Software Fault Prediction Using Machine Learning Approaches: A Survey

[1]Ashwni Kumar, [2]Mariya Khatoon, [3]Dr.D.L.Gupta
[1]M.Tech Student, [2]M.Tech Student, [3]Associate Professor
[1]Department of Computer Science and Engineering,
[1]K.N.I.T, Sultanpur, U.P, India.

Abstract: Predicting software fault is an important part of software engineering. Fault prediction means identifying modules that are prone to failure early in software development. It reduces time, effort and overall costs. Significantly improves the organization's start-up and profits by ensuring customer satisfaction. This field has attracted many researchers over the years to improve the overall quality of the software. Machine learning techniques are the most used techniques in this field today. Machine learning focuses on developing computer programs that may learn to grow and alter once exposed to new information.

In this paper, we examine a study of the various software metrics used to predict software fault using machine learning algorithms and we also presented a survey of various machine learning techniques that will help professionals interested build a fault prediction model.

IndexTerms **– Software Fault Prediction, Machine Learning, Software Metrics, Prediction Techniques.**

I. INTRODUCTION

Today we live in the world of computers where the software is used in almost all areas of life. In 2018, the global software development market is around $ 389 billion according to IT research and consultancy firm Statista [1]. These data show the importance of software. So it is necessary for software Development Company to develop error-free software. But it's practically not possible 100% error free software. We can reduce it by using it well known techniques called fault prediction models. Software fault prediction models are used to identify modules at an early stage of software development because detecting a fault at a later stage will increase the cost exceptionally high. So the quality will also decrease as it leads to customer dissatisfaction. So Software Fault Prediction models help the test team will focus more on modules that are prone to failures and optimize the use of resources [2].

Machine learning techniques play an important role in predicting software failures. Various researchers have demonstrated the importance of machine learning in software fault prediction and there is empirical evidence that the prediction model's performance is strongly influenced by the type of technique used. It is therefore essential to select the appropriate technique for the indicated dataset [3]. Therefore, this article presents various machine learning techniques that have been used in the field of predicting software defects by various researchers over the years. Changes in existing Machine Learning techniques that make them more effective on a daily basis and which attract many researchers in this field. A future dimension is also proposed to develop hybrid techniques for predicting software failures in order to improve overall software quality. Our next section discusses about software metrics and possibilities to use software metrics for software fault prediction, Section 3 discusses some selected machine learning techniques; Section 4 presents the related work carried out over the years by various researchers in chronological order, and finally section 5 presents the conclusions and future work.

II. SOFTWARE METRIC

The software metric is a measure of some software properties or its specifications. Software metrics are often used to evaluate software's ability to achieve a predefined goal. Software metric is a measure of some software properties [4]. Complexity, coupling and cohesion (CCC) metrics can be measured during software development stages, such as design or coding, and are used to evaluate software quality. Fault prediction metrics play the most important role in building a statistical forecasting model. Most fault prediction measures can be classified into two types: code measures and process measures. With code metrics such as dimensions, Hastead, McCabe, CK and OO metrics used, the absolute frequency of use of code metrics is higher than process metrics [5]. In the following some code metrics are as follows:

2.1 Cyclomatic Complexity

Measure the structural complexity of the code. It is created by calculating the number of different code paths in the program flow. A program with a complex control flow will require more testing to achieve good code coverage and will be less manageable [4].

2.2 Halsteads Product Metrics

The measures were developed by the late Maurice Halstead in order to determine a quantitative measure of complexity directly from the operators and operands of the module, as well as from the program vocabulary, from the length of the program [5].

2.3 Product Metrics

In Product Metrics contains lines of code (LOC) indicates the approximate number of lines in the code. The count is based on the code and is therefore not the exact number of lines in the source code. A very large number may indicate that a type or method is trying to do too much work and should be divided. Design measures calculated based on the requirements or the design document before the implementation of the system. Object-oriented metrics help identify failures and allow developers to see firsthand how to simplify their classes and objects [5].

