Imperial Journal of Interdisciplinary Research (IJIR)
Vol-3, Issue-6, 2017
ISSN: 2454-1362, http://www.onlinejournal.in

Predicting Root Cause Analysis (RCA) bucket for software defects through Artificial Neural Network

Prof.(Mrs.): L. A. Bewoor, Ankit Agarwal, Niket Gaikwad & Praful Khandagale.

Abstract— *In software development early prediction of defective software module is very important, as it can reduce the overall time, budget and increase the customer satisfaction by helping fulfilling customer requirements. In order to achieve this ,it is essential to develop and implement various test cases which is a time consuming as well as tedious task. In this paper we discuss a new model towards reliability and quality improvement of software systems by predicting fault-prone modules before the testing phase. It focuses on pre-processing of data i.e. data cleaning, followed by dimensional reduction, which aims at reducing the noise and dimensions of the data sets. Further, the system is trained to predict bugs efficiently at an earlier stage. The main goal is to help testers to concentrate their testing efforts to modules which have lesser bugs.*

1. INTRODUCTION.

Day by day software is needed almost everywhere from our house hold appliances to various companies. So it is necessary to increase the quality of the product. It is a challenging task for the companies to deliver reliable and a quality product. A major task in developing a reliable and quality product is the identification of bugs. These bugs degrade the overall reliability and quality of the software product, resulting in unreliable end-product and customer dissatisfaction. The Software Development Lifecycle(SDLC) gives the various phases of software cycle and the order in which these phases are to be executed.

It comprises of the following stages:

1. Requirement gathering & analysis.
2. Design.
3. Implementation or coding.
4. Testing.
5. Deployment.
6. Maintenance.

According to IEEE standards, a 'bug' is an incorrect step, instruction or data in a program. A software bug can be defined as a part of the code which would lead to an error, fault or may result in malfunctioning of the program. Despite proper planning, the occurrence of bug is inevitable.

There are various types of bugs which can be introduced in various phases of SDLC. For Example, in the requirement phase, if some requirements of the client are missed it will lead to development of incomplete product. A bug in the design phase can lead to working of product in a way which was not intended. As we proceed towards next phases of SDLC it becomes more and more harder to predict bugs in the software. It also becomes tough not to break existing functionality, when implementing new features. Software Testing it the process which is done to find the quality of the product and its deviation from the desired outcome, which was identified during the Requirement gathering & Analysis phase. The main intention of testers during this process is to identify the bugs in the software. Testing team can use the predicted defects to efficiently plan, manage and control test execution activities. The evolution of software engineering techniques has led to development of many bug prediction algorithms. These algorithms intend to find the bug-prone areas of the software. Some algorithms make use of statistical methods, some use code metrics; some are based on feature extraction. Analysis of the results produced by these algorithms have proved that these algorithms are successful in reducing the tediousness of the job of testers. Machine learning metrics of accuracy, error rate, precision, time taken, memory used, recall, F-measure etc. have been used to find out the performance of these algorithms.

2. LITERATURE SURVEY.

Studies have found that, of the overall software development process 27% of the total time required for the development of project is consumed by testing. Early prediction of bugs in software proves helpful to Software Testing & Quality Assurance (QA) team & it has been an area for continuous study for a long time. Software Defect prediction is very important in order to provide reliable and quality product. Over the past few years' software fault prediction models have got a lot of attention from developers. Still there is need of fault prediction method which is robust, easy to implement and widely applicable. Bugs can be classified on the basis of their occurrence (frequency) and the severity of the bug. According to severity, bugs can be classified as catastrophic, major, minor, and those with no effect. It is vital to predict and

remove the occurrence of major and catastrophic bugs at least.
Defect analysis is of two types: Classification and prediction.
It includes finding out defect classes and predicting future defect trends. A common approach to predict the defect is as follows:

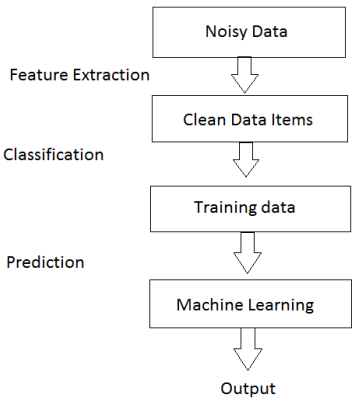

Figure 1

[1] The commonly adapted steps of defect prediction process are:

- ❖ Labeling
- ❖ Pre-Processing
- ❖ Creating Training Sets
- ❖ Prediction and assessment.

