International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Survey on Software Defect Prediction Using Machine Learning Techniques

Pooja Paramshetti[1], D. A. Phalke[2]

[1]Student, Department of Computer Engineering, D. Y. Patil college of Engineering, Akurdi, Pune 411 044, Savitribai Phule Pune University, India

[2]Prof, Department of Computer Engineering, D. Y. Patil college of Engineering, Akurdi, Pune 411 044, Savitribai Phule Pune University, India

Abstract: ***Software defect prediction plays an important role in improving software quality and it help to reducing time and cost for software testing. Machine learning focuses on the development of computer programs that can teach themselves to grow and change when exposed to new data. The ability of a machine to improve its performance based on previous results. Machine learning improves efficiency of human learning, discover new things or structure that is unknown to humans and find important information in a document. For that purpose, different machine learning techniques are used to remove the unnecessary, erroneous data from the dataset. Software defect prediction is seen as a highly important ability when planning a software project and much greater effort is needed to solve this complex problem using a software metrics and defect dataset. Metrics are the relationship between the numerical value and it applied on the software therefore it is used for predicting defect. The primary goal of this survey paper is to understand the existing techniques for predicting software defect.***

Keywords: Software defect prediction, Software Metrics, Machine learning techniques.

1. Introduction

Nowadays, software systems have become increasingly complex and versatile. It is very important to continuously identify and correct software design defects. Thus, accurately predicting whether a software entity contains design defects can help improve the quality of the software systems. Software metric is a measure of some property of a piece of software. It measured during the software development phases such as design or coding and used to evaluate the quality of software. A Software defect is a condition in a software product which does not meet a software requirement or end user. In other words, a defect is an error in coding or logic that causes a program to malfunction or to produce incorrect unexpected results. Software defect prediction is the process of locating defective modules in software. For producing high quality software, the delivering final product should have as few defects as possible. For early detection of software defects could lead to reduced development costs and rework effort and more reliable software. Therefore the defect prediction is important to achieve software quality. Defect prediction metrics play the most important role to build a statistical prediction model. Most defect prediction metrics can be categorized into two kinds: code metrics and process metrics. The prediction models can then be used by the software organizations during the early phases of software development to identify defect modules. The software organizations can use this subset of metrics amongst the available large set of software metrics. These metrics can be used in developing the defect prediction models. Many researchers have used various methods to establish the relationship between the static code metrics and defect prediction. These methods include the traditional statistical methods such as logistic regression and the machine learning methods such as Decision trees, Naive Bayes, Support Vector Machines, Artificial Neural Networks [7]. The artificial neural network searches for a set of weights that can model the data so as to minimize the mean squared distance between the network's class prediction and the actual class label of data tuples. Support Vector Machine transforms the original data in a higher dimension, from where it can find a hyper plane for separation of the data using essential training tuples. Decision tree performs the operation in tree structure form. It also uses an attribute selection measure to select the attributes tested for each non leaf node in the tree.

The rest of the paper is organized as follows. Section II describes about the software metrics. Section III describes the available literature on approaches for discrimination prevention in data mining. Section IV has the comparative analysis of these techniques. Section V gives the conclusion for the topic.

2. Software Metric

A software metric is a measure of some property of a piece of software or its specifications. Software metrics are often used to assess the ability of software to achieve a predefined goal. A software metric is a measure of some property of a piece of software [19]. Complexity, coupling, and cohesion (CCC) related metrics can be measured during the software development phases such as design or coding and used to evaluate the quality of software. Defect prediction metrics play the most important role to build a statistical prediction model. Most defect prediction metrics can be categorized into two kinds: code metrics and process metrics. Code metrics such as size, Hastead, McCabe, CK and OO metrics have used absolute use frequency of code metrics is higher than process metrics[13]. In the following some code metrics are as follows:

2.1 Cyclomatic Complexity

It measures the structural complexity of the code. It is created by calculating the number of different code paths in the flow of the program. A program that has complex control flow will require more tests to achieve good code coverage and will be less maintainable [19].

2.2 Halsteads Product Metrics

The measures were developed by the late Maurice Halstead as a means of determining a quantitative measure of complexity directly from the operators and operands in the module and also program vocabulary, program length [13].

2.3 Product Metrics

In Product Metrics contains the 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 high count might indicate that a type or method is trying to do too much work and should be split up. Design metrics computed from requirements or design document before the system has been implemented. Object oriented metrics help identify faults, and allow developers to see directly how to make their classes and objects simpler [19].

3. Literature Review

Software fault prediction is the most popular research area in these predicting defects using software metrics and machine learning techniques recently several research centers started new projects. In this investigated more software defect prediction papers published scenes year 1990 to 2014. In this paper we categorized according to the machine learning techniques.

3.1 Software Defect Prediction Based on Classification Techniques

Ezgi Erturk et al. [11] proposed a new method Adaptive Neuron Fuzzy Inference System (ANFIS) for the software fault prediction. Data are collected from the PROMISE Software Engineering Repository, and McCabe metrics are selected because they comprehensively address the programming effort. The results achieved were 0.7795, 0.8685, and 0.8573 for the SVM, ANN and ANFIS methods, respectively.

