Software Defect Prediction Survey Introducing Innovations with Multiple Techniques

M. Prashanthi, G. Sumalatha, K. Mamatha, and K. Lavanya

Abstract The software is applied in various areas, so that the quality of the software is very important. The software defect prediction (SDP) is used to solve the issues in the software and enhance the quality. Even if SDP is very helpful in testing, predicting the defective modules is not always easy. Different problems impede smooth performance and use of model defect prediction. The prediction of software defects was an interest of investigation, as early stage prediction of defects improves software quality with reduced cost and effective managing of software. Researchers from different fields help to propose different approaches that help effectively and efficiently. A number of approaches, frameworks, methods, and modeling were proposed using different data sets, metrics, and assessment strategies, in order to remove unnecessary and erroneous details from defect-prone modules. Defects in software systems are common and may cause software users various problems. During the development of different methods, the most probable defect location in large code bases was quickly predicted. Prediction of software faults is an important and beneficial way of improving software quality and reliability. The ability to predict which components in a large software system will contain the most faults in the next release contributes to better management projects, including an early estimation of possible release delays, and a "correcting guide for improving the software's quality. The identification of bugs/defects at the early stages of the software life cycle reduces the software development effort needed. A lot of research in software fault prediction using machine learning methods has been advanced. There are mainly two problems in the prediction of software defects, dimensional reduction, and imbalances of class.

Keywords Software defect prediction · SDP techniques · SDLC · SLDeep · Support vector machine (SVM) · COSTE

M. Prashanthi (✉) · G. Sumalatha · K. Mamatha · K. Lavanya
CMR Engineering College, Hyderabad 501401, India
e-mail: prashanthi.m@cmrec.ac.in

A. Kumar et al. (eds.), *Advances in Cognitive Science and Communications*,
Cognitive Science and Technology, https://doi.org/10.1007/978-981-19-8086-2_75

1 Introduction

Nowadays, for every task, the need for software is rapidly growing [1]. The software is applied in various areas, such as traffic signal command, biopharmaceutical engineering, and banking systems, due to the progression of the network society. Hence, the quality of the software products is very important, and the important five quality aspects are effectiveness, maintainability, availability, understandability, and reliability [2]. The advanced software systems are very complex, which cause various harmful negative impacts on the robustness and reliability of the software applications [3, 4]. Therefore, the software testing process in required for every software development project, which is a costly and critical process to examine the effectiveness of the resulting product. The major points considered to test the software is the total number of staff, the time required to complete the testing, and total amount need for testing [5]. The reliability and quality of the software mainly depend on the software testing, so that it is a necessary part of the whole software development procedure [6]. Anyhow, the major problems of software testing are resource-intensive activity, required processing time, and the budget need for testing [7]. There are two ways to perform software testing, such as the linear and cyclical approaches. The waterfall model is used in the linear approach, and agile, iterative, and incremental models are used in the cyclical approach [8]. The methods used to solve the issues in software testing and enhance the quality of software are called the software defect prediction (SDP) techniques [1].

The mistakes in the software development process are called software defects, which cause collapse, failure, faults, endanger human life safety, and property [2]. SDP is used to find the errors in the software before distributing it, which ensures the software quality [9]. Also, it is the major part of software development and a very expensive and complex part [10]. Moreover, it helps in the identification of the potential bugs and guides the resources for debugging [11]. So that it is helpful for enhancing the reliability and quality of the software that leads to robust and safer software artifacts [12]. The main aim of the software quality assurance (SQA) is for controlling the software development lifecycle (SDLC) and to make sure the present model meets the expectations. SQA contains various applications, like SFP, software testing, and code walkthroughs [13–15]. There are various approaches developed in recent years, which are used in the software development process. It helps the developers to assign the finite resources testing for the defective module [16–18].

Different machine learning algorithms for software defect prediction have been shown in Fig. 1. The SDP is one of the important phases of the software testing in SDLC. It is used to identify the defects in the software, and the resources are efficiently used without violating the constraints. There are various existing methods are developed for the prediction of software defects, in which some challenges are identified.

