wjert, 2020, Vol. 6, Issue 1, 256-265. *Original Article* ISSN 2454-695X

World Journal of Engineering Research and Technology

WJERT

www.wjert.org SJIF Impact Factor: 5.924

STUDY ON SOFTWARE DEFECT PREDICTION USING DATA MINING TECHNIQUES

***[1]Rakesh Kumar, [2]Dr. Dharmendra Chourishi, [3]Prof. Anurag Srivastava**

[1]M. Tech. Scholar, CSE, NRI College Bhopal.

[2]A.P., NRI College Bhopal.

[3]Asso. Prof. CSE, NIIST Bhopal.

Article Received on 08/12/2019 **Article Revised on 29/12/2019** **Article Accepted on 19/01/2020**

***Corresponding Author**
Rakesh Kumar
M. Tech. Scholar, CSE,
NRI College Bhopal A.P.,
NRI College Bhopal.

ABSTRACT

Software defect prediction has been one of the key areas of exploration in the domain of software quality. Software bug is a major problem arises in the coding implementation .There are no satisfied result found by project development team. The software bug problems mentationed in problem report and software engineer does not easily detect this software defect but by the help of data mining classification software engineers can easily classify software bug. The challenges encountered are difficulty in separating correct theories from the incorrect ones when the purpose of evaluation is in practice and difficulties in the identification of quality literature from quality lacking literature. Using data mining techniques, one can uncover hidden patterns from this data, measure the impact of each stage on the other and gather useful information to improve the software development process. The insights gained from the extracted knowledge patterns can help software engineers to predict, plan and comprehend the various intricacies of the project, allowing them to optimize future software development activities. It has been also discussed that how data mining improves the software development process in terms of time, cost, resources, reliability and maintainability.

KEYWORDS: Defect, Software, Bug, Data Mining, Classification, Intricacies.

1. INTRODUCTION

Predicting defective code in the software development process is a key aspect of software analytics. A software defect is a bug, fault, or error in a program that causes improper outcomes. Software defects are programming errors that may occur because of errors in the source code, requirements, or design. Defects negatively affect software quality and software reliability.[1] Hence, they increase maintenance costs and efforts to resolve them. Software development teams can detect bugs by analyzing software testing results, but it is costly and time-consuming by testing entire software modules. As such, identifying defective modules in early stages is necessary to aid software testers in detecting modules that required intensive testing.[2,3] In the field of software engineering, software defect prediction (SDP) in early stages is vital for software reliability and quality.[1,4] The intention of SDP is to predict defects before software products are released, as detecting bugs after release is an exhausting and time-consuming process. In addition, SDP approaches have been demonstrated to improve software quality, as they help developers predict the most likely defective modules.[5,6] SDP is considered a significant challenge, so various machine learning algorithms have been used to predict and determine defective modules.[7] With the end goal of expanding the viability of software testing, SDP is utilized to distinguish defective modules in current and subsequent versions of a software product.

Software engineering data, such as defect prediction datasets, are very imbalanced, where the number of samples of a specific class is vastly higher than another class. To deal with such data, imbalanced learning approaches have been proposed in SDP to mitigate the data imbalance problem.[7] Imbalanced learning approaches include re-sampling, cost-sensitive learning, ensemble learning, and imbalanced ensemble learning (hybrid approaches).[7,39] Re-sampling approaches can be either oversampling and under-sampling methods, and these can add or remove instances from the training data only.

1.1 Data Mining

It is processes in computer science by which relationship and pattern from data can be easily extracted and information is collected that helps in decision making as required in software development field. It easily extracts information from problem reports and take decision by the help of information software defect can detect and software quality is improved.

1.2 Classification

Classifications have a training set which provide a facility to have a common level of same classes of data. Some different type bugs in software project development: SW-bug, document bug, duplicate bug and mistaken bug. These bugs have common level bug classes of data object known as software defect in training set.

1.3 Decision Tree

Decision tree is a classifier of root node which generates other branches as a node. The common attributes of data at class level each node have own information.

