A Study on Software Defect Prediction Using Classification Techniques

Suneetha Merugula
Dept. of Information Technology, GMR Institute of Technology, Rajam, India
sunita.merugula@gmail.com

Abstract — Software defect prediction is the process of developing models for software projects is helpful for reducing the effort in locating defects. Software defect prediction aids to improve testing resources allocation by detecting defect-prone modules prior to testing. There are various classification techniques used for predicting defects. Software defect prediction models are built based on a different set of metrics and defect data of previous software release. The main objective of this paper is to help developers to identify defects based on existing software metrics using classification techniques and there by improve the software quality.

Keywords- Defect Prediction, Classification Techniques, Software Metrics.

I. INTRODUCTION

The software development life cycle generally includes analysis, design, implementation, test and release phases. The testing phase should be operated effectively in order to release bug-free software to end users. Software Reliability is an important aspect of any software system Large software systems require consistent upgrading that tries to correct the stated defects in previous Versions and add some functions to meet new requirements. The software development team tries to increase the software quality by decreasing the number of defects as much as possible. Software defect prediction helps to optimize testing resources allocation by identifying defect-prone modules prior to testing. Software defect prediction plays an important role in improving software quality and the prediction of defect where probable to occur in software can help to reduce the time and costs for software testing. Building defect prediction models for software projects is helpful for reducing the effort in locating defects. Software defect prediction models are built based supervised learning and unsupervised learning.

II. OVERVIEW OF SOFTWARE DEFECT PREDICTION

Models are built for two different defect prediction tasks within project defect prediction and cross-project defect prediction.

The defect prediction models are all built using historical data from past projects, Software metrics and past defect information can be collected from software archives such as version control systems and defect report systems. Software metrics measure complexity of software and its development process each instance can be labeled by past defect information. . With these labeled instances, we can build a prediction model and predict the unlabeled instances. This prediction is conducted within the same project. So, we call this Within-project defect prediction (WPDP).

Some projects do not have defect data because of lacking in historical data from software stores. So, in some real industrial projects, we cannot generate labeled instances to build a prediction model. Without labelled instances, we cannot build a prediction model. Defect prediction techniques train prediction models by using data from mature projects or called source projects, and use the trained models to predict defects for new projects or called target projects. So, We call this cross project defect prediction (CPDP).

III. LITERATURE REVIEW

A survey is conducted to help developers identify defects based on existing software metrics using data mining techniques especially Classification and there by improve software quality which leads to reduction in the software development cost in the development and maintenance phase. Different classification techniques have been surveyed with different data sets. Yuan Chen, et.al [1] have surveyed the different data mining classification techniques for software defect prediction. They proposed a new model based on Bayesian network and PRM to predict the software defect and manage. Hassan Najadat and IzzatAlsmadi [2] Proposed a new model based on Ridor algorithm to predict fault in modules. They also tested the different classification techniques on the data sets provided by NASA. The results shown that Ridor algorithm is better than the existing technique in terms of accuracy and extraction of number of rules. Ahmet Okutan,OlcayTanerYıldız [3], Introduced a new two metrics NOD, for the number of developers and LOCQ for source code quality apart from the metrics which is available in Promise data repository. Using Bayesian network classifier experimental shows that NOC &DIT have very limited and untrustworthy. LOCQ is more effective like CBO & WMC. NOD metric showed that there is a positive correlation between the no of developers and extent of defect prunes. LOC is proved to be one of the best metric for quick defect prediction. LCOM3 & LCOM have less effective compared to LOC, CBO, RFC, and LOCQ&WMC. Thair Nu Phyu [4] reviewed on various classification techniques such as decision tree induction, Bayesian networks, k-nearest neighbor classifier, case-based reasoning, genetic algorithm and fuzzy logic techniques. The results found that there is no proper info that which is the best classifier. Several of the classification methods produce a set of interacting loci that best predict the phenotype. However, a straightforward application of classification methods to large numbers of markers has a potential risk picking up randomly associated markers. Wen Zhang et.al [5] proposed Bayesian Regression Expectation Maximize algorithm for software effort prediction and two embedded strategies handle missing data. They used the method of ignoring the missing data in an iterative manner in the predictive model. Here they have used data sets such as ISBSG and CSBSG. When there are no missing data BREM outperforms CR, BR, and SVR& M5. When there are missing data BREM with MDT and MDI outperforms imputation technique includes MI, BMI,CMI, and Mini& M5. BRM is used for software prediction and MDI used for finding missing values embedded with BREM. Arvinder Kaur and Inderpreet Kaur [6] , they have tried to find the quality of the software product based on identifying the defects in the classes. They have done this by using six different classifiers such as Naive base, Logistic regression, Instance based (Nearest- Neighbour), Bagging, J48, Decision Tree, Random Forest. This model is applied on five different open source software to find the defects of 5885classes based on object oriented metrics. Out of which they found only Bagging and J48 to be the best. K.Sankar et.al [7], proposed a system which overcomes the problem of insufficiency in accuracy and use of large number of features. This paper proposed Feature selection techniques to predict faults in software code and it also measure the software code and performance of Naive based and SVM classifier. The accuracy is measured by F-mean metric.