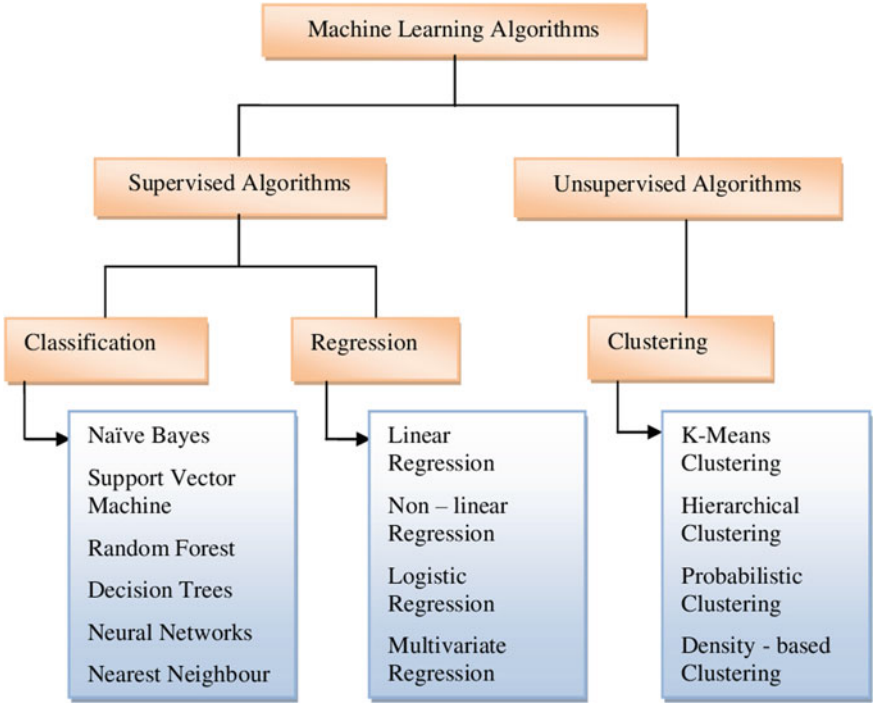

Fig. 1 Machine learning algorithms for SDP

2 Literature Review

The various existing methods used for the SDP are discussed in this section.

Majd et al. [1] developed a statement-level software defect prediction using deep-learning model (SLDeep) on static code features for predicting the software defects. This method minimized the hardships linked in pointing out the fault positions and also provided high-quality software with less effort and time. However, the effect of various nodes and layers, which are available in the long short-term memory (LSTM) network, was not exactly known.

Qiao et al. [3] devised a deep learning-based approach for predicting the defects in the software models. This method had high accuracy and attained maximum performance than the conventional methods. Anyhow, the total predictions from the change level were not known accurately.

Shi et al. [11] developed an unsupervised representation learning method for SDP, named multi-perspective tree embedding (MPT embedding). The program coding of this method was better than the conventional methods. However, this method was not used for some software, like code completion and clone detection.

Shao et al. [19] modeled a SDP model using the correlation weighted class association rule mining (CWCAR). This method was addressed the feature weights and

class imbalance of the SDP. Anyhow, this method was not applicable in real-world applications, like credit scoring and text classification.

Feng et al. [16] developed an oversampling technique named Complexity-based Over Sampling Technique (COSTE) to predict the defects in the software. This method was used to detect the diverse nature of synthetic instances, but it was not used in case of unsupervised and semi-supervised learning research.

Cai et al. [2] developed a hybrid multi-objective cuckoo search under-sampled software defect prediction model based on SVM (HMOCS-US-SVM). This method was used to remove the class imbalance (CIB) problem in data sets and used to select the parameter of SVM. However, some of the parameters of non-defective modules and SVM were not yet solved.

