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

Manjula C, Lilly Florence

Abstract—**Technology is growing rapidly which lead to the industrial growth for various manufacturing and software industries. Now-a-days, demand and use of software based application has attracted users and adopted widely in daily routine or business work purposes. For any software industry, development of a reliable software is becoming a challenging task because a faulty software module may be harmful for industry's growth. Hence there is a need to develop a technique which can be used for early prediction of software bugs. Due to complexities in manual testing, automated software defect prediction techniques have been introduced which are based on the pattern learning from the previous version and find the bugs in current software module. These techniques have attracted researchers due to its significant impact on industrial growth by identifying the bugs in software. Based on this several researches have been carried out but achieving desirable bug prediction performance is still a challenging task. To address this issue, here we present a machine learning based technique for software defect prediction. First of all, Genetic Algorithm is presented where an improved fitness function is presented for better optimization. Later, these features are processed through Decision Tree classification model. Finally, an experimental study is presented where proposed approach is compared with the state-of-art Decision Tree classification technique which shows that proposed Hybrid Approach Genetic Algorithm with Decision Tree 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 been noticed in software based applications also. 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.

Manjula.C, Associate Professor, MCA Department, PESIT- BSC, Karnataka. India, (phone: +91 9739401731, email: manjulaprasad@pes.edu)

Lilly Florence, Professor, MCA Department, Adiyamman College of Engineering, Tamil Nadu, India.

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 these 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 is 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 can be obtained using software metrics but for huge software performance remains a challenging task hence Lee et al. [14] developed a new approach based on the development pattern analysis. huge amount of 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 presented for early prediction of software defect but still several issues present in this field. This section presents a 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 performane measurement*

In this area, selection of performance measurement parameters is inconsistence hence there is 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, this testing data can be obtained from other project with the same programming language. In this stage, companies face problem due to insufficient knowledge about the testing data whereas it is compared with the training set. This issue is addressed using cross-project SDP model where training and testing both are done using different database. This technique suffers from the lower 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 database and software. Hence, for each technique or SDP data, the working process may differ. For a robust application, there is no general framework is available which can be used for any SDP data.

E. *Class-imbalance problem*

Performance of software defect prediction 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.

This section presents that still there are various issues present which need to be resolved to reduce the classification and improve the classification performance for software defect application.

To overcome this issue, we present a new hybrid approach for SDP where optimal feature selection and decision tree classification schemes are incorporated. 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 software defect prediction, existing techniques and challenges for developing a robust software defect prediction model. This section focus on the proposed strategy. The complete process is divided into two phases: (a) feature selection (b) decision tree classification. Overall system architecture is depicted in Fig 1.

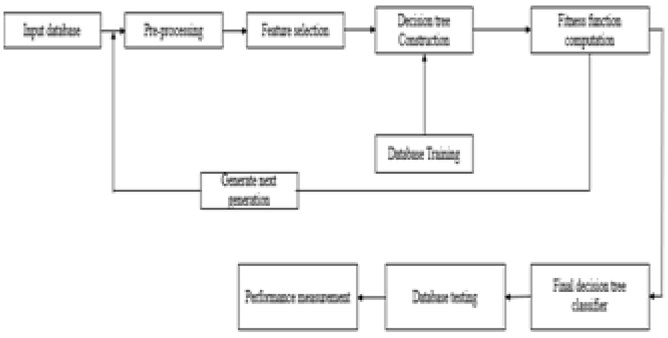

Fig 1.Overall System Architecture

A. *Genetic Algorithm for feature selection*

Here we present, genetic algorithm modeling which is which is used known as a search algorithm which depends on the selection and combination scheme. This technique tends to optimal feature selection by applying certain computations of 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. Several studies have been presented using genetic algorithm which show a significant contribution but if the problem is complex and applied for huge dataset then 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 genetic algorithm where chromosome and fitness function are modified to achieve the improved optimization performance.

1) *Chromosome design*

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

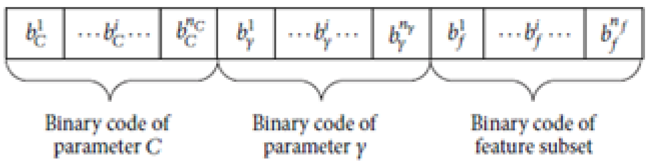

Fig. 2 Chromosome Design Structure

According to this process, $C_T^1 \sim$ denotes the binary code for the parameter which is known as tuning parameter. Similarly, $C_\gamma^1 \sim$ denotes the binary code of known as Gaussian kernel parameter, denotes the total number bits used to represent the parameter , denotes the total number of bits required for . With the help of the binary code, genotype can be expressed as:

$$p = \frac{min}{p} + \frac{d}{(2^n - 1)}\left(max_p - min_p\right) \qquad (1)$$

Where denotes the phenotype, max_p *and* m denotes 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 genetic algorithm 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:

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

Where $f = 1 - \frac{(\sum_{i=}^{n} f}{n}$ and $v = 1 - \frac{(\sum_{i=}^{l}}{l}$, denotes classification accuracy weight, denotes the weight of feature score and is score of selected feature subset.