IV. CLASSIFICATION TECHNIQUES

In this section, we briefly explain classification techniques.

A. Statistical Techniques

Statistical techniques are based on a probability model [9]. These techniques are used to find patterns in datasets and build diverse predictive models [8]. Instead of simple classification, statistical techniques report the probability of an instance belonging to each individual class (i.e., defective or not) [9].In this paper, we study the Naive Bayes and Simple Logistic statistical techniques. Naive Bayes is a probability-based technique that assumes that all of the predictors are independent of each other. Simple Logistic is a generalized linear regression model that uses a logit link function.

B. Clustering Techniques

Clustering techniques divide the training data into small groups such that the similarity within groups is more than across the groups [11]. Clustering techniques use distance and similarity measures to find the similarity between two objects to group them together. In this paper, we study the K-means and Expectation Maximization clustering techniques. K-means divides the data into k clusters and centroids are chosen randomly in an iterative manner [12]. The value of k impacts the performance of the technique [13]. We experiment with four different k values (i.e., 2, 3, 4, and 5), and found that k = 2 tends to perform the best. We also study the Expectation Maximization [16] (EM) technique, which automatically splits a dataset into an (approximately) optimal number of clusters [10].

C. *Rule-Based Techniques*

Rule-based techniques transcribe decision trees using a set of rules for classification. The transcription is performed by creating a separate rule for each individual path starting from the root and ending at each leaf of the decision tree [14]. In this paper, we study the Repeated Incremental Pruning to Produce Error Reduction (Ripper) and RIpple DOwn Rules (Ridor) rule-based techniques. Ripper [15] is an inductive rulebased technique that creates series of rules with pruning to remove rules that lead to lower classification performance [13]. Ridor [16] is a rule-based decision tree technique where the decision tree consists of four nodes: a classification, a predicate function, a true branch, and a false branch. Each instance of the testing data is pushed down the tree, following the true and false branches at each node using predicate functions. The final outcome is given by the majority class of the leaf node [13].

D. Neural Networks

Neural networks are systems that change their structure according to the flow of information through the network during training [17]. Neural network techniques are repeatedly run on training instances to find a classification vector that is correct for each training set [18]. In this paper, we study the Radial Basis Functions neural network technique. Radial Basis Functions consists of three different layers: an input layer (which consists of independent variables), output layer (which consists of the dependent variable) and the layer which connects the input and output layer to build a model [19].

E. Nearest Neighbour

Nearest neighbour (a.k.a., lazy-learning) techniques are another category of statistical techniques. Nearest neighbour learners take more time in the testing phase, while taking less time than techniques like decision trees, neural networks, and Bayesian networks during the training phase [20].