Ding and Xing [7] modeled a pruned histogram-based isolation forest method to predict the defects in the software. This method had a fast convergence rate and enhanced prediction performance using ensemble. The drawback of this method was the isolation features selection was carried out in a random way.

Sun et al. [20] devised a model to find the new defects data in the software, named collaborative filtering-based sampling methods recommendation algorithm (CFSR). This method was more effective and feasible in numerous cases to predict software defects. Anyhow, this method was not applicable in finding the differences between the homogeneous data, when the new data and historical data had similar software metrics.

Yucalar et al. [5] devised a Combining Predictors method, which is based on the multiple classifiers for the prediction of software defects. In this method, the performance was enhanced by reducing the effort needed for finding the software defects. However, this method was not compared with the empirical studies.

Zhao et al. [10] developed a cost-sensitive model, named Siamese parallel fully-connected neural networks (SPFCNN) for SDP. This method was used to solve the limited data and high dimensional problems, but it was not applicable for some forms of software defects.

Xu et al. [12] modeled Learning Deep Feature Representation (LDFR) for SDP. This method was useful in reducing the class imbalance problems of SDP. However, it was difficult to predict the reasons for defects in software modules.

Tumar et al. [8] devised an intelligent approach, named Binary Moth Flame Optimization (BMFO) with Adaptive synthetic sampling (ADASYN) for SDP. This method was useful in solving the class imbalance problems and finding the suitable features selection. Anyhow, the output was affected by the classifiers used in this method.

3 Different Models and Approaches

3.1 Soft Computing

Several soft computing methods have been recommended in the past for the prediction of software defects. Soft computing is a keyword of reference for aggregating different mechanisms related to computer science, such as AI methods, machine learning methods, and several other mechanisms which includes soft computing:

- "Artificial Neural Network"
- "Neural Network"
- "Support Vector Machine"
- "Swarm Intelligence: Ant Colony, Particle Swam Intelligence"
- "Machine learning techniques"
- "Probabilistic Reasoning"
- "Decision Tree"
- "*K*-Nearest Neighbors"
- "Evolutionary Computation"
- "Evolutionary Algorithm: Genetic Algorithm"
- "Fuzzy Logic"
- "Bayesian Belief Network".

Various statistical mechanisms such as the selection of feature subsets and PCA improve forecasting capacity in several SDP models. Software metrics play an important role, and models are supported in accurately predicting failures.

3.2 Software Metrics

Software metric is the software unit for measuring or specifying an attribute. These measurements are useful in determining software excellence. The quality metrics of software are a component of software metrics which focuses on product quality, methods and overall application features. Product metrics describe various characteristics of products such as volume, architectural design, computational completeness, efficiency, and quality. The process metrics of organizations are used to improve software development and support various tasks, such as product error detection, bug fixing during development, fault discovery during testing, and default removal time minimization. The project measurements define features and implementation of the projects involving software developers, recruitment patterns and price, project plan, and efficiency in the software's life cycle.

Various product metrics are

1. Chidamber and Kemerer.
2. The quality oriented extension to Chidamber and Kemerer metrics suite suggested by Tang et al.
3. Cohesion in Methods (LCOM3) suggested by Henderson-Sellers.
4. On the basis of McCabe's complexity metric the class level metrics built.
5. Martin suggested coupling metrics.
6. The Bansiya and Davis recommended QMOOD metrics suite.
7. Lines of Code (LOC).

3.3 A General Defect Prediction Process

In order to develop a prediction model, deficiency and measurement data from the software development efforts must be collected for use as a learning set. There is a compromise between how well a model matches its learning set and its performance in predicting the addition of information sets. We should therefore assess the performance of a model by comparing the predicted deficiency of the modules in a test against the actual deficiency. A general defect prediction process has been shown in Fig. 2.

