Research on Cross-Project Software Defect Prediction Based on Machine Learning

Baoping Wang[1,3], Wennan Wang[1], Linkai Zhu[1,2], and Wenjian Liu[1](✉)

[1] Faculty of Data Science, City University of Macau, Macau, China
wwwennan@zcst.edu.cn, andylau@cityu.mo
[2] Institute of Software, Chinese Academy of Sciences, Beijing, China
linkai@iscas.ac.cn
[3] Beijing Huayuexun Technology Co., Ltd., Beijing, China
D19092105076@cityu.mo

Abstract. In recent years, machine learning technology has developed vigorously. The research on software defect prediction in the field of software engineering is increasingly adopting various algorithms of machine learning. This article has carried out a systematic literature review on the field of defect prediction. First, this article studies the development process of defect prediction, from correlation to prediction model. then this article studies the development process of cross-project defect prediction based on machine learning algorithms (naive Bayes, decision tree, random forest, neural network, etc.). Finally, this paper looks forward to the research difficulties and future directions of software defect prediction, such as imbalance in classification, cost of data labeling, and cross-project data distribution.

Keywords: Machine learning · Software defect prediction model · Metric

1 Introduction

With the breakthrough development of various technologies and the increasing software requirement, the functions of software products are becoming more and more powerful, but the system is also becoming more and more complex. Due to the complexity of the software itself, technical architecture, team capabilities, and use environment, there are various defects in software products. Software defects can cause serious accidents and even great economic losses and casualties. 2018 In October and March 2019, two Boeing 737MAX planes belonging to Lion Air Singapore and Ethiopian Airlines crashed one after another, causing a total of 346 people to die. The 737MAX series were subsequently grounded globally, and Boeing's reputation was also severely damaged. The follow-up air crash report pointed out that the accident was caused by Boeing engineers' wrong technical assumptions, the lack of transparency in Boeing's management, and the serious inadequacy of FAA supervision; in November 2020, a software failure in Michigan, the United States, erroneously 6,000 votes Vote for Biden, as many as 47 counties have been affected, causing the US general election process to be questioned.

The defect prediction model is established by software measurement data and historical labeling data. Through the collected software historical data, Use various algorithms of machine learning to obtain a predictive model, and predict subsequent software defects on this basis, so as to reduce the cost of defect repair and improve software quality. ensuring the availability and reliability of software, have important scientific value and social significance.

2 Software Defect Prediction Research

Since the 1970s, research on software defects has begun, and software defects have been studied more through correlation and causality. Since 2000, with the popularity of configuration management tools, the software development process and software measurement data have become more and more abundant. At the same time, the performance of computer hardware has continued to improve, and machine learning algorithms have been more applied to software defect prediction research. Software defect Prediction methodologies are shown in the table below (Table 1).

Table 1. Software defect Prediction methodology

Number	Category	Methodology
[S1]	Metrics	The relationship between software defects and code lines (Akiyama 1971)
[S2]	Control flow	Cyclomatic complexity (McCab 1976)
[S3]	Error-prone modules	Metrics related to the amount of data and the structural complexity (DE) of programs (Shen et al. 1985)
[S4]	Software complexity metrics	The relationship between program complexity measures and program faults (Munson and Khoshgoftaar 1992)
[S5]	Object-oriented methods	A new suite of metrics for OO design (Chidamber and Kemerer 1994)
[S6]	Relative code churn	Relative code churn metrics (Nagappan and Ball 2005)
[S7]	Metrics—performance measures	Network analysis on these dependency graphs (Zimmermann and Nagappan 2008)
[S8]	Network measures	Logistic regression (Ma et al. 2011)
[S9]	Complex network	Sampled K-fold cross-validation method (Yang et al. 2018)

The defect prediction model is established by software measurement data and tagged defect data collected from historical data. Through the collected historical data, the machine learning method is used to construct a software defect prediction model, and then to predict the subsequent software, provide support for decision-making and improve the quality of the software, and ensure the availability and reliability of the software as an infrastructure. The software defect prediction process based on machine learning algorithms is shown in the figure below (Fig. 1).