III. MACHINE LEARNING APPROACHES FOR FAULT PREDICTION

Software fault prediction is the process of predicting which parts of software are subject to fault prone. By focusing on the fault prone files, testers can save on testing efforts. As mentioned above, it is important to detect the software defect / fault as early as possible to reduce testing costs. Therefore, the use of software metrics produced at an early stage in the software development process is a superiority of this research. The idea behind predicting software fault is to use measurements taken from the development process, e.g. source code that can be found in the software metrics. Many approaches have been designed to perform software fault prediction, starting with a simple equation, statistical analysis, expert estimation and machine learning. Of all the approaches, machine learning has proven to be the best research-based approach [6]. To automatically predict the fault prediction module, a wide range of machine learning algorithms are used. The machine learning algorithm is known as classification. The classification procedure basically means class labeling for a new sample based on a certain set of samples that have been labeled. Different machine learning algorithms will be discussed in this subsection.

3.1 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a machine learning technique to correctly classify invisible data by building an N-dimensional hyperplane. Finding the optimal hyperplane that separates the cluster from the vector is the main activity of modeling the SVM. Therefore, observations with a dependent variable category will separate side by side with observations with another category of the aircraft. The term support vector refers to all vectors close to the hyperplane. SVM modeling finds a hyperplane oriented to maximize the margin between support vectors. SVM can handle the nonlinear separator between points using the kernel function to map data to a different space, so a hyperplane will be used to separate the space [7].

3.2 Random Forest

Random forest is another type of machine learning consisting of many classification trees. Classification trees are decision trees that represent an important step forward in knowledge discovery and data mining. The random forest classifier provides a prediction with great precision. The classification algorithm classifies a new object from an input vector. The input vector is written on each tree in the forest, where each tree provides a classification result. The tree votes for this class, so the forest will choose most of the grading votes [8].

3.3 Bootstrap Aggregating

The next machine learning method is the bagging algorithm or known as bootstrap aggregation. Bootstrap aggregation is a technique that will repeatedly sample a series of data based on a uniform probability distribution. The size of each bootstrap sample and the original data are identical. An instance may occur multiple times during the same training set because sampling is done with replacement. On the other hand, another instance could be omitted from the training package. Each sample has a 1- (1-1 / N) N probability of being selected. Therefore, a bootstrap sample can contain approximately 63% of the original training data in sufficiently large data of N. An instance will be assigned to a class after training the K classifiers if the class receives the most votes.

3.4 Artificial Neural Network (ANN)

The Artificial Neural Network (ANN) is recently popular, such as multilayer Perceptron (MLP). MLP could be used to solve almost any problem, such as pattern recognition, interpolation etc. Direct acting neural networks are trained with a reverse propagation algorithm which consists of two steps (forward and reverse pass). The pass forward presented a voice to the neural network which will propagate through the network.

3.5 Naive Bayesian(Naïve Bayes)

Naive Bayesian is a fairly effective classification method that is easy to implement. The Bayesian classification consider as supervised learning strategy and statistical techniques for classification. The basic algorithm is a probability according to Bayes' theorem, which assumes that the existence of class characteristics does not depend on another existence of characteristics [9]. Naïve Bayes techniques basically work in two stages: Training stage and Prediction stage.

IV. LITERATURE REVIEW

This section presents some latest ongoing research performed in software fault prediction using machine learning techniques.

4.1 Classification Approach

Ezgi Erturk et al. [10] proposed a new Adaptive Neuron Fuzzy Inference System (ANFIS) method for predicting software fault. Data is collected from the PROMISE software engineering repository and McCabe's measurements are selected because they fully respond to the programming effort. The results obtained were 0.7795, 0.8685 and 0.8573 respectively for the SVM, ANN and ANFIS methods.

David Gray et al. [11] in this paper, the emphasis is on classification analysis rather than classification performance, it was decided to classify training data rather than having an interesting set of testers. It involves manual analysis of predictions made by classifiers of support vector machines using data from the NASA Metrics Data Program repository. The goal was to better understand how classifiers separated training data.

Surndha Naidu [12] in this paper they focused on finding the total number of fault in order to reduce time and costs. Here, for the classification of faults, they have use the ID3 algorithm (Iterative Dichotomiser 3). ID3 is an algorithm invented by Ross Quinlan used to generate a decision tree from a set of data. Faults were classified according to five attribute values such as volume, program duration, difficulty, effort and time estimate.

