ISSN No. 0976-5697

Volume 5, No. 1, Jan-Feb 2014

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

# Use of Artificial Intelligence Technique in Software Fault Prediction

Anurag Shrivastava
(M. Tech Scholar)
Department of Computer Science & Engineering,
Arya college of engineering & I.T. Jaipur, India
nrg.shrivastava@gmail.com

Vishal Shrivastava
Associate Professor,
Department of C.S.
Arya college of engineering & I.T. Jaipur, India
vishal1500371@yahoo.co.in

***Abstract:*** Decision tree usually applied for solving both classification and regression problems in many applications. This paper evaluates the capability of Decision Tree in predicting defect-prone software module and compares its prediction performance against three intelligence technique in the context of PC1 dataset. we have used PC1 dataset (NASA dataset) which has sufficient parameters for analysis. As PC1 data is highly unbalanced data different balancing techniques have been applied.

***Keywords****:* Decision tree (DT), *M*ultilayer Perceptron (MLP), Support Vector Machine (SVM),). synthetic minority over- sampling technique (SMOTE)

## I. INTRODUCTION

We have done Study and shown that the majority of defects are often found in only a few software modules[1,2].Such defective software modules may cause software failures, increase development and maintenance costs, and decrease customer satisfaction[3]. software fault prediction models which allow software engineers to focus development activities on fault-prone code, thereby improving software quality and making better use of resources[4]**.** Identification of defect-prone software modules is commonly achieved through binary prediction models that classify a module into either defective or not - defective category. These prediction models almost always utilize static product metrics, which have been associated with defects, as independent variables[5]. Recently, DT (Decision Tree) have been introduced as an effective model in both machine learning and data mining communities for solving both classification and regression problems [6,7]. It is therefore motivating to investigate the capability of DT in software fault prediction.

The objective of this paper is to evaluate the capability of DT in predicting defect-prone software modules and compare its prediction performance against three well-known statistical and machine learning models in the context of PC1 NASA datasets. The compared models are one statistical classifiers techniques: (i) Logistic Regression (LR), one neural networks techniques: (i) Multi-layer Perceptrons (MLP) and one tree structured classifiers techniques:(i) Decision Trees (DT). For more details on these techniques see[8,9,10,11].

The rest of this paper is organized as follows. Section 2 reviews the research done in the field of software fault prediction. Section 3 overviews the data description and data preparation, Section 4 overviews the techniques applied in this paper, section 5 presents the results and discussions. Finally, Section 6 concludes the paper.

## II. LITERATURE SURVEY

Software reliability is a critical field in software and an important fact of software quality. Every organization wants to assess the quality of the software product as early as possible so that poor software design leading to lower quality product can be detected and hence be improved or redesigned[12]. This would lead to significant savings in the development costs, decrease the development time, and make the software more reliable. A wide range of statistical and machine learning techniques have been developed and applied to predict defects in software. Basili et al. investigated the impact of the suite of object -oriented design metrics on the prediction of fault-prone classes using logistic regression[13]. Khoshgoftaar et al. investigated the use of the neural network as a model for predicting software quality. They used large telecommunication system to classify modules as fault prone or not fault-prone[14]. Disadvantage of neural network that it is learning in form of weights and human cannot interpret any knowledge from the weights. Khoshgoftaar et al. applied regression trees with classification rule to classify fault-prone software modules using a very large telecommunications system as a case study[15]. the advantage of DT is giving the if then rules that is human readable and understandable format. By using these rules we can design early warning prediction system for software fault prediction.

In recent years, a number of alternative modeling techniques have been proposed for software fault prediction. Alternative models include artificial neural networks, analogy-based reasoning, fuzzy system and ensemble techniques. Ensemble is used to combine the result of individual methods [16,17]. Unfortunately the accuracy of these models is not satisfactory so there is always a scope for new software fault prediction techniques.