Various other approaches like Genetic Programming, Neural Network, Decision Trees, Naive Bayes, Support Vector Machine, etc. have been proposed. The usage of the machine learning has been drastically increased since the year 2005.
It is still one of the favorable methods for fault prediction.
Genetic programming can be used, with software metrics as input and the prediction of number of faults as output. Use of subtractive fuzzy clustering can be used for finding out the number of faults and module-order modeling can be used to find whether or not the modules are fault prone. Naive Bayes algorithm has also got high performance with respect to fault prediction. It is found that various machine learning approaches such as Supervised Learning, Semi-supervised Learning, and Unsupervised Learning are being used for building fault prediction models. Among these, supervised learning approach is mostly used and is found to be more useful for predicting fault-prone modules if sufficient amount of fault data is available. Of the various bug prediction techniques, hybrid techniques have given considerably good results. Hybrid Support Vector Machine (SVM) classifier can be used. Principal Component Analysis (PCA) can be used for feature extraction.

3. PROPOSED WORK.

Bug prediction is needed to identify the different semantics and syntactical attributes analysis for identifying the error and bugs. Therefore, it is a classification and pattern analysis problem. Additionally, a lot of data is available in high dimension attributes which needs to be optimized by enhancing classification accuracy & resource consumption optimization in terms of space and time complexity. The goal here is to perform efficient bug prediction in software, so that the crucial task of testing will become easier. The approach will find applicability in software development companies, to make the task of testing easier and to get a better, reliable and less faulty end product. It will be helpful for both the testers and the end user. It will help testers to make better test cases better and help end user to get a better product.
After the initial survey it has been found that the major problems are:

- ❖ Huge Dimensions: When the data sets have huge dimensions, processing is improper and it becomes difficult to test it.
- ❖ Noisy data: Presence of noise in the relevant data makes the task of testing complex.
- ❖ Cost: The cost of defect correct is marginally high after defect correction.

In order to provide a solution to these problems, it is required to have better data sets. Using Dimension reduction huge dimensions of data sets can be reduced. Using data cleaning techniques noisy data can be removed. To avoid application of the method time and again, the system can be trained using a neural network, and regression analysis can be performed.
Below is the Proposed model for defect prediction:

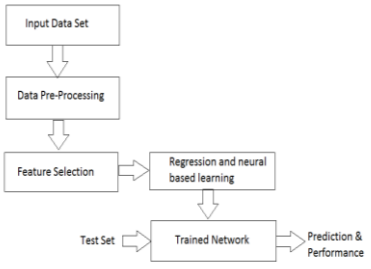

Figure 2

The Main components are discribed in the flow diagram above.

- **Input Training set :** It is simple user interface that accepts the raw training data.
- **Pre-Processing:** In this the training data is normalized & transfromed to find more relevant attributes.
- **Feature Selection:** Initally the data has lots of attributes , so using Principal Component Analysis(PCA) the dimension of the data is reduced to mind appropriate features from the training data.
- **Regression And Neural Network:** Neural Network is an Neural network is an essential data mining tool which can be implementable with the verity of applications such as recognition, pattern detection and other similar data models.But that suffers from the slow learning rate. Therefore the weight adjustment and initialization phase is modified using the regression analysis concept. That may improve the learning capability of the system.

4. PROPOSED ALGORITHM

4.1. PRICIPAL COMPONENT ANALYSIS:

Principal component analysis (PCA) is a statistical procedure.It uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components (or sometimes, principal modes of variation). The number of principal components is less than or equal to the smaller of the number of original variables or the number of observations.
This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components.
Equation 1 shows the formula for finding the Co-Variance.
Figure 1 & 2 show the use of prinicipal component analysis for selecting features from the data set given to it as input.

