International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:12, No:1, 2018

Hybrid Approach for Software Defect Prediction Using Machine Learning with Optimization Technique

C. Manjula, Lilly Florence

***Abstract*—**Software technology is developing rapidly which leads to the growth of various industries. Now-a-days, software-based applications have been adopted widely for business purposes. For any software industry, development of reliable software is becoming a challenging task because a faulty software module may be harmful for the growth of industry and business. Hence there is a need to develop techniques which can be used for early prediction of software defects. Due to complexities in manual prediction, automated software defect prediction techniques have been introduced. These techniques are based on the pattern learning from the previous software versions and finding the defects in the current version. These techniques have attracted researchers due to their significant impact on industrial growth by identifying the bugs in software. Based on this, several researches have been carried out but achieving desirable defect prediction performance is still a challenging task. To address this issue, here we present a machine learning based hybrid technique for software defect prediction. First of all, Genetic Algorithm (GA) is presented where an improved fitness function is used for better optimization of features in data sets. Later, these features are processed through Decision Tree (DT) classification model. Finally, an experimental study is presented where results from the proposed GA-DT based hybrid approach is compared with those from the DT classification technique. The results show that the proposed hybrid approach achieves better classification accuracy.

***Keywords*—**Decision tree, genetic algorithm, machine learning, software defect prediction.

I. Introduction

RECENTLY, technology is growing rapidly resulting in improvement of product quality in industrial applications. This technological growth has also been noticed in software-based applications. The use of software-based applications is ever increasing in daily routine and business life. Industrial growth depends on the quality of software hence developing a high-quality software is demanded to achieve the desired goals of industrial growth [1]. A minor defect in the software may lead to development of degraded module resulting in loss to the company. However, software testing sections evaluate the quality of software using manual testing. Manual testing becomes complex and require more human effort for software testing purpose. To overcome this issue, automated software testing is required which can be used for software defect prediction prior to complete development of software module.

Software defect prediction is a process to identify the bug in current software code with the help of machine learning or regression techniques [2]. According to this technique, previous software release information features are extracted and used to identify the bugs in newer version. With the help of this process, faulty section only will be evaluated rather than processing the entire code resulting in good quality software development with less human effect and cost. Hence, software defect prediction is widely adopted in various studies to overcome the software defect issues [3]. Conventional studies are mainly utilized following aspects of software defect prediction as code metrics computation, relationship between software defects and impact of software process on the software defectiveness [4]. However, several studies have been presented by using both process metrics and code matrices. Some studies concluded that code matrices are more significant and useful for defect prediction when compared with the code metrics performance [4], [5].

Other than this process, machine learning and data-mining techniques also have been utilized for predicting bugs in the software module [6]. In this field of machine learning process, classification is considered as most important stage which includes the identification of software bugs in terms of fault or no-fault categories by learning the previous instances. Several classification models are present for bug classification which are based on statistical classification [7], tree-based classifier [8], [9], neural-network [10] and analogy-based classification [11] schemes. Recently, Felix et al. [12] presented a hybrid approach for software defect prediction by introducing predictor variables which includes defect density, defect velocity and correlation of each predictor variable. Similarly, based on machine learning, Cheng et al. [13] presented semi-supervised learning scheme for software defect prediction. Authors discussed that generally software defect data are not labeled properly, and class-imbalance problem also exists which may degrade the performance of bug prediction. This issue is addressed by developing optimized classification scheme.

Software defect prediction for huge software systems is a challenging task. Lee et al. [14] developed a new approach based on the development pattern analysis. Huge amount of

Manjula.C. is Associate Professor, MCA Department, PES Institute of Technology Bangalore South Campus, Karnataka, India (e-mail: manjulaprasad@pes.edu)
Lilly Florence is Professor, MCA Department, Adiyamman College of Engineering, Tamil Nadu, India.

work has been carried out in this field of software defect prediction using machine learning and data-mining techniques but due to rapidly growing technology, probability of bugs also increasing hence there is a need to develop an automated system for early prediction of software bugs.