## III. DATA DESCRIPTION AND DATA PREPARATION

The dataset used in this study is mission critical NASA software projects, which are publicly accessible from PROMISE Software Engineering Repository. data set made publicly available in order to encourage repeatable, verifiable, refutable, and/or improvable predictive models of software engineering. dataset contains 1109 total Number of instances, each instances contains 21 software metrics (independent variables) and the associated dependent Boolean variable: Defective (whether or not the module has

© 2010-14, IJARCS All Rights Reserved

any defects). 21 independent metrics further divided in 5 different lines of code measure, 3 McCabe metrics, 4 base Halstead measures, 8 derived Halstead measures, and a branch-count. since data set contain, 93% non-fault and 7% fault instances. It is clear that the dataset is highly unbalanced in terms of proportion of fault vs. non-fault. Consequently, before supplying this data to the intelligence techniques, balancing is done so that the classifier while predicting will not be nfluenced by the majority class, that is, non-fault. The balancing is done using under-sampling and over -sampling for balancing the data set during cleaning and data preparation.

We used the WEKA for SMOTE, implementation available at http://www.softpedia.com/get/Internet/Servers/Database-Utils/Weka.shtml.

Finally, we normalized the data set.

SMOTE is a technique in which the minority class is over-sampled by creating synthetic samples. The minority class is over-sampled by taking out each sample and introducing synthetic samples along the line segments joining any/all of the k minority class nearest neighbours. This approach effectively forces the decision region of the minority class to become more general. Further, under-sampling is a technique in which some of the samples belonging to the majority class are removed and combined with the minority class samples. For example, under-sampling (25%) means that the majority class is reduced to 25% of its original size[18]. Then, over-sampling is a technique in which the samples belonging to the minority class are replicated a few times and combined with the majority class samples. For example, over-sampling (100%) means that the majority class is replicated once. After the dataset are balanced using one of the above-mentioned two methods, intelligence techniques is invoked. Also, ten-fold cross validation is performed throughout the study. Further, sensitivity, specificity and accuracy are computed for each data balancing method.

The quantities employed to measure the quality of the classifiers are sensitivity, specificity and accuracy, which are defined as follows[19]:

Sensitivity is the measure of proportion of the true positives, which are correctly identified.

Sensitivity = True positive / (True positive + False negative)

Specificity is the measure of proportion of the true negatives, which are correctly identified.

Specificity = True negative / (True negative + False positive)

Accuracy is the measure of proportion of true positives and true negatives, which are correctly identified.

Accuracy = (True positive + True negative) / (True positive + True negative +False positive + False negative)

## IV. OVERVIEW OF TECHNIQUES EMPLOYED

In the following, we now present an overview of the techniques applied in this paper.

### *A.* M*ultilayer* perceptron (mlp):

Multilayer Perceptron (MLP) is an example of an artificial neural network. It is used for solving different problems, example pattern recognition, interpolation, etc. It is an advancement to the perceptron neural network model. With one or two hidden layers, they can solve almost any problem. In a popular form of ANN called the multi-layer perceptron (MLP), all nodes and layers are arranged in a feed forward manner. The first or the lowest layer is called the input layer where external information is received. The last or the highest layer is called the output layer where the network produces the model solution. In between, there are one or more hidden layers, which are critical for ANNs to identify the complex patterns in the data. A cyclic arcs from a lower layer to a higher layer connect all nodes in adjacent layers. The parameters (arc weights) of a neural network model need to be estimated before the network can be used for prediction purposes. The process of determining these weights is called training.

### *B. Support vector machine (svm):*

A Support Vector Machine (SVM) is a learning technique that is used for classifying unseen data correctly. For doing this, SVM builds a hyperplane, which separates the data into different categories. The dataset may or may not be linearly separable. By "linearly separable" we mean that the cases can be completely separated (i.e., the cases with one category are on the one side of the hyperplane and the cases with the other category are on the other side).