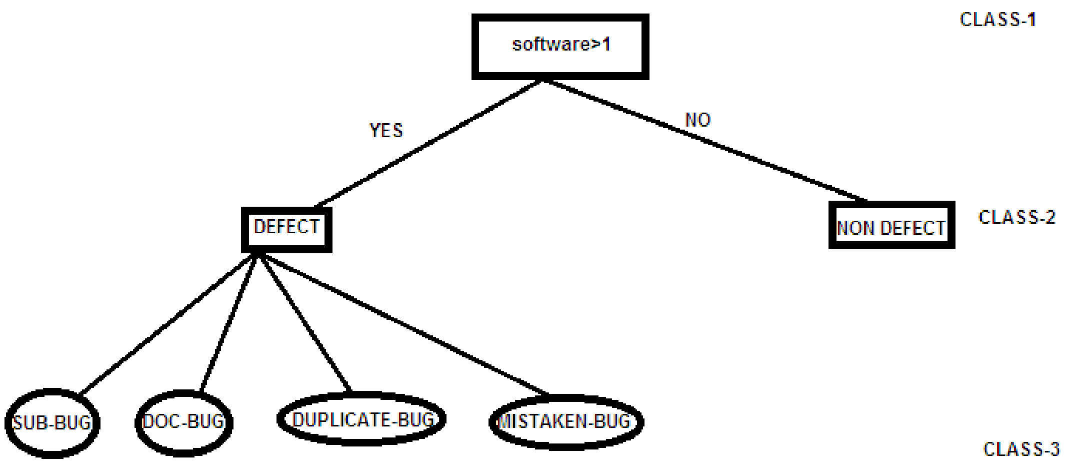

Fig. 1: Represents to check level defect of software.

1. **If software > 1** then root node extract another branches or internal node (not leaf node) show class (2).
2. **If software < 1** then shows root node on class (1).
3. **If software defect > 1** then found defect classification categories bug at class (3) not extract another node otherwise on the class (2).

2. LITERATURE REVIEW

Primary studies by definition correspond to the literature being mappings. To provide a strong mapping it is essential that selection of primary studies for mapping must be done carefully. While it is good that an exhaustive search is conducted for the selection of primary studies, in some cases it is not possible because of the number of primary studies available. In such cases the search criteria become important. For DeP studies, we can conduct an exhaustive search because the number of primary studies is not very large and since is concerned with those studies which are empirical in nature, the number shrinks further. We have selected the list of following digital libraries to perform the search:

1. IEEE Xplore
2. Springer Link
3. Science Direct
4. Wiley Online Library
5. ACM Digital Library
6. Google Scholar

The Search String: A search string is the combination of characters and words entered by a user into a search engine to find desired results. The information provided to the search engine of the digital library directly impacts the results provided by it. To ensure that all the primary studies that our mapping plans to address are covered we need to be careful in the selection and placement of keywords used in the search string.

Search string

Software.(defect + fault). (software metrics + object oriented metrics + design metrics)

Here, '.' corresponds to the Boolean AND operation, and '+' Corresponds to the Boolean OR operation. The search string was executed on all six electronic databases mentioned above and the publication year was restricted to the range 1995–2018. The literature hence obtained was processed further using a carefully designed inclusion-exclusion criteria and quality analysis criteria as described in the following sections.

The Inclusion-Exclusion Criteria: The search results obtained by execution of the search string may still fetch some primary studies that either do not add value to the mapping or do not fall within the purview of what the mapping aims to accomplish. Once all the primary studies have been obtained, a carefully designed inclusion-exclusion criteria are applied to the resultant set in order to eliminate entities that do not match the objectives of the mapping.

Inclusion Criteria

- Empirical studies for DeP using software metrics.
- Studies that provide empirical analysis using statistical, search-based and machine learning techniques.

Exclusion Criteria

- Literature Reviews and Systematic Reviews.
- Studies that do not use DeP as the dependent variable.
- Studies of non-empirical nature.
- If two studies by the same author(s) exist, where one is an extension of the previous work the former is discarded. But if the results differ, both are retained.

Review Committee: We formed a review committee that comprises of two Assistant Professors and two senior researchers to mapping in order to rate all primary studies captured from the search. All studies were examined independently on the basis of the criteria defined in The Inclusion-Exclusion Criteria. Application of the inclusion-exclusion criteria resulted in 98 studies out of the total 156 studies being selected for quality analysis.