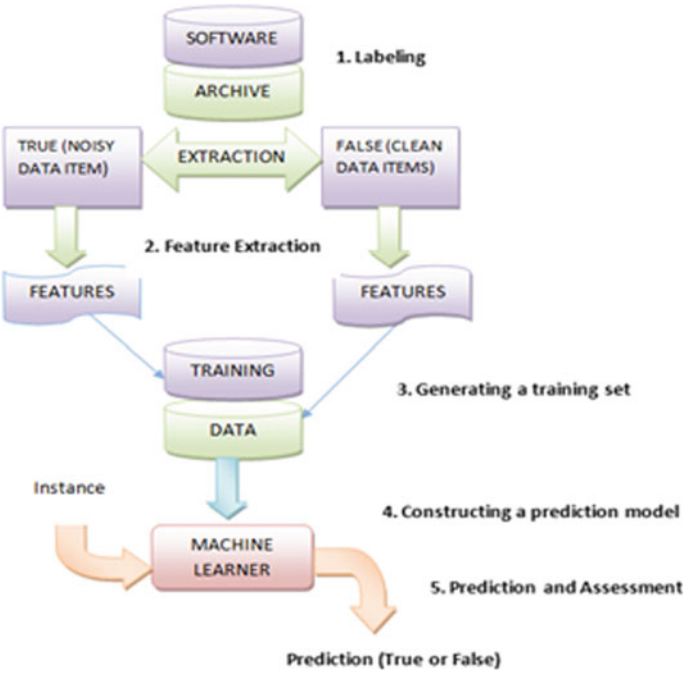

Fig. 2 General defect prediction process

3.4 Machine Learning-Based Models

Machine learning (ML) algorithms have proven to be very useful in solving a wide variety of technical problems including failure, error, and defect pulses, as the software of the system becomes more complex [21]. In cases where problem areas are not well defined, human knowledge is limited and dynamic adaptations are required to changing conditions for the development of efficient algorithms ML algorithms are very useful. Machine learning comprises various types of training, including artificial neural networks (ANNs), concept learning (CLs); Bayesian belief networks (BBNs); strengthened training (RL), genetic algorithms (GA) and genetic programming (GPs), and instance-based study (IBL) (AL).

3.5 The Fuzzy Logic Approach

The Fuzzy logic model builds on the concept or reasoning and develops an approximate value. It's a step forward from traditional Boolean logic, where True or False can exist only. The truth of any statement in the case of fuzzy logic is a degree rather than an absolute number. The greatest advantage of Fuzzy's logic, modeled upon the human intuition and behavior, is that this model does not answer traditionally, but rather gives an allotment to the human response.

The inputs of this model are placed in a series system. Then, a set of rules will be defining how inputs are used to get the output and to find the definitive value in the fuzzy set. The model has a set of metrics or RRML lists that are made of the available software metrics (Fig. 3). The measurements are relevant for their respective phases within the life cycle of software development.

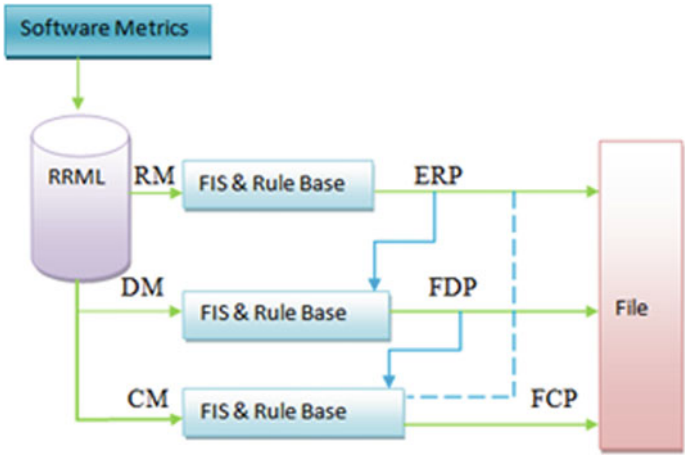

Fig. 3 Fuzzy logic approach

3.6 Code Pattern-Based Vulnerability Detection

There are two main steps in code pattern detection technology for vulnerabilities. During the workout stage, control and flow techniques are used to extract key program codes converted into a vector using the existing mainstream tool (e.g., word2vec) that can be used for supervisory training tools in suitable neural networks. During the detection phase, the same data processing is performed using a new software which identifies current vulnerabilities using the learned model. Code pattern-based vulnerability detection methods are divided into the current model training network structures of static detection methods and dynamic detection methods, including CNN, RNN, and LSTM.