Mie Thet Thwin [18] in this paper, two kinds of neural network techniques are performed. The first focuses on predicting the number of defects in a class and the second on predicting the number of lines changed per class. Two neural network models are used those are Ward neural network and General Regression neural network (GRNN) and perform the analysis result on the NASA dataset.

David Gray et al. [16] in this paper the main focus is on classification analysis rather than classification performance, it was decided to classify the training data rather than having some form of tester set. It involves a manual analysis of the predictions made by Support vector machine classifiers using data from the NASA Metrics Data Program repository. The purpose of this was to gain insight into how the classifiers were separating the training data.

Surndha Naidu[17] in this paper focused on finding the total number of defects in order to reduce the time and cost. Here for defect classification used ID3(Iterative Dichotomiser 3) algorithm. ID3 is an algorithm invented by Ross Quinlan used to generate a decision tree from a dataset. The defects were classified based on the five attribute values such as Volume, Program length, Difficulty, Effort and Time Estimator.

Cagatay Catal[1] in this paper they studied various paper in year 1990 to 2009 those are as following: In year 1990 to 1998 they used classification trees with method level metrics on two software systems of NASA and Hughes Aircraft and also applied logistic regression, classification trees, optimized set reduction (OSR) methods on 146 components of an ADA system which consists of 260,000 lines of code. The best performance was achieved with OSR technique in this study and correctness and completeness was 90%. They applied artificial neural networks and discriminate model on a very large telecommunications system which consists of 13 million lines of code. Evett, Khoshgoftar, Chien, and Allen (1998) predicted quality based on genetic programming on a military communication system and telecommunications system. Performance was quite well on these datasets. In year 2000 to 2004 applied fuzzy subtractive clustering method to predict the number of faults and then, they used module-order modeling to classify the modules into faulty or non-faulty classes. They stated that process metrics do not improve the classification accuracy and such a model does not provide acceptable results. They used principal component analysis for feature selection and then applied fuzzy nonlinear regression (FNR) to predict faults on a large telecommunications system developed with Protel language. They reported that fuzzy nonlinear regression method is an encouraging technology for early fault prediction. In this used fuzzy clustering and then, they applied radial basis function (RBF) to predict software faults. It predicted software quality by using Support Vector Machines (SVM) and 11 method level metrics. Type-I error and Type-II error were used to evaluate the performance of model. They reported that SVM performed better than quadratic discriminate analysis and classification tree. They investigated several data mining algorithms for software fault prediction on public NASA datasets and they used method level metrics. In year 2009 they focused on the high-performance fault predictors based on machine learning such as Random Forests and algorithms based on a new computational intelligence approach called Artificial Immune Systems. Public NASA datasets were used. They reported that Random Forests provides the best prediction performance for large datasets and Naive Bayes is the best result prediction algorithm for small datasets in terms of the Area under Receiver Operating Characteristics Curve (AUC) evaluation parameter.

Ruchika Malhotra[7] in this paper they analyses and compares the statistical and six machine learning methods for fault prediction. These methods (Decision Tree, Artificial Neural Network, Cascade Correlation Network, Support Vector Machine, Group Method of Data Handling Method, and Gene Expression Programming) are empirically validated to find the relationship between the static code metrics and the fault proneness of a module. In this they compare the models predicted using the regression and the machine learning methods used two publicly available data sets AR1 and AR6.

Martin Shepperd et al.[12] presented a novel benchmark framework for software defect prediction. In the framework involves both evaluation and prediction. In the evaluation stage, different learning schemes are evaluated according to that scheme selected. Then, in the prediction stage, the best learning scheme is used to build a predictor with all historical data and the predictor is finally used to predict defect on the new data.

Ahmet Okutan [14] in this paper use Bayesian networks to determine the probabilistic influential relationships among software metrics and defect proneness. The metrics used in Promise data repository, define two more metrics, i.e. Number of Developers (NOD) for the number of developers and Lack of Coding Quality (LOCQ) for the source code quality.

3.2 Software Defect Prediction Based on Clustering Techniques

Xi Tan et al.[5] in the paper, a novel software defect prediction method based on functional clusters of programs to improve the performance. Most methods proposed in this direction predict defects by class or file. Experiments carried out concluded that cluster based models can significantly improve the recall from 31.6 % to 99.2% and precision from 73.8 % to 91.6%.

Jaspreet Kaur et al.[4] in the paper, k-means based clustering approach has been used for finding the fault proneness of the Object oriented systems and found that k-means based clustering techniques shows 62.4% accuracy. It also showed high and low value of probability of detection.

3.3 Software Defect Prediction Based on Association rule mining

Alina Campan et al.[13] proposed the a novel algorithm for the discovery of interesting any length ordinal association rules in data sets. Datasets that contain several attributes with similar or comparable domains of values are frequent in data mining.