Quality Analysis: Assessing the quality of a set of primary studies is a challenging task. A quality analysis questionnaire is prepared as part of this systematic mapping to assess the relevance of studies taking part in this mapping. The questionnaire takes into consideration suggestions given in Reference.[5] A total of 18 questions, given in Table 2, together form the questionnaire and each question can be answered as “Agree” (1 point), “Neutral” (0.5 points) and “Disagree” (0 points). Hence, a study can have a maximum 18 points and minimum 0 points.

The same review committee enforces the quality analysis questionnaire.

3. METHODOLOGY

The mapping method in this study is taken from Reference.[1] Figure 2 outlines the process diagram.

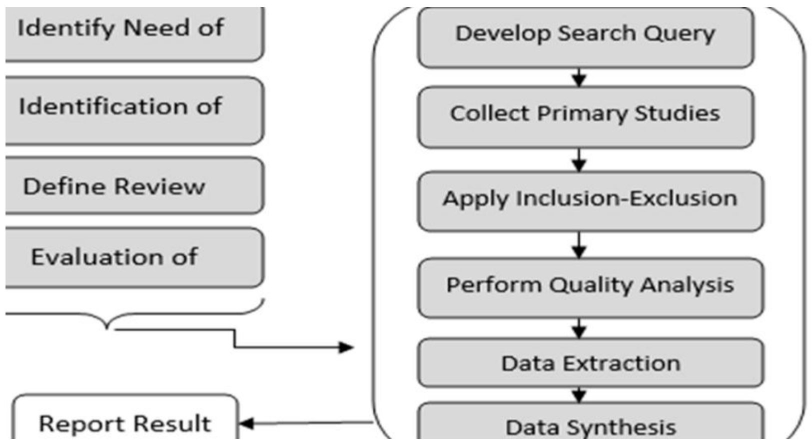

Fig. 2: The mapping process.

Data Synthesis: Data synthesis involves the accumulation of facts from the data collected during the data extraction process to build responses to the research questions.[5] In the data synthesis allows for reaching conclusive answers to the research questions identified as part of the systematic mapping. Details of the techniques used to answer the selected research questions are given below:

1. RQ1: To answer this question we use a bar chart that shows the number of studies using machine learning; search-based techniques, and statistical techniques and threshold-based techniques.
2. RQ2: This question has two parts. The first part is concerned with multi-co linearity analysis techniques and the second part deals with feature sub-selection techniques used. To answer this question, we make use of bar charts and tables. The bar charts show the number of studies using a particular technique for multi-co linearity analysis or feature sub selection. The tables show what techniques were used in which SE.
3. RQ3: This research question deals with the data used in SEs. It has three parts, the first part deals with the various datasets used for DeP studies, the second part deals with the independent variables found significant in the identified SEs, and the third part deals with the independent variables found insignificant in the selected SEs. The first part and the second part are answered with the help of a bar chart/pie chart combination while the third part does not use any diagramming method.
4. RQ4: The fourth question deals with the performance measures and statistical tests used in DeP studies. We use a combination of bar charts/pie charts to address this question.
5. RQ5: This question makes use of a bar chart. The bars are used to denote the comparative performance of a learning technique.
6. RQ6: This question is addressed using a table. Since the number of studies is limited, tables are used to summarize which study uses what search-based technique.
7. RQ7: This question uses bar charts and tables to show the distribution of studies that address security related defects and vulnerabilities.
8. RQ8: This question does not use any diagramming method.
9. RQ9: This question does not use any diagramming method.

4. RESULTS AND DISCUSSION

Table 1: Model building techniques used in DeP Studies.