II. Issues and Challenges

Considerable amount of research has been reported [2], [4], [12] on early prediction of software defect but still several issues present in this field. This section presents brief information about issues and challenges in the software defect prediction (SDP) field.

A. Attribute and Fault Relationship

Due to improper attribute selection, it becomes a challenging task for researchers to identify the module whether it is faulty or non-faulty. Moreover, implementation of metrics as code metrics, requirement metrics and design metrics becomes a confusing task for proper analysis.

B. Standard Parameters for Performance Measurement

In this area, selection of performance measurement parameters is inconsistent hence there are no standard criteria for comparing the performance of defect prediction models.

C. Issues with Cross-Project Defect Prediction

Generally, in machine learning techniques, the learning process is carried out by using locally available data and it is also very similar to the testing dataset. For real-time application scenario of SDP, the testing data can be obtained from other projects with the same programming language. In this stage, companies face problem due to possible inconsistency in the testing data when compared with the training dataset. This issue is addressed using cross-project SDP model where training and testing both are done using different databases. However, this technique suffers from lack of accuracy.

D. Lack of General Framework

This is a very vast field of research and researchers have introduced various techniques which are carried out using different databases and software [7]. Hence, for each technique or SDP data, the working process may differ. For a robust application, there is no general framework available which can be used for any SDP data.

E. Class-Imbalance Problem

Performance of SDP model depends on the distribution of data class and training. Class distribution is known as the labeling of the class available training dataset. If the number of assigned class and available class does not match, then this problem is known as class-imbalance problem which is responsible for training error leading towards performance degradation.

From the discussion presented in this section it is clear that still there are various issues which need to be resolved to improve the classification performance for software defect prediction. To overcome this issue, we present a hybrid approach for SDP where optimal feature selection and DT classification schemes are incorporated. The rest of the article is organized as follows: Section III provides description of proposed model, Section IV provides detailed experimental study and Section V gives concluding remarks regarding proposed technique.

III. Proposed Model

Previous section presents a brief introduction about SDP, existing techniques and challenges for developing a robust SDP model. This section focuses on the proposed strategy. The complete process is divided into two phases: (a) feature selection (b) DT classification. Overall system architecture is depicted in Fig. 1.

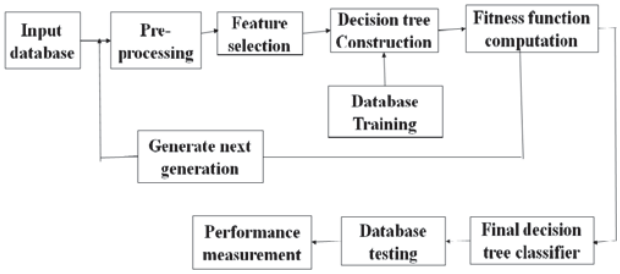

Fig. 1 Overall System Architecture

A. GA for Feature Selection

Here we present, GA modeling approach which is used as a selection and combination scheme. This technique tends to perform optimal feature selection by applying certain computations for the given problem. Initially, problem is evaluated, and a best solution is obtained from this stage to build the next stage. This process is repeated for several iterations until the desired solution is obtained. In the literature [6] it is shown that GA has a significant influence but when applied to complex problem and huge dataset it suffers from premature convergence problem of local optima and requires more time for computation. In order to overcome this issue, we present a new architecture for GA where chromosome and fitness function are modified to achieve the improved optimization performance.

1) Chromosome Design

Chromosome design is an important task in GA hence first of all, we preset GA chromosome modeling using Gaussian kernel function.