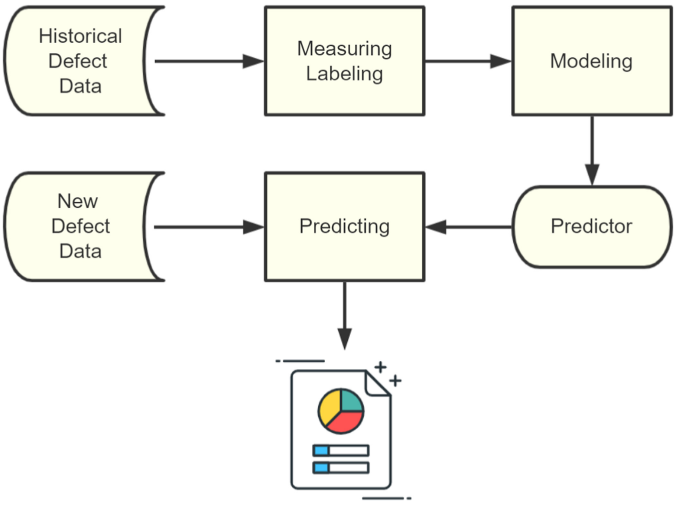

Fig. 1. Defect prediction process

In the 1970s, (Akiyama 1971) based on the hypothesis that complex source code may cause defects, and clearly gave the relationship between software defects and code lines: $D = 4.86 + 0.018L$. (McCabe 1976) and (Halstead 1977) respectively proposed cyclomatic complexity metric and Halstead complexity metric. During this period, the research was not on predictive models, but only on fitting models with the correlation between the number of defects and the measurement.

From the 1980s to the 1990s, (Shen et al. 1985) established a linear regression model and tested it on a new software module. (Munson and Khoshgoftaar 1992) proposed a modular classification model that is divided into high-risk and low-risk categories. (Chidamber and Kemerer 1994) proposed several object-oriented metrics, namely CK metrics, (Basili et al. 1994) used CK metrics to predict defects in object-oriented systems.

Since 2000, because of the popularization of configuration management tools, there have been more and more measurement data in the software development process, and the application of measurement data in defect prediction has become more and more extensive. (Nagappan and Ball 2005) proposed relative code change churn indicators, (Zimmermann and Nagappan 2008) proposed dependency graph indicators. (Yang et al. 2018) established an object-oriented software network based on the relationship between software elements and their evolutionary relationship. (Tian 2020) proposed a software defect prediction model based on program slicing.

3 Cross-Project Defect Prediction Research

Cross-Project Defect Prediction (CPDP) refers to the use of predictive models trained from software metrics of other projects to identify software modules that are prone to defects in software projects.

(Zimmermann et al. 2009) collect data from open-source software and commercial software, and build defect prediction models based on logistic regression algorithms. They considered the recall rate, precision and accuracy values of performance indicators, and assumed success when all indicators were equal to or greater than 0.75. Studies have shown that only 3.4% meet this standard.

(Ma et al. 2011) proposed the "naive Bayesian transfer" method, which uses the naive Bayes method to weight the training data in the training set. The NASA MDP and SOFTLAB data sets are used. Studies have proved that, compared with the nearest neighbor sample selection and the traditional naive Bayes method, the prediction model produces better recall and PF results without any further processing.

(Pan 2013) proposed a transfer defect learning method that uses the existing method to transfer component analysis (TCA), and then performs TCA+by extending TCA. They studied different data normalization techniques. TCA+applies automatic selection for the best standardization strategy. The research concluded that z-score normalization provides better results than no normalization.