The SVM is a powerful learning algorithm based on recent advances in statistical learning theory (Vapnik, 1998). SVMs are learning systems that use a hypothesis space of linear functions in a high dimensional space, trained with a learning algorithm from optimization theory that implements a learning bias derived from statistical learning theory (Cristianini and Shawe-Taylor, 2000) . SVMs have recently become one of the popular tools for machine learning and data mining and can perform both classification and regression.

### *C. Decision tree(DT):*

A decision tree is a predictive machine-learning model that decides the target value (dependent variable) of a new sample based on various attribute values of the available data.

The internal nodes of a decision tree denote the different attributes, the branches between the nodes tell us the possible values that these attributes can have in the observed samples, while the terminal nodes tell us the final value (classification) of the dependent variable. The J48 Decision tree classifier follows the following simple algorithm. In order to classify a new item, it first needs to create a decision tree based on the attribute values of the available training data. So, whenever it encounters a set of items (training set) it identifies the attribute that discriminates the various instances most clearly. This feature that is able to tell us most about the data instances so that we can classify them the best is said to have the highest information gain. Now, among the possible values of this feature, if there is any value for which there is no ambiguity, that is, for which the data instances falling within its category have the same value for the target variable, then we terminate that branch and assign to it the target value that we have obtained.

## V. RESULTS

Results of above three techniques is as follows”

© 2010-14, IJARCS All Rights Reserved

***A. SVM:***

===Summary ===

| | | |
|---|---|---|
| Correctly Classified Instances | 312 | 90.0931 % |
| Incorrectly Classified Instances | 22 | 09.9069 % |
| Mean absolute error | 0.0691 | |
| Root mean squared error | 0.2628 | |
| Relative absolute error | 53.131 % | |
| Root relative squared error | 103.6412 % | |
| Coverage of cases (0.95 level) | 93.0931 % | |
| Mean rel. region size (0.95 level) | 50 % | |
| Total Number of Instances | 333 | |

=== Confusion Matrix ===

| a | b | <-- classified as |
|---|---|---|
| 0 | 23 | a = TRUE |
| 0 | 310 | b = TRUE |

***B. J-48:***

=== Summary ===

| | | |
|---|---|---|
| Correctly Classified Instances | 312 | 96.0931 % |
| Incorrectly Classified Instances | 23 | 3.9069 % |
| Mean absolute error | 0.1131 | |
| Root mean squared error | 0.2542 | |
| Relative absolute error | 87.0291 % | |
| Root relative squared error | 100.2537 % | |
| Coverage of cases (0.95 level) | 100 % | |
| Mean rel. region size (0.95 level) | 100 % | |
| Total Number of Instances | 333 | |

=== Confusion Matrix ===

| a | b | <-- classified as |
|---|---|---|
| 0 | 23 | a = TRUE |
| 0 | 310 | b= True |

***C. MLP:***

=== Summary ===

| | | |
|---|---|---|
| Correctly Classified Instances | 310 | 93.4925 % |
| Incorrectly Classified Instances | 25 | 6.5075 % |
| Kappa statistic | 0.0547 | |
| Mean absolute error | 0.09 | |
| Root mean squared error | 0.2562 | |
| Relative absolute error | 69.259 % | |
| Root relative squared error | 101.02 % | |
| Coverage of cases (0.95 level) | 95.7958 % | |
| Mean rel. region size (0.95 level) | 55.7057 % | |
| Total Number of Instances | 333 | |

=== Confusion Matrix ===

| a | b | <-- classified as |
|---|---|---|
| 1 | 22 | a = TRUE |
| 3 | 307 | b= TRUE |

From the above result it is clearly seen that Decision Tree that is J 48 gives the best result in compare to other two techniques.

