilla/layout/forms, apache/httpd/modules/ssl and apache/httpd/ modules/ mappersare used for the validation purpose. The data extraction for the validation purpose is automated by developing an algorithm that employs web scraping and regular expressions. The study suggests GEP and SVR as stable regression techniques for bug prediction using entropy of changes. Halili, and Rustemi (2016) has pointed out that with the rapid development of technology, there are various sophisticated software which enable them to solve problems in various spheres of our life. With the introduction of sophisticated software, there is a need also for new terms where it will be stored these data because we know that software cannot function if it does not have the most significant part and that is database. They have introduced the terms Big Data, Data Warehouses, Data Mining and their classification and done analysis for Regression technique (linear and multiple regression). Regression as technique although is predictive technique, but based on analyzes conducted to reach the conclusion most scientists, have concluded that the reliability percentage is around 95%. Through their paper they have demonstrated this scale of reliability through c examples Rong, Li and Cui (2016) in their article has pointed out that software defect prediction is not only crucial for improving software quality, but also helpful for software test effort estimation. It has been observed that 80% of the fault happens in 20% of the modules. Therefore, there is a need to find out the most error prone modules accurately and correct them in time to save time, money and energy. There is one method that is Support vector machine (SVM) which is an advanced classification method that fits the defection classification. However, studies show that, the value of parameters of SVM model has a remarkable influence on its 23 classification accuracy and the selection process lacks theory guidance that makes the SVM model uncertainty and low efficiency. In their paper, a CBA-SVM software defect prediction model is proposed, which take advantage of the nonlinear computing ability of SVM model and optimization capacity of bat algorithm with centroid strategy (CBA). Through the experimental comparison with other models, CBA-SVM is proved to have a higher accuracy. Periasamy and Mishbahulhuda (2017) have mentioned in their paper that Software defect prediction work focuses on the number of defects remaining in a software system. The software defect prediction model helps in early detection of defects and contributes to their efficient removal and producing a quality software system based on several metrics. A prediction of the number of remaining defects in an inspected is fact can be used for decision making. An accurate prediction of the number of defects in a software product during system testing contributes not only to the management of the system testing process but also to the estimation of the product's required maintenance. Defective software modules cause software failures, increase development and maintenance costs, and decrease customer satisfaction. It strives to improve software quality and testing efficiency by constructing predictive models from code attributes to enable a timely identification of fault-prone modules. The main objective of paper is to help developers identify defects based on existing software metrics using data mining techniques and thereby improve the software quality. In this paper, we will discuss data mining techniques that are association mining, classification and clustering for software defect prediction. This helps the developers to detect software defects and correct them.

4. Tools and Techniques

To accomplish the research goal, the data is collected from open software repositories. Data in these repositories provide us with historic information, which is stored at the time of development of the software and testing of the software products. The large number of open software repositories, such as, source control repositories, bug repositories, achieved communication repositories, deployment logs and code repositories are publically available. The finest part of these repositories is that they are a treasure house of actual software development and approved for the international software communities for standard researches in software intelligence. Hassan et.al. Mentioned the following table which consists of information about software development.

5. Data Collection

Data plays a very critical role in any organization and hence the selection of data (or dataset) is an extremely important component for any software system. The data has to be in a reliable, clean and accurate form to enable a developer to reach to some conclusion. The bug prediction dataset is a group of models and metrics of software system and their histories. The dataset is a dynamic component to accomplish bug prediction at the class level, as it uses a number of metrics, which can be used to create generalized linear regression models and the number of post-release defects. The performance of these models is evaluated by comparing the prediction results against the actual post-release defects provided as part of the dataset.

6. Data Cleaning

Data Cleaning is supportive in improving the quality of the data as it deals in detecting inconsistencies, removing errors, missing values. Rahm et.al in their paper addresses the issue of data cleaning, which is a foremost part of 34 extractions, transformation, loading (ETL) process in a data warehouse. There are a variety of tools available to clean the data, but at times, a major portion of the data needs to be cleaned manually which are tough to write and maintain. Though various types of tools are available, yet there is a complexity in a cleaning problem. Feature Selection in a larger dataset, all the variables are not so important to consider, the more the number of variables, the complexity will be on the increase. Therefore, it is always desirable to reduce the variables and should include important variables in a dataset. Through a Feature Selection technique, we can reduce the variable and locate the importance of the variable in a dataset.

