International Journal of Business Intelligent
Volume: 04 Issue: 02 December 2015,Pages No.85- 90
ISSN: 2278-2400

Machine Learning Based Approach for Predicting Fault in Software Engineering by GMFPA: A Survey

P.Patchaiammal[1], R.Thirumalaiselvi[2]
Research Scholar, Department of Computer Applications,Bharath University,Chennai
[2]Assistant Professor, Department of Computer Science, Govt. Arts College (Men) (autonomous), Chennai
Email: sarandsk1@gmail.com, sarasselvi@gmail.com

Abstract-Fault propensity of software is the prospect that the component contains faults. To forecast fault proneness of modules different techniques have been proposed which includes statistical methods, machine learning techniques, etc. Machine learning techniques can be used to analyze data from different perspectives and enable developers to retrieve useful information. Machine learning techniques are proven to be useful in terms of software bug prediction. This is leading to increase in development of machine learning methods for exploring data sets, which can be used in constructing models for predicting quality attributes such as fault proneness. This research work analyzed various fault prediction techniques and proposed a new algorithm GMFPA (Genetic Metric Fault Prediction Algorithm) to explore the fault prone modules by using metrics. The GMFPA is used to identify fault with highest gain value by forming the hypothesis set.

Keywords:Fault Prediction, Goal-Question-Metrics, Nature of Defect, Genetic Algorithm.

I. INTRODUCTION

Every organization wants to evaluate 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. This would lead to significant savings in the development costs, decrease the development time, and make the software more reliable. The quality of the software can be measured in terms of various attributes such as fault proneness, maintenance effort, testing effort, etc.A software fault prediction is a confirmed technique in accomplishing high software reliability.Prediction of fault-prone modules provides one way to support software quality engineering through improved scheduling and project control. Quality of software is increasingly importantfor software. Although there is diversity in the definition of software quality, it is widely accepted that a project with many defects lack quality. Methodologies and techniques for predicting the testing effort, monitoring process costs, and measuring results can help in increasing efficiencyof the software. Being able to measure the fault-pronenessof software can be a key segment towards directing the softwaretesting and improving the effectiveness of the entiredevelopment.Fault prediction is a difficultregion of study and the theme of severalprecedinglearning. Software practitioners and researchers have investigated various ways of predicting, where faults are likely to occur in software to varying scale of success. These studies typically generate fault prediction models that permit software engineers to focus development activities on fault-prone code, thereby improving software quality and making better use of resources.

There are numerousamount of software having faults is delivered to the market. Fault is a flaw that results in failure. One should have to know the clear difference between bug, fault, and failure. Failure is deviation of software actions from the expected outcomes. A fault in software is a flaw that results in failure. Bug occurs when specified requirements of the software do not conform. So, the main goal is to have software that contains less number of faults as much as possible. Machine learning technique is the best technique used for software fault prediction. Software fault prediction approaches are much more cost-effective to detect software faults compared to software reviews.The goal of this research was to identify and measure the occurrences of faults and the efficiency of their removal by development phase in order to target software development process improvement strategies. Through our analysis of the system data, the study confirms that catching faults in the phase of origin is an important goal. The faults that migrated to future phases are on average 10 times more costly to repair. This research work proposes a new algorithm to estimate the fault-proneness in the design phase, using complexity metrics GQM(Goal-Question-Metric).

II. MACHINE LEARNINGMETHOD

“A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P improves with experience E”(Figure 1).

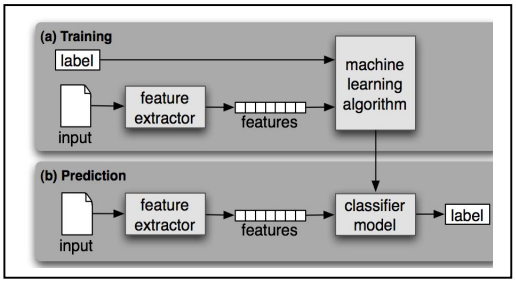

Fig. 1. Machine Learning.

International Journal of Business Intelligent
Volume: 04 Issue: 02 December 2015,Pages No.85- 90
ISSN: 2278-2400

Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of computer programs that can teach themselves to grow and change when exposed to new data.

A. Formation of Machine Learning

The set of items over which the concept is defined is called the set of instances, which is denoted by X. The concept or function to be learned is called the target concept, denoted by 'f'. In general, 'f' can be any Boolean valued function defined over the instances X; that is, f: X →{0, 1}.

