INFLUENCES OF MACHINE LEARNING ALGORITHM FACTORSING SOFTWARE FAULT PREDICTION

P.Sivapadmini, Dharani
Assistant Professor, UG Scholar
Department of CSE
Marri Laxman Reddy Institute of Technology
Hyderabad

Abstract

Software Engineering is a branch of computer science that enables tight communication between system software and training it as per the requirement of the user. We have selected seven distinct algorithms from machine learning techniques and are going to test them using the data sets acquired for NASA public promise repositories. The results of our project enable the users of this software to bag up the defects are selecting the most efficient of given algorithms in doing their further respective tasks, resulting in effective results.

1. INTRODUCTION

Nowadays developing software systems is a difficult process that involves planning, analyzing, designing, implementing, testing, integrating, and maintaining. A software engineer's work is developing a system in time with a limited budget which is done in the planning phase. While doing the development process we can have a few defects like not proper design, where the logic is poor, data handling is improper, etc. and these defects cause errors which lead to re-do the work, increasing in development and cost of maintenance These all are responsible for the decrease in customer satisfaction. In this point of view, faults are grouped on the basis of sternness, corrective and advanced actions are taken as per the sternness defined. The selected machine learning algorithms for comparison are

(i) Support Vector Machine (SVM): This algorithm tries to model data so that the examples of separate categories are divided by a clear gap that is as wide as possible. Apart from linear classification, they can also perform non-linear classification efficiently.

(ii) Random Forests (RF): It is an ensemble learning method consisting of multiple decision trees and outputs the class that is the mode or mean of the trees' prediction. While building, it uses feature randomness to ensure that the individual trees are uncorrelated.

(iii) Naive Bayes: It is a probabilistic classifier based on the Bayes theorem, which works on the primary assumption that features are conditionally independent.

(iv) Radial Basis Function (RBF)

(v) Multilayer Perceptron (MLP)

(vi) Multinomial Naive Bayes

(vii) Bagging

1.1 Problem statement:

There are a great variety of studies that have developed and applied statistical and machine learning based models for defect prediction in software systems. we have used logistic regression to examine the effect of the suite of object-oriented design metrics on the prediction of fault-prone classes and used the neural network in toe modules of large telecommunication systems as fault-prone or not and compared it with a non-parametric discriminant model. The results of their study have shown that compared to the non-parametric discriminant model, the predictive accuracy of the neural network model had a better result. Then made a case study by using regression trees to classify fault-prone modules of enormous telecommunication systems

1.2 Motivation:

Developing a software system is an arduous process that contains planning, analysis, design, implementation, testing, integration, and maintenance. A software engineer is expected to develop a software system on time and within a limited budget predetermined during the planning phase. During the development process, there can be some defects such as improper design, poor functional logic, improper data handling, wrong coding, etc. and these defects may cause errors in whitehead to rework, increases in development and maintenance costs decrease in customer satisfaction.

1.3 Objective:

Software Engineering is a comprehensive domain since it requires tight communication between system stakeholders and delivering the system to be developed within a determined time and a limited budget. Delivering the customer requirements include securing high performance by minimizing the system. Thanks to the effective prediction of system defects on the front line of the project life cycle, the project's resources and the effort of the software developers can be allocated more efficiently for system development and quality assurance activities

2. LITERATURE SURVEY

The rapid growth of search machine learning has resulted in the creation of different learning algorithms that can be used across different applications. Additionally, the ability of machine learning algorithms to solve real-world problems will often determine its ultimate value making the reproduction and application of algorithms in new tasks critical to the field's progress. However, the current research

landscape features numerous publications regarding software fault prediction model development. These can be placed into categories based on ensemble, clustering, and classification methods.

Use this document as a template by simply typing your text into it.