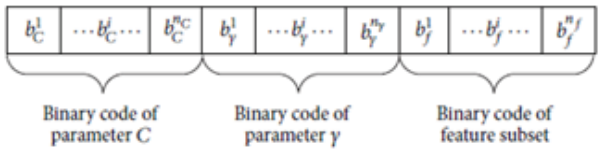

Fig. 2 Chromosome Design Structure

According to this process, $C_T^1 \sim C_T^{n_t}$ denotes the binary code for the parameter T which is known as tuning parameter. Similarly, $C_\gamma^1 \sim C_\gamma^{n_\gamma}$ denotes the binary code of γ known as

Gaussian kernel parameter, n_T denotes the total number bits used to represent the parameter T, n_γ denotes the total number of bits required for γ. With the help of the binary code, genotype can be expressed as:

$$p = \underset{p}{min} + \frac{d}{(2^n - 1)}(max_p - min_p) \quad (1)$$

where$p$ denotes the phenotype, $max_p\ and\ min_p$ denote the maximum and minimum parameters of the genotype model. With the help of this model, we can obtain the minimum and maximum limit of the function which helps to decide the local optimal minima for better convergence.

2) Fitness Function Computation

Fitness function is an important part of GA which is used for identifying the best fit population from the generated solution. This solution is further processed for classification analysis. Better fitness value will lead to the better classification performance. The fitness function can be computed as:

$$\mathcal{F} = W_F f + W_V v + W_A \times Accuracy \quad (2)$$

where $f = 1 - \frac{(\sum_{i=1}^{n_f} F_i)}{n_f}$ and $v = 1 - \frac{(\sum_{i=1}^{l} V_i)}{l}$, W_A denotes classification accuracy weight, W_F denotes the weight of feature score and f is score of selected feature subset , W_V denotes weight of feature set and V is feature set vector.

B. GA Implementation

With the help of (1) and (2), we try to improve the performance of GA. There are various stages present in the GA which are as follows:

(1) Input Dataset. First of all, we provide all input data obtained from the SDP repository. These databases are further divided into training and testing for performance analysis.

(2) Data pre-processing: In next stage, we apply data pre-processing which is used for discarding the huge variations in the input dataset resulting in organizing the dataset in a range. According to this process, each feature value is linearly scaled as:

$$r' = \frac{r - min}{max - min} \quad (3)$$

$r$denotes the original value and r' denotes the scaled pre-processed output value.

(3) Population Initialization: Initially a random population is generated based on the input features which are further used for best feature selection in terms of population.

(4) Genetic Operations: Once the population is generated and the process is initiated then we apply genetic operations such as selection, mutation, crossover for generating next solutions.

(5) Evaluate the Parameters: After generation of parameters, each parameter is evaluated and if achieves the best fit criteria then it is considered as next population.

(6) Termination: Once each process is completed and desired criteria are achieved or maximum number of iterations are completed then the GA process is terminated and the final output is considered as most optimal feature sub-set from the given input.

C. DT Classification Model

Previous section discussed about GA process for best feature selection. In next phase, we study about classification and prediction of software defects using DT classifier model.

DT classification is a technique which performs recursive partition on the given input sub-space based on its attribute values. In this process, data are divided into various nodes and these nodes are further divided into two or more sub-spaces known as leaf based on attribute value. Here each leaf is assigned to one class and instances are identified by traversing the constructed tree root to the leaf. In this work, we used ID3 based DT classification scheme due to its simple nature which follows to-down computation along with the greedy search algorithm. According to this process, any attribute which is having best split, is assigned as current node. This process is repeated until one of the following conditions is achieved: Each attribute is considered in the current path and current node has all target values. A pseudo code is also presented in Fig. 3. Here S denotes a training set, input features are denoted by F, target feature denoted by c and splitiing criterion is denoted SC.