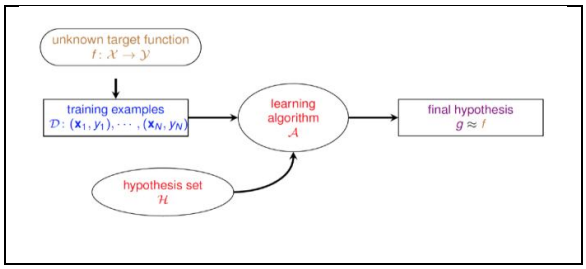

Fig. 2. Machine Learning Formation.

The illustration of the Machine Learning formation is shown in Figure 2. For a set of possible instances X (<x_1, x_2 … x_n>) an unknown target function 'f' is formed for each instances of X such that $X \rightarrow Y$ where Y is discrete value. All the target functions are grouped to make the hypothesis set H. The goal (g) is to find the output 'y' that is a hypothesis g such that g(x) = f(x) for all x in X. Usually H is determined by the human designer's choice of hypothesis representation.

B. Machine Learning Techniques

The various machine learning techniques are used to predict the value from the set of complex data set by forming the hypothesis space H. The important techniques are defined below.

Decision Tree: Decision Trees are great and standard tools for classification and prediction. It produces classifiers in the form of structure of a tree where each leaf node represents decision node. In this classification starts from root of the tree and continues to move down until leaf node is reached. Classification helps in classifying faulty and non-faulty modules. Prediction helps in predicting faulty and non-faulty modules. Decision tress helps in developing fault prediction models that predicts faults to improve quality.

Genetic Algorithm:A geneticalgorithm (GA) is a search technique used in computing to find exact or approximate solutions to optimization and search problems. Genetic algorithms are categorized as global search heuristics and a particular class of Evolutionary Algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover. With help of Genetic algorithm classification of the software components into faulty/fault-free systems is performed.

Neural Network: Neural Network helps in recognizing patterns from the data set. An artificial neural network is composed of many artificial neurons that are interconnected together according to specific network architecture. The goal of the neural network is to transform the inputs into meaningful outputs. Adaptive Resonance Neural Network is generally used or defect prediction in software systems. It helps in identifying faulty modules very excellently. The benefit of using this technique is that it assists in decreasing effort and cost of developing software.

Naive Bayes: It is a classifier based on Bayes theorem used in software fault prediction. It resolves the several difficulties like spam classification (to predict whether email is spam or not), medical diagnosis (given list of symptoms, predict whether patient has cancer or not) and so on. This method can be used to predict faulty and non-faulty modules.

Logistic Regression:Logistic Regression is a statistical classification model. It is used to predict a binary response from a binary predictor. It is used for predicting the outcome of class label based on one or more predictor variables. Logistic regression measures the relationship between a dependent variable and one or more independent variables, which are usually continuous. Logistic Regression technique uses class level metrics for software fault prediction. This technique is based upon the statistical based approach.

Bagging method: It creates base learners on many data subsets that are uniformly sampled from the original data, and then uses a linear combination to aggregate them. It is also referred as Bootstrap Aggregating. It also helps in identifying faulty and non-faulty modules with data sets that suffers from imbalance problem. This method can increase the performance.

III. RELATED WORK

The literature surveys of machine learning in software engineering studies of the defect data predictions are mapping in the Table. 1.

Table 1 - Survey of Machine learning techniques in software engineering fault prediction

S.No.	Authors/Year	Methodology Name	Result
1	Pandiyan G. , P. Krishnakumari [Jan-2015][1]	Tabu Search Fault Localization with Path Branch and Bound procedure on Software Engineering (TSFLPBB)	1. Improvesthe prediction rate. 2. Identifiesthe errors in the software program. 3. Greatlylocalizes the faults
2	DeepinderKaur [Nov -2014][2]	K-means clusteringwith three different distancemeasures Euclidean, Sorensen and	1. Results are displayed with the help of ROC curve. 2. K-means clustering with Sorensen

International Journal of Business Intelligent
Volume: 04 Issue: 02 December 2015,Pages No.85- 90
ISSN: 2278-2400