In this paper, we study the KNN nearest neighbour technique. KNN [21] considers the K most similar training examples to classify an instance. KNN computes the Euclidean distance to measure the distance between instances [22]. We find K = 8 to be the best-performing K value of the five tested options (i.e., 2, 4, 6, 8, and 16).

F. Support Vector Machines

Support Vector Machines (SVMs) use a hyperplane to separate two classes (i.e., defective or not). The number of features in the training data does not affect the complexity of an SVM model, which makes SVM a good fit for experiments where there are fewer training instances than features [20].In this paper, we study the Sequential Minimal Optimization (SMO) SVM technique. SMO analytically solves the large Quadratic Programming (QP) optimization problem which occurs in SVM training by dividing the problem into a series of possible QP problems [23].

G. Decision Trees

Decision trees use feature values for the classification of instances. A feature in an instance that has to be classified is represented by each node of the decision tree, while the assumption values taken by each node is represented by each branch. The classification of instances is performed by following a path through the tree from root to leaf nodes by checking feature values against rules at each node. The root node is the node that best divides the training data [20]. In this paper, we study the Logistic Model Tree (LMT) and J48 decision tree techniques. Similar to model trees (i.e., regression trees with regression functions), LMT [24] is a decision tree like structure with logistic regression functions at the leaves [24]. J48 [50] is a C4.5-based technique that uses information entropy to build the decision tree. At each node of the decision tree, a rule is chosen by C4.5 such that it divides the set of training samples into subsets effectively [25].

H. Ensemble Methods

Ensemble methods combine different base learners together to solve one problem. Models trained using ensemble methods typically generalize better than those trained using the standalone techniques [27]. In this paper, we study the Bagging, Adaboost, Rotation Forest, and Random Subspace ensemble methods. Bagging (Bootstrap Aggregating) is designed to improve the stability and accuracy of machine learning algorithms. Bagging predicts an outcome multiple times from different training sets that are combined together either by uniform averaging or with voting [26]. Adaboost performs multiple iterations each time with different example weights, and gives a final prediction through combined voting of techniques [26]. Rotation Forest applies a feature extraction method to subsets of instances to reconstruct a full feature set for each technique in the ensemble. Random Subspace [25] creates a random forest of multiple decision trees using a random selection attribute approach. A subset of instances is chosen randomly from the selected attributes and assigned to the learning technique [26].

CONCLUSION

Software defect prediction is the process of tracing defective components in software prior to the start of testing phase. Occurrence of defects is inevitable, but we should try to limit these defects to minimum count. Defect prediction leads to reduced development time, cost, reduced rework effort, increased customer satisfaction and more reliable software. Software defect prediction model helps in early detection of defects using Classification Technique. In this paper we have discussed the various classification techniques.

REFERENCES

[1] Y. Chen, P. Du, Xi , X.-H. Shen, “Research on Software Defect Prediction Based on Data Mining”, Computer and Automation Engineering (ICCAE), 2nd International Conference, (2010), vol. 1, pp. 563- 567.

[2] H. Najadat and I. Alsmadi, “Enhance Rule Based Detection for Software Fault Prone Modules”,International Journal of Software Engineering and Its Applications, vol. 6, no. 1, (2012).I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[3] A.Okutan, O. T.Yildiz, “Software defect prediction using Bayesian networks”, Empirical Software Engineering, (2014), vol. 19, no. 1, pp. 154-181.R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[4] T. Nu Phyu, “Survey of Classification Techniques in Data Mining”, International Multi Conference of Engineers and Computer Scientists, (2009); Hong Kong.

[5] W. Zhang, Y. Yang, Q. Wang, “Using Bayesian Regression and EM algorithm with missing handling for software effort prediction”, Information and software technology, vol. 58, (2015), pp. 58-70.