7. Boruta

Boruta is one of the most significant Feature Selection packages to explore the relevant element from a large dataset. It uses a Wrapper algorithm, which is better than the filter method as in the Wrapper Method classifier is used as a black bSox returning a feature ranking. The R package Boruta is available at http://CRAN.R-project.org/package=Boruta). When the R package Bouruta is in operation, it displays the entire confirmed variable and the rejected variable in a dataset. When the box plot is drawn Green, Blue and Red represent a Z- Scores of confirmed, minimal or average, rejected attribute respectively.

8. Regsubsets

Regsubsets are used for regression subsets selection to come across the model that ideally suits the data by calculating its R2, AIC and BIC values. The model can be ranked according to adjusted R2 criteria and BIC, when the graph is being plotted as shown in Figure 4.2, there are two indicators, black and white. Black indicates that the variable is included in the model and white signifies that it is excluded in the model.

9. Fselector

FSelector is also considered to be one of the important selection functions for selecting the attributes from a dataset. This function is used to find the irrelevant and redundant attribute as much as possible from a given dataset. F Selector consists of Feature Selection algorithm, like wrapper and filter. The Wrapper method uses a predictive model and trains a new model for each subset. The Wrapper Algorithm used in F Selector is best first search, backward search, forward search, hill climbing search. The Filter Method uses a proxy measure to score a feature sub-set. Filters are usually less computationally intensive than wrappers. It is generally used as feature ranking. Filter methods have also been applied as a pre-processing step for wrapper methods. Filter method used in F Selector are CFS, chi squared, information gain, gain ratio, symmetrical uncertainty, linear correlation, rank correlation, one R, relief, consistency, random forest importance. Other algorithma used are cutoff. k, cutoff. k. percent, cutoff. biggest. Diff as. simple. formula. Filter method Random Forest, Information Gain, Linear Correlation and Rank Correlation were used.

10.Data Analysis

The regression methods, Linear Model, Random Forest, Decision Tree, Support Vector Machine, Neural Network and Decision Stump were applied in software modules Ant, Ivy, Tomcat, Berek, Camel, Lucene, POI, Synapse and Velocity with all the metrics and with only optimal metrics like RFC, LOC and WMC, which was derived from Feature Selection techniques. Performance Parameters Accuracy, Mean Square Error, R Squared and Correlation were calculated to find the most optimal machine learning models which are as follows: Machine learning models applied combining all the Software Metrics The machine learning models with the tuning parameters as discussed in Table 1 was applied with all the software metrics WMC, RFC, CBO, LCOM, NOC, DIT, CA, CE, NPM, LCOM3, LOC, DAM, MOA, MFA, CAM, IC, CBM, AMC, MAX_CC, AVG_CC to obtain the correlation, R-Squared, Mean Square Error and Accuracy as shown in Tables 2, 4, 4 and 5 below respectively. The graph was plotted to compare the performance parameters by computing the mean average of each nine software modules with respective.

TABLE 1. Correlation Calculated Combining of All the Software Metrics

Machine Learning Model	**Ant**	**Ivy**	**Tomcat**	**Berek**	**Camel**	**Lucene**	**Synapse**	**Velocity**
Linear Model	0.69	0.39	0.45	0.51	0.05	0.37	0.14	0.19
Random Forest	0.64	0.56	0.43	0.73	0.16	0.48	0.25	0.25
Neural Network	0.24	0.47	0.47	0.1	0.01	0.43	0.01	0.35
Decision Tree	0.53	0.12	0.47	0.65	0.03	0.35	0.14	0.5
SVM	0.47	0.38	0.1	0.87	0.07	0.45	0.16	0.09
Decision Stump	0.49	0.5	0.23	0.71	0	0	0.12	0

The Correlation Comparative Analysis was done of the machine learning model using all the software metrics. Figure 5.26 depicted that Random Forest has the highest correlation at 0.44 and Neural Network and Decision Stump have lowest correlation at 0.29 using all the software metrics.