		Canberra	distance is better than Euclidean distance and Canberra distance. 3. Beneficialfor improving software reliability and achieve high quality software fault predictors.
3	Er. RohitMahajan, Dr. Sunil Kumar Gupta, Rajeev Kumar Bedi [jun 2014][4]	Various machine learning techniques	1. Machine learning models have better features than statistical methods or expert opinion. 2. The result describes that machine learning models are mostly used and these models are used to increase the usage of public datasets for fault prediction in future.
4	AditiPuri, HarshpreetSingh [Jan 2014][3]	Genetic Algorithm technique	1. Genetic Algorithm is used to find critical classes and metrics that are fault prone. 2. The proposed Genetic Algorithm technique shows high low value of Probability 3. The error and accuracy values are calculated and recorded as 0.294 and 0.705 respectively.
5	Gurvinder Singh, Baljit Singh Saini&Neeraj Mohan [2013][5]	Various machine learning techniques	1. Software Fault prediction models have the potential to improve the quality of software systems and reduce the expenditure related with delivering those systems 2. Techniques have shown better results than other algorithms with lower values and accuracy is implemented.
6	A.A. ShahrjoojHaghighi [2012]	Classification algorithms - Bayes	Proposed a fault detection system with higher performance, which decreases the cost of software fault detection.
7	L.Madeyski and M.Jureczko [2011]	Spam filtering technique	1. Each module considered asan e-mail message 2. New modules wereclassified as FP and NFP onbasis of past history 3. FPFinder tool was used totrack bugs.
8	Parvinder S. Sandhu, Sunil Khullar, Satpreet Singh, Simranjit K. Bains, ManpreetKaur, Gurvinder Singh [2010]	Genetic Algorithm technique	1. Predict the knowing fault prone data atearly stages of lifecycle combined with data available during code 2. Help the project managers to build the projects with more accuracy 3. It will reduce the testing efforts as faultyareas are already predicted
9	Yue Jiang [2009]	Various machine learning techniques	1. Explored five questions regarding fault prediction model 2. Author found that large data set is

International Journal of Business Intelligent
Volume: 04 Issue: 02 December 2015,Pages No.85- 90
ISSN: 2278-2400

			more informative than small data set. Results confirmed that variance is a very significant feature in accepting fault prediction models
10	SaiqaAleem , Luiz Fernando Capretz , Faheem Ahmed	Different machine learning techniques is explored for software bug prediction	1. Multiple techniques can be combined in order to get more accurate results. 2. Most of the machine learning methods performed well on software bug datasets.

The above review of various fault predictions analysis shows that machine learning technique participate an imperative function in finding and predicting the faults in software. This review also shows that there is no combination metrics used in Genetic Algorithm for fault prediction during the design phases.

IV.METRICS IN MACHINE LEARNING

A. Task

A quantitative measure of the degree to which a system, component, or process possesses a given attribute (IEEE Standard Glossary of Software Engineering Terms).Software metrics are numerical data related to software development. Metrics strongly support software project management activities. They relate to the four tasks of management as follows:

Planning - Metrics serve as a basis of cost estimating, training planning, resource planning, scheduling, and budgeting.

Organizing - Size and schedule metrics influence a project's organization.

Controlling - Metrics are used to status and track software development activities for compliance to plans.

Improving - Metrics are used as a tool for process improvement and to identify where improvement efforts should be concentrated and measure the effects of process improvement efforts.The employed approach for metric used in the machine learning is GQM. In GQM, the goals are gradually refined into several questions and each question is then refined into metrics. Also, one metric can be used to answer multiple questions. In this research workthe metrics are defined by using the Software Quality attributes.

B. Goal-Question-Metrics Approach(GQM)

GQM presents a systematic approach for integrating goals to models of the software processes, products and quality perspectives of interest based upon the specific needs of the project and the organization. (Basili et al, 1994).In other words, this means that in order to improve process one have to define measurement goals, which will be, after applying GQM method, refined into questions and consecutively into metrics which will supply all the necessary information for answering those questions.The GQM method provides a measurement plan that deals with the particular set of problems and the set of rules for obtained data interpretation. The interpretation gives the answer if the project goals were attained.

The GQM approach provides a framework involving *three* steps:

Goal: The major goals of the development project.

Question: It is derived from goals that must be answered in order to determine if the goals are achieved.

Metrics: Measurements that provide the most appropriate information for answering the questions.

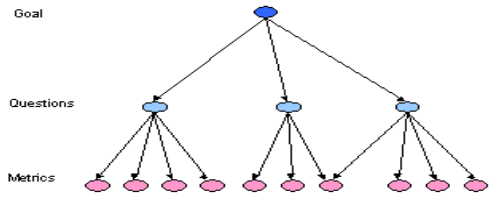

Fig. 3.GQM Approach Framework.