Sl. No.	Class of learners	No. of studies
1	Bayesian Learners	42
2	Decision Tree	49
3	Discriminat Analysis	13
4	Neural Networks	26
5	Regression	40
6	SVM	26
7	Threshold	9

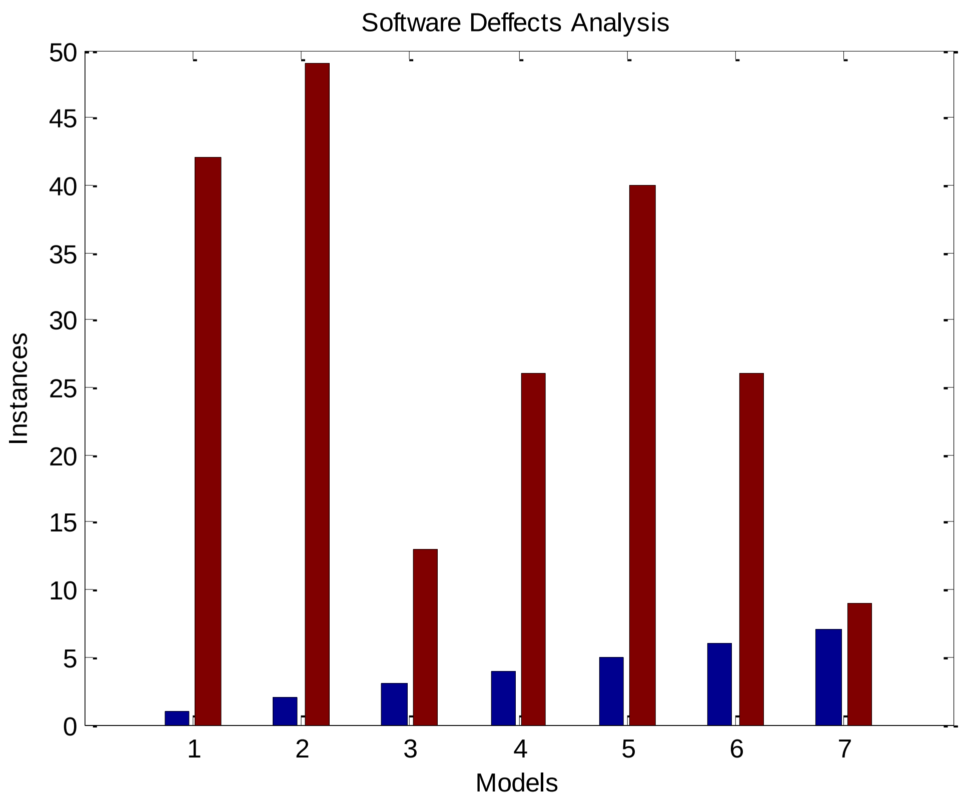

Figure 3: Learning methods used for DeP.

The most used learning method is the Decision Tree as shown in figure3, 44 out of 98 studies selected for this systematic mapping use some variant of the decision tree method. Techniques like C4.5, J48, CART, and Random Forest come under the decision tree class. Bayesian learners.[65] i.e., Naïve Bayes, Bayes Net, etc. have been used by 39 studies. Regression, Discriminant analysis and Threshold based classification have been performed in 35, 8 and 4 studies respectively. Both support vector machine and neural network have been used in 21 studies.

5. CONCLUSIONS

The research questions in this systematic mapping were constructed by taking into account the following definition of an ideal DeP model: "An ideal DeP model should be able to classify defects on the basis of severity, should detect security-related defects and system vulnerabilities and should have the ability to detect defects on systems on which it was not trained".

Results of this study showed that the existing literature has covered some of the parts of the DeP process fairly well, for example, the sizes of datasets used are large, and nearly all machine learning methods have been examined. But when taking into account the overall approach and effectiveness of DeP studies, there are a lot of shortcomings. Studies have not made much use of multi-co-linearity analysis techniques and only half of the studies selected for mapping have used feature sub-selection techniques.

6. Scope of future work

The following future guidelines are provided on the basis of the results of this study:

1. Datasets used for DeP studies should undergo thorough pre-processing that includes multi-co-linearity analysis and feature sub selection.
2. Most of the researches in defect prediction involve data obtained from open source software systems. Few studies use industrial datasets. It is important that industrial data be used for building defect prediction models so that the models can be generalized.
3. Future studies should make extensive comparisons between search-based techniques, machine learning techniques and statistical techniques.