Mie Thet Thwin [13] in this paper, two types of neural network techniques are performed. The first focuses on predicting the number of faults in a class and the second focuses on predicting the number of changed rows per class. Two neural network models are used: the Ward neural network and the general regression neural network (GRNN) and perform the analysis on the NASA dataset.

4.2 Clustering Approach

Xi Tan et al. [14] in this paper, a new method to predict software fault based on functional clusters of programs to improve performance. Most of the methods proposed in this sense foresee errors per class or file. Experiments concluded that cluster-based models can significantly improve recall from 31.6% to 99.2% and accuracy from 73.8% to 91.6%.Open in Google Translate Feedback.

Jaspreet Kaur et al. [15] in the paper, k-means clustering approach was used to find the fault proneness of object-oriented systems and found that k-means clustering techniques show an accuracy of 62.4%. It also showed a high and low detection probability value.

4.3 Association Rule Mining Approach

Alina Campan et al. [16] proposed a new algorithm for discovering ordinal association rules of any length of interest in datasets. Data sets that contain multiple attributes with domains of similar or comparable values are common in data mining.

Gabriela Czibula et al. [17] proposed a new supervised method for detecting software entities with architectural flaws, based on the exploration of relational association rules, called SDDRAR (detection of software design errors using relational association rules). Experiments are performed on open source software to detect defective classes in object-oriented software systems, for example the FTP4J project is a Java implementation of a complete FTP client. Each of these four versions has 27 classes (and 8 interfaces excluded from the analysis) and the class names are the same for each version.

D. Rodrguez et al. [18] have proposed EDER-SD (Evolutionary Decision Rules for Subgroup Discovery), an algorithm based on an evolutionary calculation that induces rules that describe only fault-prone modules. EDER-SD has the advantage of working with continuous variables because the conditions of the rules are defined using ranges.

4.4 Hybrid Approach

Kamei et al. [19] proposed a method of predicting the failure-prone module that combines exploration of association rules and analysis of logistic regression. Predictive performance of the proposed method with different thresholds of each measure of interest of the rule (support, confidence and elevation) using a module defined in the Eclipse project.

Martin Shepperd et al. [20] studied on publicly available NASA data sets have been widely used in this research to classify software modules into defect and non-defect categories. In this regard, Promise Data Repository 2 has played an important role in making software engineering data sets publicly available. For example, 96 software error data sets are available. Among them are 13 of the 14 datasets provided by NASA and which were also available for download from the NASA Data Program (CDM) website.

In the table 4.1 it gives the overview of the literature on the use of various Machine Learning methods for software fault prediction. This table summarizes the applied modeling techniques, the datasets used and the empirical configuration of many studies. As shown in table 4.1, a huge number of approaches have been applied to find the best approach to predict software fault in term of accuracy and comprehensibility.

Table 4.1: Literature Overview of the Application of Machine Learning Approaches for Software Fault Prediction

Authors	*Dataset*	*Metrics*	*Approach*	*Result*
Catal et al. [21]	5 public NASA datasets	13 method level metrics	Random Forest Naive Bayes J48 Immunos 1 & 2 CLONALG AIRS1 & 2 AIRS 2 Parallel	Naïve Bayes algorithm is better for smaller dataset, while Random forest algorithm perform best for larger dataset
Catal et al. [22]	6 Object Oriented metrics of Chidamber Kemerer metrics, 4 metrics from KC1 projects, and Halstead & McCabe metric	WMC (Weighted Methods per Class) DIT (Depth of Inheritance Tree) RFC (Response for a Class) NOC (Number of Children) CBO (Coupling Between Object Classes) LCOM (Lack of Cohesion in Methods) Percent Pub Data Access To Pub Data Dep On Child Fan In	AIRS - Artificial Immune Recognition System algorithm	The best fault prediction result is combination between CK metrics and the lines of code (LOC) metric
Shanthini [23]	Public domain KC1 NASA data set, method level and class level metrics	21 method-level metrics proposed by Halstead and McCabe. 10 class-level oriented metrics are used	Naïve Bayes SVM K-Star Random Forest	Precision, recall and accuracy of SVM show better result compare to other machine learning methods for both class and method level metrics.
Mundada et al. [24]	PROMISE repository: JM1/Software Defect Prediction	Object Oriented CK metrics	Artificial Neural Network and Resilient Back Propagation	ANN show better accuracy compare to previous research
Bishnu et al. [25]	AR3, AR4, AR5 from Promise data Iris dataset	Lines of Code (LoC) metric, Unique Operator (UOp), Cyclomatic Complexity (CC), Total Operator (TOp), Unique Operand (UOpnd), Total Operand (TOpnd)	Quad Tree-Based K-Means Clustering Algorithm	K-Means algorithm more efficient in the term of iterations except for AR5, and the SSE.