C. *Metrics of Software Quality*

The various measures of Software Quality are

Correctness – degree to which a program operates according to specification

Defects/KLOC

Defect is a verified lack of conformance to requirements

Failures/hours of operation

Maintainability – degree to which a program is open to change

Mean time to change

Change request to new version (Analyze, design etc)

Cost to correctIntegrity - degree to which a program is resistant to outside attackFault tolerance, security & threats

Usability – easiness to use

Training time, skill level necessary to use, Increase inproductivity,subjective questionnaire or controlled experiment.

D. Nature of Defect

International Journal of Business Intelligent
Volume: 04 Issue: 02 December 2015,Pages No.85- 90
ISSN: 2278-2400

Project Defects are classified as Critical, Major, Minor,Cosmetic and Suggestions.

Critical-If the stated requirement is not fulfilled and is identified during testing then the Bug is said to be a Critical Fault.

Major - Though the requirement is met and there is a bug, which has a high impact, then the bug is called a Major Fault.

Minor - Though the requirement is met and there is a bug that has a low impact, then the bug is said to be a Minor Fault.

Cosmetic-Cosmetic defects are the bug which does not affect further processing of the application and testing.

Suggestions - Anything apart from the above is classified as Suggestions. This is for the betterment of the system.

Some of the GQM formulas for software defects are listed in the Table. 2.

Table 2 - GQM formulas for Defect(Fault)Discovery

S. No.	Defect	Formula
1	Defect distribution across phases (%)	Defect Distribution for phase % = (No. of defects per phase) / (Total defects) * 100
2	Defect distribution by severity (%)	Defect Distribution by Severity = (No. of defects for each severity type) / (Total defects) * 100
3	Defect distribution by Cause (%)	Defect Distribution by Cause = (No. of defects in each Cause) / (Total defects) * 100

V.GENETIC METRIC FAULT PREDICTION ALGORITHM (GMFPA)

The following algorithm is used to find the fault in the design phase by forming the hypothesis set H(S) by using the metric GQM. The highest value frommetric is assigned as the fault gain prediction.

A. Algorithm

Genetic Metric Fault Prediction Algorithm

***Input*:** Fault in design with priority

***Output*:** Gain value G(S,A)

Procedure Gain(S,F(A),H(S),i)

S – Set of faults (S_1, S_2…S_n)

QM: Quality measure attribute in design phase.

F(A): subset where X=A that is faults predicted

H(S): Hypothesis of fault set S.

A: Possible defects prediction values.

i :training set with 1 to n.

Gain(S, A) :fault distribution

begin

A←0

Initialize F(A)

Evaluate H(S)

i=1

while(i<n) do

begin

//Find F(A) for the design phase .

while(QM) do

if (fault) then

Form F(A)=(F(A)/H(S))

//alter H(S)

H(S)=H(S)+F(A)

Increment A by 1.

Increment i by 1.

Else

Increment i by 1.

end

end

Evaluate

$Gain(S, A)= H(S) - \sum (|F(A)|/|S|) * H(F(A))$

end.

end Gain.

B. Logic

Choose attribute A and K distinct values, divides the training set E into subsets (E_1,E_2,….,E_k). The expected entropy remains after trying attribute A (with branches i=1,2,…k). The relationship between hypotheses can bedefined more precisely as follows.

$$Gain(S, A)= H(S) - \sum_{V \in values(A)} (|S_V|/|S|) * H(S_V)$$

where,

V – Possible defects prediction values of A.

S – Set of faults.

S_V– subset where X_A = V that is the identified fault.

The above GMFPA Algorithm is used to find the Gain (S,A) so that to find the fault occurred in the software design phase to improve the quality of the software and also save the software from the failure. According to the algorithm the set F(A) is calculated by GQM metric formula. At the beginning, the A is initialized by 0. The H(S) is a two tailed test set is formed by adding F(A) to H. The value of A and i is incremented by 1 and the process continues till the n design quality attribute. The set of predicted is collected in H(S). From the H(S) the Gain (S,A) is calculated by highest value and that is the fault has assigned with highest priority to change.