## VI. CONCLUSION

Fault prediction modeling is an important area of research and the subject of many previous studies.These studies typically produce fault prediction models which allow software engineers to focus development activities on fault-prone code, thereby improving software qualityand making better use of resources. The term ‘fault’ is used interchangeably in this study with the terms ‘defect’ or ‘bug’ to mean a static faultin software code. It does not denote a ‘failure’ (i.e. the possible result of a fault occurrence)This models help us many ways. In any software project, there can be a number of faults. It is very essential to deal with these faults and to try to detect them as early as possible in the lifecycle of the project development. Predict the fault from the software helps us to make our software without any fault.

## VII. REFERENCES


[1]. N.Fenton and N.Ohlsson," Quantitative analysis of faults and failures in a complex software system", IEEE Transactions on Software Engineering, 2000, pp.797-814

[2]. A. Koru and J. Tian, " An empirical comparison and characterization of high defect and high complexity modules ", Journal of Systems and Software, 2003, pp. 153-163

[3]. A. Koru and H. Liu, 2005," Building effective defect-prediction models in practice", IEEE Software, 2005, pp. 23– 29.

[4]. Karim O. Elish and Mahmoud O. Elish,"Predicting defect-prone software modules using support vector machines", The Journal of Systems and Software 81,2008,pp.649–660

[5]. K. Emam, S. Benlarbi , N. Goel and S.Rai, " Comparing case-based reasoning classifiers for predicting high risk software components", Journal of Systems and Software 55 (3),2001,pp.301–310.

[6]. L.Breiman,"Random forests", Machine Learning 45,2001,pp.5–32.

[7]. T.Khoshgoftaar, E.Allen and J.Deng,"Using regression trees to classify fault-prone software modules", IEEE Transactions on Reliability 51 (4),2002,pp. 455–462

[8]. J.Han and M.Kamber, "Data Mining: Concepts and Techniques "second ed.,2001,Morgan Kauffman

[9]. D. Hosmer and S. Lemeshow, " Applied Logistic Regression", second ed.,2000 John Wiley & Sons, New York.

[10]. R.Duda,P. Hart and D. Stork," Pattern Classification", second ed.,2001,John Wiley & Sons, New York.

[11]. R.Webb," Statistical Pattern Recognition" second ed.,2002, John Wiley & Sons, New York

[12]. R. Malhotra and A. Jain,"Fault Prediction Using Statistical and Machine Learning Methods for Improving Software Quality",Journal of Information Processing Systems, Vol.8, No.2,2012, pp. 241-262

[13]. V. Basili,L.Briand and W.Melo," A validation of object-oriented design metrics as quality indicators", IEEE Transactions on Software Engineering 22 (10),1996, pp. 751–761.

[14]. T. Khoshgoftaar, E. Allen,J. Hudepohl and S. Aud," Application of neural networks to software quality modeling of a very large telecommunications system", IEEE Transactions on Neural Networks 8(4),1997, pp. 902–909.

[15]. T. Khoshgoftaar, E. Allen and J. Deng," Using regression trees to classify fault-prone software modules" IEEE Transactions on Reliability 51 (4), 2002, pp. 455–462.

[16]. K. Vinay Kumar, V. Ravi and Mahil Carr, “Software Cost Estimation using Soft Computing Approaches,” Handbook on Machine Learning Applications and Trends: Algorithms,


© 2010-14, IJARCS All Rights Reserved

Methods and Techniques, Eds. E. Soria, J.D. Martin, R. Magdalena, M.Martinez, A.J. Serrano, IGI Global, USA, 2009.

[17]. J.S.Pahariya, V. Ravi, M. Carr and M.Vasu,“ Computational Intelligence Hybrids Applied to Software Cost Estimation”, International Journal of Computer Information Systems and Industrial Management Applications, ISSN: 2150-7988 Vol.2,2010, pp.104-112

[18]. D.Anil Kumar and V. Ravi, "Predicting credit card customer churn in banking using data mining", international journal of data analysis techniques and strategies, vol.1(1),2008,pp. 4-28.

[19]. T. Fawcett, "An introduction to ROC analysis", Pattern Recognition Letters, Vol. 27,2006,pp.861–874.

© 2010-14, IJARCS All Rights Reserved