(Goel et al. 2018) This article takes the multi-class/polynomial classification of cross-item defect prediction as an example. Use set-based statistical models-gradient enhancement and random forest for classification. To determine the performance of polynomial classification for cross-project defect prediction, an empirical study was conducted. According to the number of defects, the class level information can be divided into one of three defined multi-class class 0, class 1 and class 2. The conclusion is: multiple/multi-category classification is applicable to cross-project data, and the results are comparable to project defect data.

(Li et al. 2019) First, a new domain adaptation method based on subspace alignment (SA) is introduced in CPDP, which can reduce the difference in data allocation between the source item and the target item. Then, a discriminant SA (DSA) method is proposed for CPDP, which can make full use of the class label information of the source item. The experimental results of five public projects in the NASA data set show that DSA is superior to related competitive methods.

(Gong et al. 2020) A new class imbalance learning method is proposed for the problem of class imbalance within and across projects. The conclusion of the study is that this method has better area under the curve (AUC), recall rate and F measurement. (Zhu et al. 2020) proposed an improved Transfer Naive Bayes (ITNB) based on Naive Bayes. The conclusion of the study is that in WPDP and CPDP defect prediction, the accuracy and accuracy of the ITNB model have achieved better results. (Wang 2020) This article focuses on the comparative study of oversampling and ensemble learning methods in software defect prediction. Aiming at the treatment of category imbalance in software defect prediction, it is studied how to effectively combine the imbalance processing method at the data level and the imbalance processing method at the algorithm level to obtain better defect prediction performance. Aiming at the problem of category imbalance in defect prediction, a hybrid sampling technique HS SKM based on S MODE and X-mode s clustering is proposed, and this technique is combined with the traditional random forest algorithm to obtain a Hybrid sampling random forest algorithm HSRF. Aiming at the problem of feature selection in defect prediction, a feature selection algorithm based on conditional information entropy and random subspace, FSC ERS, is proposed, and an integrated learning algorithm combining this algorithm with Bootstrap sampling.

4 Conclusion

From the above research status at home and abroad, it can be seen that in recent years, many studies have been carried out on cross-project software defect prediction, and according to the characteristics of cross-project software data, various algorithm improvements have been used to improve the accuracy of defect prediction. Several key issues of the existing cross-project software defect prediction technology have not been effectively resolved, and can be used as a direction for future research considerations.

1. The problem of imbalanced training data classification
   Classification imbalance is the main factor that affects the quality of the data set. The defective data set contains defective modules and non-defective modules. Among them, defective modules often belong to a minority category, while non-defective modules belong to a majority category. Traditional classification models often aim to maximize the overall classification accuracy, but reduce the classification accuracy of minority classes, which will affect the performance of the defect prediction model to a certain extent.
2. The problem of labeling cost of unlabeled training data
   Marking defective modules is costly. How to select samples that can build a better predictive model for marking without marking samples, avoid marking the entire software project module, thereby reducing the high cost of marking samples.
3. Data distribution of source and target projects in cross-project defect prediction
   For the defect data of different projects, due to the different programming styles of the developers of different projects, the different development platform environments, and the different functions and complexity of the projects, the distribution of the defect data of different projects will be very different. In order to construct an effective defect prediction model, reducing the difference in the distribution of defect data of different projects is the key.

Acknowledgement. The work was supported by Macau Foundation, Project number: MF2012.

References

Akiyama, F.: An example of software system debugging. In: Freiman, C.V., Griffith, J.E., Rosenfeld, J.L. (eds.) IFIP Congress, no. 1, pp. 353–359. North-Holland (1971). ISBN: 0-7204-2063-6

Basili, R., Marziali, A., Pazienza, M.T.: Modelling syntactic uncertainty in lexical acquisition from texts. J. Quant. Linguist. **1**(1), 62–81 (1994). https://doi.org/10.1080/09296179408590000