Input: DT classifier (S, F, SC, c)
Output: constructed decision tree
Step 1: formulate a tree T with single root node.
Step 2: if further splits are not possible then
Step 3: consider T as leaf and label c
Step 4: else
Step 5: $\forall f_i \in F$ find f which has the best split criterion $SC(f_i, S)$
Step 6: label this f with t
Step 7: for each value v_i o f
Step 8: construct each sub-tree
Step 9: connect each label with edge
Step 10 end

Fig. 3 Tree construction using DT

Input: training set, samples and attributes
Output: weight vector as classified output for each instance
Step 1: Set initial weight to $W[1.K] = 0$.
Step 2: for $i = 1: N$
Step 3: select random set from the attributes
Step 4: compute nearest matching hit
Step 5: compute nearest miss from the attribute set.
Step 6: for: 1: total attributes
Step 7: $W[A] - \frac{d(A,R,H)}{N} + \frac{d(A,R,M)}{N}$
Step 8: end
Step 9: return W

Fig. 4 Weight vector

As discussed before, splitting criteria can be obtained by computing information gain which can be defined as:

$$G(S, A) = E(S) - \sum_{v \in V(A)} \frac{|S_{A=v}|}{|S|} E(S_{A=v}) \quad (4)$$

where $E(S)$ denotes the entropy index for the given dataset.

In next phase, we compute the difference based on the considered input attribute instances. This can be expressed as:

$$d(A, I_1, I_2) = \begin{cases} 0 & I_1[A] = I_2[A] \\ 1 & otherwise \end{cases} \quad (5)$$

Finally, weight vector of each attribute can be computed as given in Fig. 4. This complete process is used for SDP which provides an output vector for given input set in terms of defective or non-defective instance.

IV. Experimental Study

The proposed hybrid GA-DT approach is tested for open source software defect dataset. Proposed approach is implemented using MATLAB tool for PROMISE dataset [15].

A. Database Description

A brief description of dataset information is presented in Table I.

TABLE I
DATASET DETAILS

Dataset Name	Dataset details				
	Language	*Details*	*Modules*	*Non-Defective/ Defective*	*% defect*
PC3	C	Flight software	1563	1403/160	10.23
PC4	C	Flight software	1458	1280/178	12.20
KC3	Java	Satellite data processing	458	415/43	9.38

Table I shows basic information about the input dataset including total percentage of defect present in the database. We have conducted two experiments on these datasets. First experiment is carried out with the help of DT classification where other external optimization schemes are not incorporated whereas second experiment is a combination of GA and DT classifier. Finally, we present a comparative study between these two experiments in terms of classification accuracy performance and other statistical parameters.

B. Performance Measurements

This sub-section provides brief details about performance measurement parameters used in this work. In this study, first of all we analyze classification confusion matrix. This matrix is represented in Table II.

TABLE II
CONFUSION MATRIX STRUCTURE

	Defective predicted	Defect free predicted
Observe defective	True Positive	False negative
Defect free	False positive	True negative

Based on the parameters as given in Table II, accuracy also can be computed as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (6)$$

Similarly, sensitivity and specificity also can be computed as given in (7) and (8):

$$Sensitivity = \frac{TP}{TP + FN} \quad (7)$$

and

$$Specificity = \frac{TN}{FP + TN} \quad (8)$$

C. Experimental Study of Test Case 1 for PC3

This sub-section deals with the experimental analysis of test case 1 where optimization scheme is not included for SDP. Performance of this approach is obtained as presented in Table III.

TABLE III
PERFORMANCE MEASUREMENT FOR PC3 WITHOUT OPTIMIZATION

Parameter Name	Accuracy (%0	Sensitivity	Specificity
Obtained value	89.26	0.235	0.967

D. Experimental Study of Test Case 2 for PC4

In this section, PC4 dataset is considered for experimental study where total 1458 records are present in the module. Performance of this experiment is given in Table IV.

TABLE IV
PERFORMANCE MEASUREMENT FOR PC4 WITHOUT OPTIMIZATION

Parameter Name	Accuracy (%)	Sensitivity	Specificity
Obtained value	88.64	0.426	0.949

E. Experimental Study of Test Case 3 for KC3

Finally, we present experimental study without optimization for KC3 database which contains less number of modules as 458 with 9.38% defective attributes. Performance obtained is given in Table V.