TABLE 2. R Squared Calculated Combining All the Software Metrics

Machine Learning Model	**Ant**	**Ivy**	**Tomcat**	**Berek**	**Camel**	**Lucene**	**Synapse**	**Velocity**
Linear Model	0.48	0.14	0.23	0.26	0.05	0.37	0.32	0.25
Random Forest	0.41	0.31	0.18	0.53	0.16	0.48	0.23	0.24
Neural Network	0.06	0.23	0.23	0.01	0.01	0.432	0.36	0.16
Decision Tree	0.3	0.02	0.01	0.43	0.4	0.32	0.04	0.18
SVM	0.22	0.14	0.3	0.45	0.07	0.33	0.14	0.5
Decision Stump	0.23	0.45	0.13	0.45	0.43	0.322	0.16	0.09

The R-Squared Comparative Analysis was done of the machine learning model using all the software metrics. The Figure 5.27 depicted that the Random Forest has highest R-Squared value as 0.33 and Neural Network has the lowest R Squared values as 0.18.

TABLE 3. Mean Square Error Calculated Combining All the Software Metrics

Machine Learning Model	Ant	Ivy	Tomcat	Berek	Camel	Lucene	Synapse	Velocity
Linear Model	0.23	2.26	0.34	0.35	1.56	0.06	1.99	0.63
Random Forest	0.5	0.22	0.19	1.44	0.78	1.4	0.77	0.53
Neural Network	2.24	0.35	1.76	2.11	0.87	1.34	1.85	3.22
Decision Tree	0.51	0.17	0.13	1.24	0.55	.1.65	0.77	1.33
SVM	0.43	0.14	0.27	1.33	0.67	1.65	0.77	0.96
Decision Stump	0.55	0.24	0.23	1.65	0.67	1.88	0.99	1.02

The Mean Square Error comparative analysis was done of the machine learning model using all the software metrics. Figure 5.28 depicted that the Random Forest and Support Vector Machine has the lowest mean square value as 0.74 and the Neural Network has the highest Mean Square Error as 1.7.

Table 4. Accuracy calculated combining all the software metrics

Machine Learning Model	Ant	Ivy	Tomcat	Berek	Camel	Lucene	Synapse	Velocity
Linear Model	87.45	94.33	96.54	65.46	45.34	75.99	85.45	73.33
Random Forest	87.44	94.33	34.35	76.65	48.66	54.45	76.87	72.21
Neural Network	34.34	97	34.56	40.54	77.54	45.67	41.34	77.6
Decision Tree	87.65	97.65	95.44	87.43	56.44	80.54	65.76	65.87
SVM	97.65	97.55	97.45	73.33	88.55	55.46	81.43	76.45
Decision Stump	87.65	96.45	94.33	77.33	98.33	80.43	83.23	84.34

The Accuracy Comparative Analysis was done of the Machine Learning Model using all the software metrics. Figure 5.29 depicted that the Support Vector machine has the highest value as 83.04 and the Neural Network has the lowest accuracy as 53.13

11.Conclusion

The objective of the software bug prediction is tracing of bug constituents in software well before software testing. The bug prediction techniques were applied in all the software modules of CK and OO metrics. Bug prediction results in decreased cost of development costs, time, increased customer satisfaction and more reliable software. Therefore, bug prediction techniques are considered significant in order to accomplish important to achieve optimum quality of and refrain from repeating past mistakes. The dataset used in this paper is from the Promise Repository, which is well-recognized and a publicly accepted repository. The experiment was conducted using software modules, such as Ant, Ivy, Tomcat, Berek, Camel, Lucene, POI, Synapse and Velocity using Feature selection techniques: Boruta, regsubsets and FSelector (Random Forest, Information Gain, Linear Correlation, and Rank Correlation).The objective of the Research was achieved by locating the optimal software metrics, optimal machine learning models and preventive measures that can be used to prevent defects in future software version releases Measures To Prevent Defect In Software Release The earlier the defect is detected, the cost involved is reduced and the resources are fully utilized. Moreover, it becomes much easier to rectify the defect during the initial stage. Complexity metrics are better predictors of fault potential in comparison to other well-known historical predictors of faults, i.e. prior modifications and prior faults. iii. Self-review the code has a major contribution in preventing the software defect. Proper Defect Tracking system needs to be implemented. Files that have anti-patterns tend to have a higher density of bugs than others as anti-patterns can increase the bugs in the future. Anti-patterns can be removed from systems using refactoring.