Dejaeger et al. [26]	An open source Eclipse Foundation & NASA IV&V facility dataset	Halstead, Line Of Code, and McCabe complexity metrics	Fifteen Bayesian Network (BN) classifiers	Compared to common Naïve Bayes classifier, Augmented Naive Bayes classifiers show better performance.
Okutan et al. [27]	Promise data repository	9 datasets from Promise data repository, NOD for the number of developers and LOCQ for the source code quality.	Bayesian networks	LOC, LOCQ, and RFC are the most effective metrics.
Kumar et al. [28]	45 real-life datasets from the PROMISE repository	Chidamber and Kemerer Java Metrics	Majority Voting Ensemble (MVE) method	MVE approach performs the best result. Fault prediction model developed using MVE method consume less fault removal cost as compare to other techniques
Alighardashi et al. [29]	NASA and PROMISE with 10 datasets	20 object oriented metrics	Feature selection method by using combination of filter feature selection methods	The results show the effectiveness of the used method. Therefore, our proposed WF method can find the best features with the highest speed for the improvement of the fault prediction accuracy. In this research, we use the five filter methods

V. CONCLUSION AND FUTURE SCOPE

In this paper, we provide a detailed survey of various Machine Learning (ML) techniques for software fault prediction. Software fault prediction is needed to minimize software testing costs and times. Multiple error-prone modules require more resources. According to the survey, we can see that the software flaw is indeed a big problem in software engineering. Predicting software fault modules using different ML techniques aims to improve the quality of the software development process. The purpose of this survey is to access the research work conducted by various researchers on ML techniques to predict software fault in order to help interested professionals build a model to predict fault. After a detailed examination, we have found that the random forest, naive Bayes, and the neural network are good enough for software fault prediction, but no single technique is appropriate for all types of data. It is therefore preferable to choose the result in the forecast model set. Therefore, in the future, we are planning to implement a heterogeneous ensembling to increase the overall efficiency of the system.

REFERENCES

[1] M. Jrgensen, K. Molkkenstvold, "how large are software cost overruns? A review of the 1994 CHAOS report," Information and Software Technology 48 (4) (2006) 297{301}.

[2] Rajkumar G and K.Alagarsamy, "The Most Common Factors For The Failure Of Software Development Project," vol. 11, pp. 74-77, January 2013.

[3]Rathore, S. S., & Kumar, S, "A study on software fault prediction techniques Artificial Intelligence Review," 1–73. https://doi.org/10.1007/s10462-017-9563-5.

[4] E. E. Mills, Software metrics," 2000.

[5] A. Campan, G. Serban, T. M. Truta, and A. Marcus, "An algorithm for the discovery of arbitrary length ordinal association rules," DMIN, vol. 6, pp. 107-113, 2006.

[6] Catal, C., & Diri, B. (2009), "A systematic review of software fault prediction studies. Expert systems with applications," 36(4), 7346-7354.

[7] Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. Numerical Recipes: The Art of Scientific Computing, Section 16.5, Support Vector Machines, Cambridge University Press, The 3rd Edition, 2007.

[8] Leo Breiman. "RANDOM FORESTS. Machine Learning," pp. 5-32.2001.

[9] John, G. H., & Langley, P. "Estimating continuous distributions in Bayesian classifiers," In the 11th Conference on Uncertainty in artificial intelligence, pp. 338- 345,1995.