2.1 BASED ON CLASSIFICATION METHODS

EzgiErturk the data set for the experiment are collected from the PROMISE Software Engineering Repository and applied McCabe software metrics. The algorithm they are using in the experiment was SVM, ANN, and ANFIS (new adaptive model proposed), the performance measure was 0.7795, 0.8685, and 0.8573 respectively. Another successful paper was published by Surndha Naidu et al., the primary goal of this paper was to find the total number of defects in order to minimize time and cost. The defect was classified into five parameters Volume, Program length, Difficulty, Effort, and Time Estimator. They were using the ID3 classification algorithm, to classify defects. Malkit Singh et al. Explored the fault-prone of software early software testing by developing a model with the Levenberg-Marquardt algorithm-based neural network tool for data collected from the PROMISE repository of empirical software engineering data and then comparing the accuracy of LM with the polynomial function-based neural network. The experiment showed that Levenberg-Marquardt (LM) has higher accuracy (88.1%). So, that neural network-based machine learning has good accuracy. Saiqa Aleem et al., in this study they were used around fifteen data sets (AR1, AR6, CM1, KC1, KC3, etc.) with various machine learning methods. Measured the performance of each method and finally conclude that SVM, MLP, and bagging had high accuracy and performance According to Venkata U.B et al. They evaluate different predictive models for real-time software defect datasets experiment showed that there are not any exact techniques exist for each data set; however, IBL and 1 R were relatively better consistency in prediction accuracy compared to with other methods. According to Martin Shepperd et al. investigation, in order to predict and evaluate software defects, they were using a novel benchmark framework. In the evaluation stage, different learning schemes are evaluated according to the 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 defects in the new data.

2.2 BASED ON CLUSTERING METHODS

Xi Tan et al. experiments the software defect prediction model based on the function cluster for the purpose of improving the performance of the model. After applying this method, the researcher upgrades the performance 31.6
% to 99.2% recall and precision from 73.8 % to 91.6%. Jaspreet Kaur et al. investigated that, the fault proneness of object-oriented programming via applying k-mean based clustering approach and finally they conclude as they have got 62.4% of accuracy. Model building using clustering algorithms (EM and X-means) from three promise repository data (AR3, AR4, AR5) with an objective of predicting software faults. The first thing in the experiment setting was normalizing the data set into 0 to 1, then an attribute selection algorithm was applied that was Cfs SubsetEval and without attribute reduction. Experiment results showed that X-mean more have accuracy (90.48another model AR3 without attribute reduction.

BASED ON ENSEMBLE APPROACHES

Model building using an ensemble approach was conducted by Shanthini et al., The purpose of this research was to address software fault prediction using an ensemble approach. The data set was categorized into three which are method level, class level, and package levels. They were using NASA KC1 data for both method and class level metrics and eclipse data for package level with an ensemble method (bagging, boosting, stacking, and voting). The experiment result shows that bagging performs better for method and package level data. Method level result using AUC- curve performance measurement was bagging (0.809), boosting (0.782), staking (0.79), and voting (0.63). Similarly, the performance measure of package-level data using AUC-Curve was bagging (0.82), boosting (0.78), staking (0.72), and voting (0.76). In the case of class-level metrics, the performance result could not be similar to other metrics using AUC-Curve bagging (0.78), boosting (0.74), staking (0.8), and voting (0.82). According to Arvinder Kaur et al., the main purpose of the research was to evaluate the application of random forest for predicting fault-prone classes using open-source software. The researcher used JEdit open-source software with object-oriented metrics to conduct studies. Based on the experiment result the accuracy of RF is 74.24 % and its precision is 72 %, its recall is 79 %, its F-measure is 75 %, and its AUC is 0.81. YI PENG et al., The goal of the paper was to assess the quality of ensemble approaches in software fault prediction with an analytical hierarchal process. The researcher uses 13 different performance measures for 10 publicly NASA MDP data. An ensemble method in his paper was Bagging, Boosting, and stacking Based on the performance measure AdaBoost of decision tree gives the best result accuracy of 92.53 %, in this case, decision tree as base classifier.

3. SYSTEM ANALYSIS

3.1 Existing System:

There are a great variety of studies that have developed and applied statistical and machine learning-based models for defect prediction in software systems. Logistic regression was used to examine the effect of the suite of object-oriented design metrics on the prediction of fault-prone classes. The neural network was used in toe modules of large telecommunication systems as fault prone or not and compared with a non-parametric discriminant model. The results of the study have shown that compared to the non-parametric discriminant model, the predictive accuracy of the neural network model had a better result.