Gabriela Czibula et al.[3] they proposed a novel supervised method for detecting software entities with architectural defects, based on relational association rule mining, called SDDRAR (Software Design Defect detection using Relational Association Rules). Experiments on open source software are conducted in order to detect defective classes in object oriented software systems for example the FTP4J project is a Java implementation of a full-featured FTP client. Each of these four versions has 27 classes (and 8 interfaces which are excluded from the analysis), and the class names are the same for every version, too.

D. Rodrguez et al.[9] they have propose EDER-SD (Evolutionary Decision Rules for Subgroup Discovery), a this algorithm based on evolutionary computation that induces rules describing only fault-prone modules. EDER-SD has the advantage of working with continuous variables as the conditions of the rules are defined using intervals.

3.4 Software Defect Prediction Based on Hybrid Approach

Kamei et al.[2] proposed a fault-prone module prediction method that combines association rule mining with logistic regression analysis. The prediction performance of the proposed method with different thresholds of each rule interestingness measure (support, confidence and lift) using a module set in the Eclipse project.

Martin Shepperd et al.[10] studied on the publicly available NASA datasets have been extensively used as part of this research to classify software modules into defect prone and not defect-prone categories. In this regard, the Promise Data Repository 2 has served an important role in making software engineering data sets publicly available. For example, there are 96 software defect datasets available. Amongst these are 13 out of the 14 data sets that have been provided by NASA and which were also available for download from the NASA Metrics Data Program (MDP) website.

Yuan Jiang,Ming Li et al.[9] in this paper they address two practical issues first, it is rather difficult to collect a large amount of labeled training data for learning a well-performing model and second, in a software system there are usually much less defective modules than defect-free modules, therefore learning techniques would have to be conducted over an imbalanced data set therefore they proposing a novel semi-supervised learning approach named Random Committee with Under Sampling(Rocus). This method incorporates recent advances in disagreement-based semi-supervised learning with under-sampling strategy for imbalanced data.

4. Comparative Analysis

This section illustrates comparative analysis of various techniques used for software defect prediction along with their advantages and limitations. The comparative study is shown in the following table1.

Table 1: Comparative Analysis

Techniques	*Data set used*	*Advantages*	*Limitations*
Artificial Neural network	NASA AR1,AR6 And MDP	No need to know metrics relationships. It has self learning capability therefore get more accuracy	It can not manage imprecise information
Support Vector Machine	NASAAR1 , AR6	Using different kernel function it gives better prediction result	Not suitable for large number of software metrics
Decision Tree	NASA AR1,AR6	Performing operation on tree structure therefore more accurate result compare to others	Construction of decision tree is complex
Association Rule	NASA MDP repository	Generated rules using historical data and predict defect	Require Continues value of software metrics
Clustering	NASA MDP repository	It suitable for small dataset	Dataset should be unlabeled

5. Conclusion

This paper presents a survey of various machine learning techniques for software defect predication. From the survey, it can be observed that software defect is indeed a major issue in software engineering. Software defect module prediction using different machine learning techniques is to improve the quality of software development process. By using this technique, software manager effectively allocate resources. For predicting defects we analyzed the advantages and limitation of Artificial neural network, Support vector machine, Decision tree, Association rule and Clustering machine learning techniques.

6. Acknowledgment

The authors would like to thank the publishers and researchers for making their resources available. We also thank the college authority for providing the required infrastructure and support. Finally we would like to extend our heartfelt gratitude to friends and family members.

References

[1] A. C. Catal, Software fault prediction: “A literature review and current trends," Expert systems with applications, vol. 38, no. 4, pp. 4626-4636, 2011.
[2] 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.
[3] 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.
[4] 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.
[5] Xi Tan, Xin Peng, Sen Pan,Wenyun Zhao, “Assessing software quality by program Clustering and Defect Prediction” 18th Working Conference on Reverse Engineering 2011.
[6] Tao WANG, Weihua LI, Haobin SHI, Zun LIU, “Software Defect Prediction on Classifier Ensemble”, Journal of Information & Computational Sciences, 8:16(2011) 4241-4254.
[7] R. Malhotra, “Comparative analysis of statistical and machine learning methods for predicting faulty modules,”Applied Soft Computing, vol. 21pp. 286-297 2014.
[8] M. M. T. Thwin and T.-S. Quah, “Application of neural networks for software quality prediction using object-oriented metrics,” Journal of systems and software, vol. 76, no. 2, pp. 147-156, 2005.
[9] 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.
[10] 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.
[11] E. Erturk and E. A. Sezer, “A comparison of some soft computing methods for software fault prediction,” Expert Systems with Applications, 2014.
[12] M. Shepperd, D. Bowes, and T. Hall, “Researcher bias: The use of machine learning in software defect prediction,” Software Engineering, IEEE Transactions on, vol. 40, no. 6, pp. 603-616, 2014.
[13] 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.
[14] Okutan, Ahmet, and Olcay Taner Yıldız. "Software defect prediction using Bayesian networks." *Empirical Software Engineering* 19.1 (2014) 154-181
[15] J. Nam, Survey on software defect prediction," 2010.
[16] 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.
[17] 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).
[18] 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.
[19] E. E. Mills, Software metrics," 2000.