TABLE V
PERFORMANCE MEASUREMENT FOR KC3WITHOUT OPTIMIZATION

Parameter Name	Accuracy (%)	Sensitivity	Specificity
Obtained value	85.31	0.167	0.916

F. Proposed Experimental Study of Test Case 4 for PC3

Here we present, experimental study using DT classification where GA is also incorporated. Table IV shows performance for PC3 database. Similarly, we have conducted experiments for each database given in Table I and evaluated their performance. Tables VI, VII and VIII show performance analysis of PC3, PC4 and KC3 dataset using proposed approach.

TABLE VI
PERFORMANCE MEASUREMENT FOR PC3 WITH OPTIMIZATION

Parameter Name	Accuracy (%0	Sensitivity	Specificity
Obtained value	91.68	0.45	0.97

TABLE VII
PERFORMANCE MEASUREMENT FOR PC4WITH OPTIMIZATION

Parameter Name	Accuracy (%)	Sensitivity	Specificity
Obtained value	92.09	0.593	0.963

TABLE VIII
PERFORMANCE MEASUREMENT FOR KC3WITH OPTIMIZATION

Parameter Name	Accuracy (%)	Sensitivity	Specificity

Obtained value	93.36	0.623	0.976

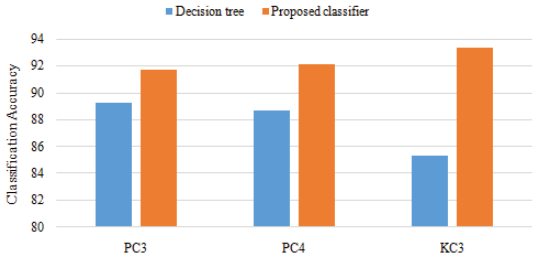

Fig. 5 Classification Accuracy Performance

Fig. 5 shows classification accuracy performance comparison for considered data base. Statistical performance comparison is depicted in Fig. 6 where specificity and sensitivity parameters are computed and compared.

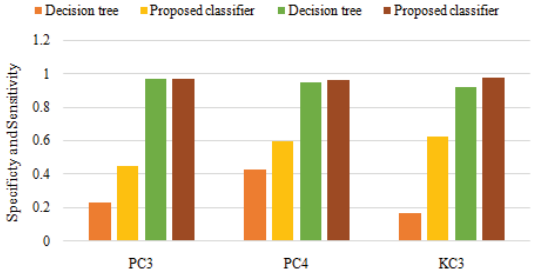

Fig. 6 Specificity and Sensitivity Performance

Complete study shows a significant improvement in classification performance which can be beneficial for SDP applications.

V. Conclusion

This work mainly focused on the SDP analysis using machine learning techniques. Several techniques have been developed to achieve the objective of early defect prediction in software applications but due to some certain limitations, classification accuracy of bug prediction still remains a challenging task. To overcome this issue, we present a combined scheme of feature optimization and classification using GA with DT classification. An extensive experimental study is presented for PROMISE SDP dataset repository. Experimental study shows that proposed approach achieves better performance when compared with existing models.

References

[1] Grbac, TihanaGalinac, Per Runeson, and DarkoHuljenić. "A quantitative analysis of the unit verification perspective on fault distributions in complex software systems: an operational replication." Software quality journal 24, no. 4 (2016): 967-995.

[2] P. Bishnu and V. Bhattacherjee, "Software fault prediction using quad tree based k-means clustering algorithm," IEEE Transactions on Knowledge and Data Engineering, vol. 24, no. 6, pp. 1146–1150, 2012.

[3] Malhotra, Ruchika. "A systematic review of machine learning techniques for software fault prediction." Applied Soft Computing 27 (2015): 504-518.

[4] Tantithamthavorn, Chakkrit, Shane McIntosh, Ahmed E. Hassan, and Kenichi Matsumoto. "An empirical comparison of model validation techniques for defect prediction models." IEEE Transactions on Software Engineering 43, no. 1 (2017): 1-18.