[6] A. Kaur and I. Kaur, “Empirical Evaluation of Machine Learning Algorithms for Fault Prediction”, Lecture Notes on Software Engineering, vol. 2, no. 2, (2014).

[7] K. Sankar, S. Kannan and P.Jennifer, “Prediction of Code Fault Using Naive Bayes and SVM Classifiers Middle-East Journal of Scientific Research”, vol. 20, no. 1, (2014), pp.108-113.

[8] A. Berson, S. Smith, and K. Thearling, “An overview of data mining techniques,” Building Data Mining Applica- tion for CRM, 2004.

[9] S. Kotsiantis, “Supervised machine learning: A review of classification techniques,” Informatica, vol. 31, pp. 249–268, 2007.

[10] N. Bettenburg, M. Nagappan, and A. E. Hassan, “Think locally, act globally: Improving defect and effort prediction models,” in Proceedings of the 9th IEEE Working Conference on Mining Software Repositories. IEEE Press, 2012, pp. 60–69.

[11] K. Hammouda and F. Karray, “A comparative study of data clustering techniques,” Fakhreddine Karray Univer- sity of Waterloo, Ontario, Canada, 2000.

[12]] G. Hamerly and C. Elkan, “Learning the k in k-means,” in Advances in Neural Information Processing Systems, 2004, pp. 281–288.

[13] B. Lemon, “The effect of locality based learning on software defect prediction,” Ph.D. dissertation, West Virginia University, 2010.

[14] S. Kotsiantis, “Supervised machine learning: A review of classification techniques,” Informatica, vol. 31, pp. 249– 268, 2007.

[15] W. W. Cohen, “Fast Effective Rule Induction,” in Pro-ceedings of the International Conference on Machine Learning, 1995, pp. 115–123.

[16] B. R. Gaines and P. Compton, “Induction of ripple-downrules applied to modeling large databases,” Journal of Intelligent Information Systems, vol. 5, no. 3, pp. 211– 228, 1995.

[17] Y. Singh and A. S. Chauhan, “Neural networks in data mining,” Journal of Theoretical and Applied Information Technology, vol. 5, no. 6, pp. 36–42, 2009.

[18] T. G. Dietterich, “An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization,” Machine learning, vol. 40, no. 2, pp. 139–157, 2000.

[19] A. Panichella, R. Oliveto, and A. De Lucia, “Crossproject defect prediction models: L’union fait la force,” inProceedings of the Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE). IEEE, 2014, pp.164–173.

[20] T. G. Dietterich, “An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization,” Machine learning, vol. 40, no. 2, pp. 139–157, 2000.

[21] T. Cover and P. Hart, “Nearest neighbour pattern classification, ”IEEE Transactions on Information Theory,vol. 13, no. 1, pp. 21–27, 1967.

[22] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch,“Benchmarking classification models for software defect prediction: A proposed framework and novel findings, ”IEEE Transactions on Software Engineering, vol. 34,no. 4, pp. 485–496, 2008.

[23]] J. C. Platt, “Fast training of support vector machines using sequential minimal optimization,” in Advances in kernel methods. MIT Press, 1999, pp. 185–208.

[24] N. Landwehr, M. Hall, and E. Frank, “Logistic model trees,” Machine Learning, vol. 59, no. 1-2, pp. 161–205,2005.

[25] L. Sehgal, N. Mohan, and P. S. Sandhu, “Quality prediction of function based software using decision tree approach,” in Proceedings of the International Conference on Computer Engineering and Multimedia Technologies (ICCEMT), 2012, pp. 43–47.

[26] T. Wang, W. Li, H. Shi, and Z. Liu, “Software defect prediction based on classifiers ensemble,” Journal of Information & Computational Science, vol. 8, no. 16, pp. 4241–4254, 2011.

[27] Z.-H. Zhou, “Ensemble learning,” in Encyclopedia of Biometrics. Springer, 2009, pp. 270–273.