[10]E. Erturk and E. A. Sezer, "A comparison of some soft computing methods for software fault prediction," Expert Systems with Applications, 2014.

[11]D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson,"Software defect prediction using static code metrics underestimates defect-proneness," in Neural Networks (IJCNN), The 2010 International Joint Conference on, pp. 1-7, IEEE, 2010.

[12]Naidu, M. Surendra, and N. GEETHANJALI,"Classification of Defects in Software using Decision Tree Algorithm." International Journal of Engineering Science and Technology (IJEST) 5.06 (2013).

[13]M. M. T. Thwin and T.-S. Quah, Application of neural networks for software quality prediction using object-oriented metrics," Journal systems and software, vol. 76, no. 2, pp. 147-156, 2005

[14]Xi Tan, Xin Peng, Sen Pan,Wenyun Zhao, "Assessing software quality by program Clustering and Defect Prediction" 18th Working Conference on Reverse Engineering 2011.

[15]Jaspreet Kaur, Parvinder S. Sandhu, "A k-means Based Approach for Prediction of Level of Severity of Faults in Software systems", Proceedings of international Conference on Intelligent Computational Systems, 2011.

[16]A. Campan, G. Serban, T. M. Truta, and A. Marcus, "An algorithm for the discovery of arbitrary length ordinal association rules," DMIN, vol. 6, pp. 107-113, 2006.
[17]G. Czibula, Z. Marian, and I. G. Czibula, "Detecting software design defects using relational association rule mining," Knowledge and Information Systems, pp. 1-33, 2012.
[18]D. Radjenovi_c, M. Heri_cko, R. Torkar, and A. Zivkovi_c, "Software fault prediction metrics: A systematic literature review," Information and Software Technology, vol. 55, no. 8, pp. 1397-1418 ,2013.
[19]Y. Kamei, A. Monden, S. Morisaki, and K.-i. Matsumoto, "A hybrid faulty module prediction using association rule mining and logistic regression analysis," in Proceedings of the Second ACM-IEE international symposium on Empirical software engineering and measurement, pp. 279-281, ACM, 2008.
[20]M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data quality: some comments on the nasa software defect datasets," Software Engineering, IEEE Transactions on, vol. 39, no. 9, pp. 1208 -1215, 2013.
[21]Catal, C., & Diri, B. (2009). "Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem," Information Sciences, 179(8), 1040-1058.
[22]Catal, C., Diri, B., & Ozumut, B. (2007, June). "An artificial immune system approach for fault prediction in object oriented software.,"In Dependability of Computer Systems, 2007. DepCoS-RELCOMEX'07. 2nd International Conference on (pp. 238-245). IEEE.
[23] A. Shanthini,, "Applying Machine Learning for Fault Prediction Using Software Metrics," vol. 2, no. 6, pp. 274– 278, 2012.
[24]D. Mundada, A. Murade, O. Vaidya, and J. N. Swathi, "Software Fault Prediction Using Artificial Neural Network And Resilient Back Propagation," *Int. J. Comput. Sci. Eng.*, vol. 5, no. 03, pp. 173–179, 2016.
[25]P. S. Bishnu. and V. Bhattacherjee, "Software Fault Prediction Using Quad Tree-Based K-Means Clustering Algorithm," vol. 24, no. 6, pp. 1146–1150, 2012.
[26]K. Dejaeger, T. Verbraken, and B. Baesens, "Prediction Models Using Bayesian Network Classifiers," vol. 39, no. 2, pp. 237–257, 2013.
[27]A. Okutan and O. Taner, "Software defect prediction using Bayesian networks," no. 2, pp. 154–181, 2014.
[28]Kumar, L., Rath, S., & Sureka, A. (2017),"Using Source Code Metrics and Ensemble Methods for Fault Proneness Prediction". arXiv preprint arXiv:1704.04383.
[29]F. Alighardashi, M. Ali, and Z. Chahooki, "The Effectiveness of the Fused Weighted Filter Feature Selection Method to Improve Software Fault Prediction," no. 8, 2016.