3.7 Capture Recapture Analysis

This defect prediction technique is based on an assessment of patterns of defects found by independent defect detection activities in a particular software artifact. The count of latent defects is estimated by overlapping of defects identified by independent activities or groups of testers (the amount of defects remaining in a system). Default pooling is also called capture/recaptures techniques.

3.8 Expert Opinions

When experts are available, they use them for predictions based on their experience, the fastest and easiest method of defect prediction. The disadvantage of this methodology is its subjective nature and its inability to degrade correctly in lower granularity levels. This method may be useful if defect predictions at project or large component level are to be carried out, and if experts have experience of forecasting them, but if defect predictions are to be made at lower granularities (sub-systems, functions, files, etc.), this method does not scale down. This method should not be used.

3.9 Causal Model

Causal models try to establish causal relations with the expected number of defects or number of latent system failures, between software process and product attributes. Fenton and Neil criticize the application of statistical software defect prediction models for the lack of causal link modeling (BBNs). In very early stages in software projects, Bayesian Nets were used to demonstrate their application to defect forecasting.

3.10 Analogy-Based Predictions

Analog estimates are based on measurements between past and current projects collected and compared in an analogy in order to determine the most analogous project (s). Typically, size, application type, functionality complexity, and other parameters for predictions of a software defect are used to identify like projects for estimating. The analysis could be carried out at the level of the project, sub-system, or component.

3.11 Multivariate Regression

Models based on regression use statistical regression to make predictions of defects using software metrics or changing code as predictor variables. In a software project or modules (sub-systems/functions, etc.), multiple linear regression can be used to estimate the number of expected defects. As independent variables in regression-based models, a variety of software processes and product metrics have been used. Code complexity metrics and source code evolution (change) metrics are most common.

3.12 Constructive Quality Model (COQUALMO)

The model is used to build a quality model referred to as COQUALMO by expertly determined introductions and removals sub-models. The first estimated use of the sub-model Deficiency Introduction (DI) under this model was the number of non-trivial demands, design, and coding defects introduced. The DI sub-model uses the estimation of software size and other project and process attributes (platform, personal, etc.).

4 Automatic Software Program Repair

The primary purpose of early automating patching technology was to prevent worms from spreading that slowly penetrated all aspects of computer software safety through technological development.

The three phases of patching process are software failure, patch generation, and patch assessment. The location of software defects is a requirement for automatic repair and is primarily used to identify potential program defects or vulnerabilities. Common fault location technologies today are divided into two categories: static fault location technology and dynamic fault location technology. Data dependence and

data dependence relationship for program code tested is obtained by engineering for the location of the static defect generated mainly via program analysis technologies for confirming and locating the fault location. By running the default test case, a dynamic location technology receives information on the program execution and locates the default state position in the testing program by analyzing the execution flow of the program.

5 Conclusion

Early detection of software defects plays an important role in the software development cycle. In the automotive sector, development of software has largely adopted the model development paradigm that enables the easier integration of multi-provider functionality. Deficiencies are detected early and the intended functionality, robustness and compliance with model safety standards is verified and validated extensively—the quality and confidence of automotive software can be substantially improved. Effective approaches and instruments support cost reduction and reduction in development time. SDP is now dignified as a developing research zone using ML technologies. It is a challenging task to detect software failures during the first phase of SDLCS, as well as to finance high-quality software systems. The main highlight in this paper were several methods for predicting defects such as integrated approach, cross-project model, and machine learning algorithms. On the basis of the analysis, the best solution can be selected to analyze, predict, and avoid all mistakes and their limitations.