References

[1]. H. Solanki, “Comparative Study of Data Mining Tools and Analysis with Unified Data Mining Theory,” International Journal of Computer Applications, vol. 75, no. 16, pp. 23–28, 2013.

[2]. E. Hassan and Tao, Xie, “Mining software engineering data”, in Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2 (ICSE '10), Vol. 2. ACM, New York, NY, USA, 2010. pp. 503-504.

[3]. M. S. Rawat, and S. K. Dubey, "Software defect prediction models for quality improvement: A literature study." IJCSI International Journal of Computer Science Issues Vol.9 No. 5,pp. 295, 2012.

[4]. M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," Proc. 6th Int. Conf. Predict. Model. Softw. Eng. - PROMISE '10, pp 1-10, 2010.

[5]. Y. Suresh, J. Pati, and S. K. Rath, "Effectiveness of software metrics for object-oriented system," Procedia Technologyvol. 6, pp. 420–427, 2012.

[6]. M. S. Rawat, and S. K. Dubey, "Software defect prediction models for quality improvement: A literature study." IJCSI International Journal of Computer Science Issues Vol.9 No. 5,pp. 289, 2012.

[7]. M. S. Rawat, and S. K. Dubey, "Software defect prediction models for quality improvement: A literature study." IJCSI International Journal of Computer Science Issues Vol.9 No. 5,pp. 292, 2012

[8]. S. Kim, H. Zhang, R. Wuand L. Gong. "Dealing with noise in defect prediction "in Proceedings of the 33rd International Conference on Software Engineering (ICSE '11). ACM, New York, NY, USA, 2011. pp. 481-490.

[9]. M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data Quality: Some Comments on the NASA Software Defect Datasets," IEEE Transactions on Software Engineering, vol. 39, no. 9, pp. 1208–1215, 2013.

[10]. M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," Proc. 6th Int. Conf. Predict. Model. Softw. Eng. - PROMISE '10, pp 2-4, 2010.

[11]. R. Goyal, P. Chandra, and Y. Singh, "Identifying influential metrics in the combined metrics approach of fault prediction," Springerplus, vol. 2, no. 1, pp. 1–8, 2013.

[12]. R. Subramanyam and M. Krishnan, "Empirical analysis of CK metrics for object-oriented design complexity: implications for software defects," IEEE Transactions on Software Engineering, vol. 29, no. 4, pp. 297–310, 2003.

[13]. F. Provost and R. O. N. Kohavi, "Guest Editors' Introduction: On Applied Research in Machine Learning," New York, vol. 132, no. 1998, pp. 127–132, 1998.

[14]. Pradesh and A. Pradesh, "The Importance of Statistical Tools in Research Work," Int. J. Sci. Innov. Math. Res., vol. 3, no. 12, pp. 50–58, 2015.

[15]. T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schröter, and C. Weiss, "What makes a good bug report?," IEEE Trans. Softw. Eng., vol. 36, pp. 618–643, 2010.

[16]. H. Wang, "Software Defects Classification Prediction Based On Mining Software Repository," Dissertation, 2014.

[17]. M. Jureczko. "Significance of different software metrics in defect prediction," Softw. Eng. An Int. J., vol. 1, no. 1, pp. 86–95, 2011.

[18]. T.N. Zimmermann, N. Nagappan, and Zeller, A. "Predicting bugs from history software evolution". Springer Berlin Heidelberg, pp 69-88, 2008