Chidamber, S.R., Kemerer, C.F.: A metrics suite for object oriented design. IEEE Trans. Softw. Eng. **20**(6), 476–493 (1994). https://doi.org/10.1109/32.295895

Goel, L., Sharma, M., Khatri, S., Damodaran, D.: Prediction of cross project defects using ensemble based multinomial classifier. ICST Trans. Scalable Inf. Syst. 159974 (2018). https://doi.org/10.4108/eai.13-7-2018.159974

Gong, L., Jiang, S., Bo, L., Jiang, L., Qian, J.: A novel class-imbalance learning approach for both within-project and cross-project defect prediction. IEEE Trans. Reliab. **69**(1), 40–54 (2020). https://doi.org/10.1109/TR.2019.2895462

Halstead, M. H.: Elements of Software Science, Operating, and Programming Systems Series. Elsevier Science, 7 (1977)

Li, Z., Qi, C., Zhang, L., Ren, J.: Discriminant subspace alignment for cross-project defect prediction. In: Proceedings of the 2019 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Internet of People and Smart City Innovation, SmartWorld/UIC/ATC/SCALCOM/IOP/SCI 2019, pp. 1728–1733 (2019). https://doi.org/10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00308

Ma, Y., Luo, G., Li, J., Chen, A.: Software defect prediction using transfer method. In: 2011 International Conference on Computational Problem-Solving, ICCP 2011 (2011). https://doi.org/10.1109/ICCPS.2011.6092261

Mccabe, T. J.: A Complexity Measure. IEEE Trans. Softw. Eng. **SE-2**(4) (1976). https://doi.org/10.1109/TSE.1976.233837

Munson, J.C., Khoshgoftaar, T.M.: The detection of fault-prone programs. IEEE Trans. Softw. Eng. **18**(5), 423–433 (1992). https://doi.org/10.1109/32.135775

Nagappan, N., Ball, T.: Use of relative code churn measures to predict system defect density. In: Proceedings of the 27th International Conference on Software Engineering, ICSE 2005 (2005). https://doi.org/10.1145/1062455.1062514

Nagappan, N., Murphy, B., Basili, V.R.: The influence of organizational structure on software quality: an empirical case study. In: Proceedings of the International Conference on Software Engineering (2008). https://doi.org/10.1145/1368088.1368160

Pan, S.J.: Transfer defect learning. In: Proceedings of the International Conference on Software Engineering, pp. 382–391, 22 May 2013

Shen, V.Y., Yu, T.J., Thebaut, S.M., Paulsen, L.R.: Identifying error-prone software—an empirical study. IEEE Trans. Softw. Eng. **SE-11**(4), 317–324 (1985). https://doi.org/10.1109/TSE.1985.232222

Tian, Y.: Research on software defect prediction based on program slice (2020)

Wang, H.: Research on software defect predication based on ensemble learning (2020)

Yang, Y., Ai, J., Wang, F.: Defect prediction based on the characteristics of multilayer structure of software network. In: Proceedings of the 2018 IEEE 18th International Conference on Software Quality, Reliability, and Security Companion, QRS-C 2018 (2018). https://doi.org/10.1109/QRS-C.2018.00019

Zhu, K., Zhang, N., Ying, S., Wang, X.: Within-project and cross-project software defect prediction based on improved transfer Naive Bayes algorithm. Comput. Mater. Contin. **63**(2), 891–910 (2020). https://doi.org/10.32604/cmc.2020.08096

Zimmermann, T., Nagappan, N.: Predicting defects using network analysis on dependency graphs. In: Proceedings of the International Conference on Software Engineering (2008). https://doi.org/10.1145/1368088.1368161

Zimmermann, T., Nagappan, N., Gall, H., Giger, E., Murphy, B.: Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. In: ESEC-FSE 2009 - Proceedings of the Joint 12th European Software Engineering Conference and 17th ACM SIGSOFT Symposium on the Foundations of Software Engineering (2009). https://doi.org/10.1145/1595696.1595713