COVARIANCE-

$$COV(x,y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1}$$

Equation 1

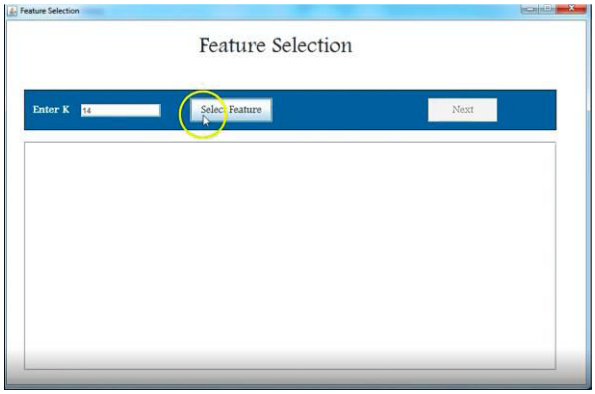

Figure 3

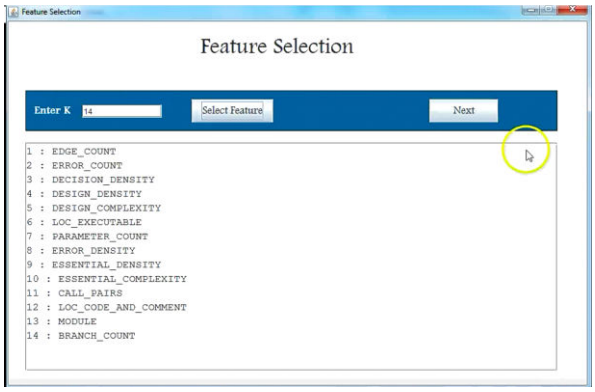

Figure 4

4.2. BACKPROPOGATION:

Backpropagation, is a common method used for training artificial neural networks and is used in conjunction with an optimization method such as gradient descent. The algorithm has two phase cycle mainly, propagation & weight update. When an input vector is given to the network, it is propagated forward through the network, layer by layer, until it reaches the output layer. The output of the network is then compared to the output which is desired, using a loss function, and an error value is calculated for each of the neurons in the output layer. Then these error values are propagated backwards, starting from the output, until each neuron has an associated error value which roughly represents its contribution to the original output.

Phase 1:

Each propagation involves the following steps:

- Forward propagation of a training pattern's input through the neural network in order to generate the network's output Value(s).
- Backward propagation of the propagation's output activations through the neural network using the training pattern target in order to generate the

deltas (the difference between the targeted and actual output values) of all output and hidden neurons.

Phase 2:

For each weight, the following steps must be followed:

- The weight's output delta and input activation are multiplied to find the gradient of the weight.
- A ratio (percentage) of the weight's gradient is subtracted from the weight

Figure 3 shows the output of backpropagation algorithm.

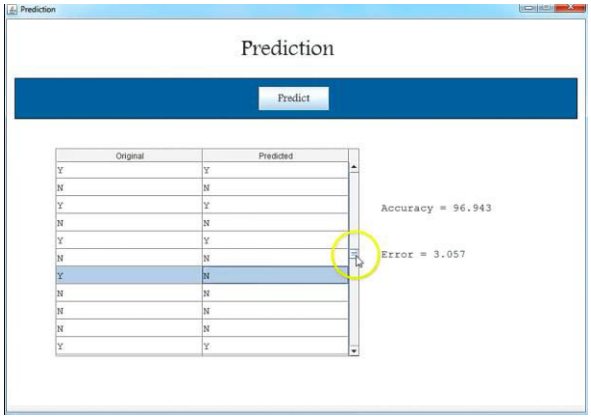

Figure 5

5. CONCLUSION

After the study, it has been observed that early prediction of bugs leads to better resource planning, test planning and ultimately to a reliable and less faulty end product. Thus, the use of such kind of techniques will help the testers and will lead to development of a less faulty product. With the help of regression analysis approach, the occurrence of new faults and re-occurrence of existing faults will not take place. By training the neural network, the learning capacity of the system will be improved, which is a futuristic approach for bug prediction.

6. REFRENCES

[1] Surbhi Parnekar, Ati Jain, Vijay Birchha, "An Approach to Efficient Software Bug Prediction using Regression Analysis and Neural Networks.", International Journal of Innovative Research in Computer and Communication Engineering (An ISO 3297: 2007 Certified Organization) Vol. 3, Issue 10, October 2015.

[2] Proceedings of 2015 Global Conference on Communication Technologies (GCCT 2015)

[3] Grishma BR, Anjali C, "Software Root Cause Prediction using Clustering Techniques: A Review."

[4] Onur Kutlubay, Mehmet Balman, Doğu Gül, Ayşe B. Bener, "A Machine Learning Based Model for Software Defect Prediction."

[5] Marek Leszak, Dewayne E. Perry, Dieter Stoll "A Case Study in Root Cause Defect Analysis."

[6] H.S. Sukhla, Deepak Kumar Verma," A review on software bug prediction."