International Journal of Business Intelligent
Volume: 04 Issue: 02 December 2015,Pages No.85- 90
ISSN: 2278-2400

C. Case study

To form H(S) a table is designedby showing the various faults captured. From the set of S evaluate the Gain(S, A) to find the fault. The analysis originated by grouping the problematic reportsby severity. In Table 3 the sample data report of design phase is shown. In this the majority of the reports curtailed from faultsattributed to highest priority faultis critical. This accounts that critical faults are to be solved to improve the quality[14].

Table 3 - Sample table of severityof problem reports in design phase [14]

Nature of design Fault	Priority value
Critical	**Highest**
Major	Medium
Minor	Very less
Cosmetic	Less
Unknown	None

Hypothesis set H(S) = {Critical, Major, Minor, Cosmetic, Unknown}

A=high priority=Critical.

Gain (S,A) = Highest Critical faults.

According to this study the critical faults are needed to be solved first to improve quality and reduce cost.

VI. CONCLUSIONS

In this research work, various techniques of Machine Learning to predict faults in software systems (like decision tree, neural network, naive Bayes and so on) were reviewed.The main aim is to survey the performance of different techniques in software fault prediction. This research work has proposed a new algorithm named GMFPA for finding the highest priority fault value to resolve quickly to improve quality. The fault prediction in software design is significant because it can help in guiding test effort, decreasing cost, and increasing quality of software and its reliability. The future work will predict the bug by forming model and the algorithm GMFPA implementation in real world data set for a particular problem.

REFERENCES

[1] Pandiyan G. and P. Krishnakumari, "An Efficient Software Defect Prediction Model Using Optimized Tabu Search Branch and Bound Procedure", ARPN Journal of Engineering and Applied Sciences ISSN 1819-6608, JANUARY 2015.

[2] DeepinderKaur , "A Comparative Study of Various Distance Measures for Software fault prediction" , International Journal of Computer Trends and Technology (IJCTT) – volume 17 Issue 3, November 2014.

[3] Er. RohitMahajan, Dr. Sunil Kumar Gupta, Rajeev Kumar Bedi, "Comparison of Various Approaches of Software Fault Prediction: A Review," International Journal of Advanced Technology & Engineering Research (IJATER), July 2014.

[4] AditiSanyal, Balraj Singh, "A Systematic Literature Survey on Various Techniques for Software Fault Prediction", International Journal of Advanced Research in Computer Science and Software Engineering, January 2014.

[5] Gurvinder Singh, Baljit Singh Saini, and Neeraj Mohan, "A Systematic Literature Review on software Fault Prediction based on Qualitative and Quantative Factors," ISSN (Print): 2319-2526, Volume-2, Issue 2, 2013.

[6] Gurvinder Singh, Baljit Singh Saini, and Neeraj Mohan, "A Systematic Literature Review on software Fault Prediction based on Qualitative and Quantative Factors," ISSN (Print): 2319-2526, Volume-2, Issue 2, 2013.

[7] Shahrjooj A. A., Haghighi, M. A., "Appling Mining Schemes to Software Fault Prediction: A Proposed Approach Aimed at Test Cost Reduction" ,Proceedings of the World Congress on Engineering, 2012.

[8] Madeyski and Jureczko.M "Which process metrics improve software defect prediction models? An empirical study," submitted to Information and Software Technology, 2011.

[9] Parvinder S. Sandhu, Sunil Khullar, Satpreet Singh, Simranjit K. Bains,"ManpreetKaur, GurvinderSinghA Study on Early Prediction of Fault Proneness in Software Modules using Genetic Algorithm",International Scholarly and Scientific Research & Innovation 4(12), 2010.

[10] SaiqaAleem,Luiz Fernando Capretz and FaheemAhmed ,"Comparative Performance Analysis Of Machine Learning Techniques For Software Bug Detection",Kamloops, British Columbia, Canada, V2C6N6

[11] Ritika Sharma. N. B., "Study of Predicting Fault Prone Software Modules," International Journal of Advanced Research in Computer Science and Software Engineering, 2012.

[12] Salfner. F., "Event-based Failure Prediction: An Extended Hidden Markov Model Approach", Dissertation. de—Verlag in Internet GmbH, Berlin, Germany, 2008.

[13] Salfner. F and Malek. M., "Using hidden semi-Markov models for effective online failure prediction",In Proceedings of the IEEE 26th International Symposium on Reliable Distributed Systems (SRDS). 2007.

[14] Dolores Zage Wayne Zage., "An Analysis of the Fault Correction Process in a Large-Scale SDL ProductionModel", Software Engineering Research Center Ball State University Muncie, in 47306 USA