B. *Genetic Algorithm implementation*

With the help of these two-function, we try to improve the performance of genetic algorithm. There are various stages present in the genetic algorithm which are as follows:

(1) *Input Dataset.* First of all, we provide all input data obtained from the software defect prediction 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:

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

denotes the original value and denotes the scaled pre-processed output value.

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

(4) *Genetic operations:* once the population is generating 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 considered as next population.

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

C. *Decision tree classification model*

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

Decision tree classification is a technique of classification which performs recursive partition on the given input sub-space based on its attribute values. In this process, data is 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 decision tree 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 are achieved: each attribute is considered in the current path and current node has all target values. A pseudo code is also presented in figure 3. Here denotes a training set , input features are denoted by , target feature denoted by and splitiing criterion is denoted .

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

Fig.3 Tree construction using DT (Decision Tree) Classification

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

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

Where E 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:

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

Finally, weight vector of each attribute can be computed as given in Fig. 4.

```
Input : training set,  samples and attributes
Output: weight vector as classified output for each instance
Step 1: Set initial weight to    $W[1.K]$ :.
Step 2: for    $i = 1$
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:    $W[A] - \frac{d(A,R,H)}{N} + \frac{d(A,}{N}$
Step 8: end
Step 9: return
```

Fig.4 Weight vector

This complete process is used for software defect prediction which provides an output vector for given input set in terms of defective or non-defective instance.

IV. Experiemtnal study

After discussing, genetic algorithm optimization and decision tree classification scheme. The proposed 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 the below given 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 decision tree classification where other external optimization schemes are not incorporated whereas second experiment is a combination of genetic algorithm and decision tree 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 analyzeclassification 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 these parameters as given in table 2, accuracy also can be computed as follows:

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

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

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

And

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

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 software defect prediction. performance of this approach is obtained as presented in Table III.

TABLE III. Performance measurement for PC3

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

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 for KC3 database which contains less number of modules as 458 with 9.38% defective attributes. Obtained performance is given in Table V.

TABLE V. PERFORMANCE MEASUREMENT FOR KC3

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

Similarly, we conducted other experimental study where we have considered genetic algorithm for feature optimization and decision tree classification technique is used for measuring the performance.

F. Proposed experimental study of test case 4 for PC3

Here we present, experimental study using, decision tree classification where genetic algorithm is also incorporated. Table IV shows performance for PC3 database.

TABLE VI. PERFORMANCE MEASUREMENT FOR PC3

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

Similarly, we have conducted experiments for each database given in table 1 and evaluated their performance. Table 7 and 8 shows performance analysis of PC4 and KC3 database using proposed approach.

TABLE VII. PERFORMANCE MEASUREMENT FOR PC4

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

Finally, KC3 performance is presented in Table VIII.

TABLE VIII. PERFORMANCE MEASUREMENT FOR PC4

Parameter Name	Accuracy (%)	Sensitivity	Specificity
Obtained value	93.36	0.623	0.976

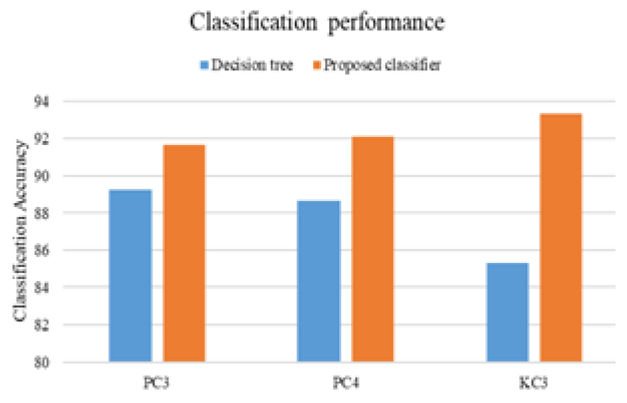

Fig. 5 Classification Accuracy Performance.

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

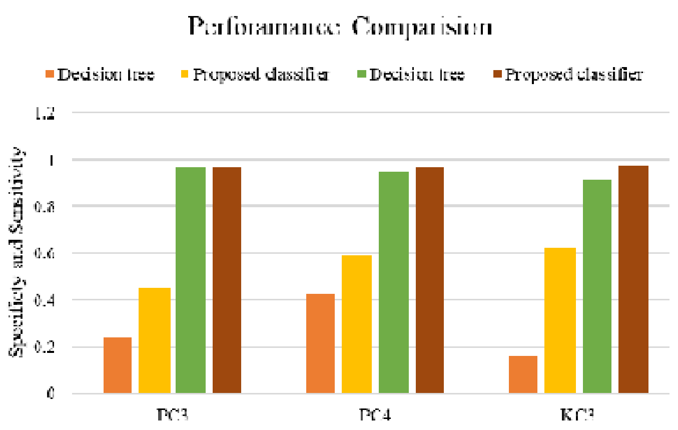

Fig. 6 Specificity and Sensitivity Performance

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

V. CONCLUSION

This work is mainly focused on the software defect prediction 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 genetic algorithm with decision tree 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, Tihana Galinac, Per Runeson, and Darko Huljenić. "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 learni ng 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 Tihana Galinac Grbac. "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., azzahra Amazal, 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.

Manjula.C is from Bangalore, India. Has done MCA, Mphil in Computer Science.

Having 17 years of Teaching experience and 5 years of Industry experience.

Has 5 publications.