REFERENCES

1. Sunita Tiwari and Neha Chaudhary, "Data mining and Warehousing" Dhanpati Rai and Co.(P) Ltd. First Edition, 2010.
2. J.R.Quinlan, "C4.5: programs for machine learning", Morgan Kaufmann, San Francisco, 1993.
3. M. Shepperd, C. Schofield, and B. Kitchenham, "Effort estimation using analogy," in of the 18th International Conference On Software Engineering, Berlin, Germany, 1996; 170- 178.
4. Alsmadi and Magel, "Open source evolution Analysis," in proceeding of the 22nd IEEE International Conference on Software Maintenance (ICMS'06), phladelphia, pa. USA, 2006.

5. Boehm, Clark, Horowitz, Madachy, Shelby and Westland, ”Cost models for future software life cycle Process: COCOMO2.0.” in Annals of software Engineering special volume on software process and prodocuct measurement, J.D. Arther and S.M. Henry, Eds, j.c. Baltzer AG, science publishers, Amsterdam The Netherlands, 1995; 1: 45-60.
6. Pal A. K., and Pal S., “Analysis and Mining of Educational Data for Predicting the Performance of Students”, (IJECCE) International Journal of Electronics Communication and Computer Engineering, 2013; 4(5): 1560-1565. ISSN: 2278-4209.
7. Ribu, Estimating, “Object oriented software projects With use cases”, M. S. thesis, University of Oslo Department of informatics, 2001.
8. Nagwani N. and Verma S., “Prediction data mining Model for software bug estimation using average Weighted similiarity,” In proceeding of advance Computing conference (IACC), 2010.
9. Hampherey Watts S., “A discipline for software Engineering reading”, Ma, Addison Wesley, 1995.
10. Zhou, Y.; Hareton, L. Empirical analysis of object-oriented design metrics for predicting high and low severity faults Software Engineering. IEEE Trans, 2006; 32: 771–789.
11. Hassan, A.E.; Holt, R.C. The top ten List: Dynamic Fault Prediction. In Proceedings of the 21st IEEE.
12. Le Hoang Son 1,2, Nakul Pritam 3, Manju Khari 4, Raghvendra Kumar 5, Pham Thi Minh Phuong 6 and Pham Huy Thong 7,8,* Empirical Study of Software Defect Prediction: A Systematic Mapping.
13. Beecham, S.; Hall, T.; Bowes, D.; Gray, D.; Counsell, S.; Black, S. A Systematic Review of Fault Prediction Approaches used in Software Engineering; The Irish Software Engineering Research Centre: Limerick, Ireland, 2010.
14. Catal, C.; Diri, B. A Systematic Review of Software Fault Prediction studies. Expert Syst. Appl., 2009; 36: 7346–7354.
15. Fawcett, T. An Introduction to ROC Analysis. Pattern Recognit Lett., 2006; 27: 861–874.
16. Hall, T.; Beecham, S.; Bowes, D.; Gray, D.; Counsell, S. Asystematic Literature Review on Fault Prediction Performance in Software Engineering. IEEE Trans. Softw. Eng., 2012; 38: 1276–1304. [International Conference on Software Maintenance, Budapest, Hungary, 26–29 September 2005 Report EBSE-2007-001; Keele University and Durham University: Staffordshire, UK, 2007].
17. He, H.; Garcia, E.A. Learning from Imbalanced Data. IEEE Transactions on Knowledge and Data Engineering. IEEE Trans. Knowl. Data Eng., 2009; 21.

18. Kitchenham, B.A. Guidelines for Performing Systematic Literature Review in Software Engineering; Technical.
19. Naeem Seliya, T.M.; Khoshgoftaar, J.; VanHulse, J. Predicting Faults in High Assurance Software. In Proceedings of the IEEE 12th International Symposium on High Assurance Systems Engineering, San Jose, CA, USA, November 2010; 3–4: 26–34.
20. Radjenovic, D.; Hericko, M.; Torkar, R.; Zivkovic, A. Software fault prediction metrics: A Systematic literature review. Inf. Softw. Technol, 2013; 55: 1397–1418.
21. Wen, S.; Li, Z.; Lin, Y.; Hu, C.; Huang, C. Systematic literature review of machine learning based software development effort estimation models. Inf. Softw. Technol, 2012; 54: 41–59.