[5] Nam, Jaechang, Wei Fu, Sunghun Kim, Tim Menzies, and Lin Tan. "Heterogeneous defect prediction." IEEE Transactions on Software Engineering (2017).

[6] Maua, Goran, and TihanaGalinacGrbac. "Co-evolutionary multi-population genetic programming for classification in software defect prediction." Applied Soft Computing 55, no. C (2017): 331-351.

[7] Seiffert, C., Khoshgoftaar, T.M., Van Hulse, J. and Folleco, A., 2014. An empirical study of the classification performance of learners on imbalanced and noisy software quality data. Information Sciences, 259, pp.571-595.

[8] L. Guo, Y. Ma, B. Cukic, and H. Singh, "Robust Prediction of Fault-Proneness by Random Forests," Proc. 15th Int'l Symp.Software Reliability Eng., 2004

[9] Rathore, S.S. and Kumar, S., 2017. A decision tree logic based recommendation system to select software fault prediction techniques. Computing, 99(3), pp.255-285.

[10] Arar, Ö.F. and Ayan, K., 2015. Software defect prediction using cost-sensitive neural network. Applied Soft Computing, 33, pp.263-277.

[11] Idri, A., AzzahraAmazal, F. and Abran, A., 2015. Analogy-based software development effort estimation: A systematic mapping and review. Information and Software Technology, 58, pp.206-230.

[12] E. A. Felix and S. P. Lee, "Integrated Approach to Software Defect Prediction," in IEEE Access, vol. 5, pp. 21524-21547, 2017.

[13] M. Cheng, G. Wu, M. Yuan and H. Wan, "Semi-supervised Software Defect Prediction Using Task-Driven Dictionary Learning," in Chinese Journal of Electronics, vol. 25, no. 6, pp. 1089-1096, 11 2016.

[14] T. Lee, J. Nam, D. Han, S. Kim and H. Peter In, "Developer Micro Interaction Metrics for Software Defect Prediction," in IEEE Transactions on Software Engineering, vol. 42, no. 11, pp. 1015-1035, Nov. 1 2016.

[15] Software Defect Dataset, Promise Repository, http://promise.site.uottawa.ca/SERepository/datasets-page.html.accessedaround14/11/2017.

Manjula.C. isfrom Bangalore, Karnataka India, has completed Bachelor of Science, Masterof Computer Applications and MPhil in Computer Science. Having 17 years of Teaching experience and 5 years of Industry experience. Currently working as an Associate Professor at PES Institute of Technology Bangalore South Campus, Bangalore, Karnataka, India. Has 5 publications in National/ International Journals. Has organized more than 10workshops and seminars.

M. Lilly Florence is from Hosur, Tamil Nadu has completed her Bachelor's Degree in mathematics and Master degree MCA, M.Tech.(IT)and Doctorate in Computer Science.She has 17 years of teaching experience. She is good in teaching all Programming Languages. Prof. Lilly Florence has published 20 research papers in National and International Journals, also she has published 24 Papers in various National and International Conferences. She is an author of three text books namely, Operating Systems, Computer Graphics and Multimedia and Computer Architecture and Organization. Prof. Lilly has organized more than 20 workshops, seminars for various groups of audience. She has visited more than 20 colleges as a Technical Resource Person. She has received grants from DRDO, DST, ISRO, etc to organize FDP and seminars. She is acting as a Computer Society of India Student Branch Counselor for Adhiyamaan College of Engineering .in Research, she is a recognized supervisor of Periyar University and Bharathiyar University. She has produced one Ph.D Scholar and currently she is guiding 6 Ph.D scholars. Dr. Lilly has undertaken 2 research projects funded by Department of Science and Technology for Rs. 23.00 lakhs. Prof. Lilly is a life member of Computer Society of India and ISTE. Also, she is a BOS member of MCA board.