References

1. Majd A, Vahidi-Asl M, Khalilian A, Poorsarvi-Tehrani P, Haghighi H (2020) SLDeep: statement-level software defect prediction using deep-learning model on static code features. Exp Syst Appl 147:113156
2. Cai X, Niu Y, Geng S, Zhang J, Cui Z, Li J, Chen J (2020) An under-sampled software defect prediction method based on hybrid multi-objective cuckoo search. Concurr Comput Pract Exp 3(5):e5478
3. Qiao L, Li X, Umer Q, Guo P (2020) Deep learning based software defect prediction. Neurocomputing 385:100–110
4. Bowes D, Hall T, Petrić J (2018) Software defect prediction: do different classifiers find the same defects? Softw Qual J 26(2):525–552
5. Yucalar F, Ozcift A, Borandag E, Kilinc D (2020) Multiple-classifiers in software quality engineering: combining predictors to improve software fault prediction ability. Eng Sci Technol Int J 23(4):938–950
6. Lee SH, Lee SJ, Shin SM, Lee EC, Kang HG (2020) Exhaustive testing of safety-critical software for reactor protection system. Reliab Eng Syst Saf 193:106667
7. Ding Z, Xing L (2020) Improved software defect prediction using pruned histogram-based isolation forest. Reliab Eng Syst Saf 204:107170

8. Tumar I, Hassouneh Y, Turabieh H, Thaher T (2020) Enhanced binary moth flame optimization as a feature selection algorithm to predict software fault prediction. IEEE Access 8:8041–8055
9. Yu Q, Jiang S, Zhang Y (2017) A feature matching and transfer approach for cross-company defect prediction. J Syst Softw 132:366–378
10. Zhao L, Shang Z, Zhao L, Zhang T, Tang YY (2019) Software defect prediction via cost-sensitive Siamese parallel fully-connected neural networks. Neurocomputing 352:64–74
11. Shi K, Lu Y, Liu G, Wei Z, Chang J (2020) MPT-embedding: an unsupervised representation learning of code for software defect prediction. J Softw Evol Process e2330
12. Xu Z, Li S, Xu J, Liu J, Luo X, Zhang Y, Zhang T, Keung J, Tang Y (2019) LDFR: learning deep feature representation for software defect prediction. J Syst Softw 158:110402
13. Johnson AM Jr, Malek M (1988) Survey of software tools for evaluating reliability, availability, and serviceability. ACM Comput Surv (CSUR) 20(4):227–269
14. Wang H, Khoshgoftaar TM, Napolitano A (2010) A comparative study of ensemble feature selection techniques for software defect prediction. In: 2010 Ninth international conference on machine learning and applications. IEEE, pp 135–140
15. Hall T, Beecham S, Bowes D, Gray D, Counsell S (2011) A systematic literature review on fault prediction performance in software engineering. IEEE Trans Softw Eng 38(6):1276–1304
16. Feng S, Keung J, Yu X, Xiao Y, Bennin KE, Kabir MA, Zhang M (2020) COSTE: complexity-based oversampling technique to alleviate the class imbalance problem in software defect prediction. Inf Softw Technol 129:106432
17. Limsettho N, Bennin KE, Keung JW, Hata H, Matsumoto K (2018) Cross project defect prediction using class distribution estimation and oversampling. Inf Softw Technol 100:87–102
18. Nagappan N, Ball T, Zeller A (2006) Mining metrics to predict component failures. In: Proceedings of the 28th international conference on software engineering, pp 452–461
19. Shao Y, Liu B, Wang S, Li G (2020) Software defect prediction based on correlation weighted class association rule mining. Knowl Based Syst 105742
20. Sun Z, Zhang J, Sun H, Zhu X (2020) Collaborative filtering based recommendation of sampling methods for software defect prediction. Appl Soft Comput 90:106163
21. Song Q, Jia Z, Shepperd M, Ying S, Liu J (2010) A general software defect-proneness prediction framework. IEEE Trans Softw Eng 37(3):356–370