3.2 Proposed System: In this paper, the author is evaluating the performance of various machine learning algorithms such as SVM, Bagging, Naïve Bayes, Multinomial Naïve Bayes, RBF, Random Forest, and Multilayer Perceptron Algorithms to detect bugs or defects from Software Components. Defects will occur in software

components due to poor coding which may increase software development and maintenance cost and this problem leads to dissatisfaction from customers. To detect defects from software components various techniques were developed but right now machine learning algorithms are gaining lots of popularity due to their better performance. So, in this paper, also the author is using machine learning algorithms to detect defects from software modules. In this, the paper author is a use dataset from NASA Software components and the name of those datasets are CM1 and KC1. I am also using the same datasets to evaluate the performance of above mention algorithms.Advantages of the proposed system:

1. Predicted model is used for evaluating the performance measures.
2. We can apply various datasets in this project. But we are using NASA datasets in our project.
3. Software defects are classified to the extent.
4. Advance measures can be taken in the selection of algorithm
5. Provides Better results.
6. Identify defects in the early stage of the project which in turn result in customer loyalty.

4. CONCLUSION & FUTURE WORK

In this experimental study, seven machine learning algorithms are used to predict the defectiveness of software systems before they are released to the real environment and/or delivered to the customers, and the best category which has the most capability to predict the software defects are tried to find while comparing them based on software quality metrics which are accuracy, precision, recall, and F- measure. We carry out this experimental study with four NASA datasets which are PC1, CM1, KC1, and KC2. These datasets are obtained from the public PROMISE repository.

The results of this experimental study indicate that tree- structured classifiers in other words ensemble learners which are Random Forests and Bagging have better defect prediction performance compared to their counterparts. Especially, the capability of Bagging in predicting software defectiveness is better. When applied to all datasets, the overall accuracy, precision, recall and F-measure of Bagging is within 83,7- 94,1%, 81,3-93,1%, 83,7- 94,1% and 82,4-92,8% respectively.

For the PC1 dataset, Bagging outperforms all other machine learning techniques in all quality metrics. However, Naive Bayes outperforms Bagging in precision and F-Measure while Bagging outperforms it in accuracy and recall for the CM1 dataset. Random Forests outperforms all machine learning techniques in all quality metrics for the KC1 dataset. Finally, for the KC2 dataset, MLP outperforms all machine learning techniques in all quality metrics for the KC2 dataset.

5. REFERENCES

[1] Victor R Basili, Lionel C. Briand, and Walcelio L Melo. ´ A validation of object-oriented design metrics as quality indicators. IEEE Transactions on software engineering, 22(10):751–761, 1996.

[2] Evren Ceylan, F Onur Kutlubay, and Ayse B Bener. Software defect identification using machine learning techniques. In 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO'06), pages 240–247. IEEE, 2006.

[3] Karim O Elish and Mahmoud O Elish. Predicting defectprone software modules using support vector machines. Journal of Systems and Software, 81(5):649– 660, 2008.

[4] Norman Fenton, Paul Krause, and Martin Neil. Software measurement: Uncertainty and causal modeling. IEEE software, 19(4):116–122, 2002.

[5] Lan Guo, Yan Ma, Bojan Cukic, and Harshinder Singh. Robust prediction of fault-proneness by random forests. In 15th International Symposium on Software Reliability Engineering, pages 417–428. IEEE, 2004.

[6] Taghi M Khoshgoftaar, Edward B Allen, and Jianyu Deng. Using regression trees to classify fault-prone software modules. IEEE Transactions on reliability, 51(4):455–462, 2002.

[7] Taghi M Khoshgoftaar, Edward B Allen, John P Hudepohl, and Stephen J Aud. Application of neural networks to software quality modeling of a very large telecommunications system. IEEE Transactions on Neural Networks, 8(4):902–909, 1997.

[8] Sunghun Kim, Hongyu Zhang, Rongxin Wu, and Liang Gong. Dealing with noise in defect prediction. In 2011 33rd International Conference on Software Engineering (ICSE), pages 481–490. IEEE, 2011.

[9] Yan Ma, Lan Guo, and Bojan Cukic. A statistical framework for the prediction of fault-proneness. In Advances in Machine Learning Applications in Software Engineering, pages 237–263. IGI Global, 2007.

[10] Ruchika Malhotra. A systematic review of machine learning techniques for software fault prediction. Applied Soft Computing, 27:504–518, 2015.

[11] Jinsheng Ren, Ke Qin, Ying Ma, and Guangchun Luo. On software defect prediction using machine learning. Journal of Applied Mathematics, 2014, 2014.