A Novel Ensemble Feature Selection and Software Defect Detection Model on Promise Defect Datasets

E. Sreedevi, Y. Prasanth

Abstract*: The prediction of software defects is an essential step before building high quality software. Although much research has been done for analyzing the software metrics and feature extraction. Unfortunately, traditional models failed to predict the defects using the multiple software projects data. As the number of software projects and modules increases, the sparsity and uncertainty of the data increases, which affects the overall true positive rate of the defect prediction process. In this paper, a hybrid ensemble feature selection and defect prediction model was designed and implemented on the openscience software defect dataset. ReliefF, Chi-square and improved predictive correlation measures are used in our ensemble feature selection process. Experimental results show that proposed model has high defect detection rate, recall and F-measure compared to the traditional software defect prediction models.

***Index Terms*: Machine learning, defect detection, decision tree.**

I. INTRODUCTION

Software defect detection aims to detect the software defects of new type of softwares with the bug training data. It plays an essential role in optimizing the software quality in recent software frameworks. Unfortunately, it is still difficult to discover the new type of defects based on current models. In a defect classification model, new type of software programs will be categorized into non-defect and defect classes. In the existing works, software defects are classified using many popular models such as SVM, Naïve Bayes, decision tree and Bagging models. Most of the traditional defect prediction models are adopted using the ensemble classifiers with high classification error rate.

A defect of software is a bug, error, fault, failure that leads to a mistaken or unpredictable outcome. Faults are fundamental system features in software systems. They appear from design or production or external environment. Software defects are programming errors that perform differently than anticipated. Most faults are due to source design or implementation phases and some are due to the wrong compiler code. Software failures will not only reduce the quality of software, increase costs, but also delay the development plan. To solve this sort of problem, software fault forecasting is proposed. SDP can improve the efficiency and allocation of resources effectively by providing software testing. Software flaws must be recognized and corrected in the early stages of SDLC for the development of high-quality software.

Prediction of bug software is a key activity in the development of software. The prediction of defect modules is necessary to improve the software quality testing before software is deployed and also improving overall software performance. In addition, the early prediction of the software defect optimizes the adaptation of software to different platforms and thus increases resource use.

Several techniques for addressing the Software Bug Prediction (SBP) problem have been suggested in the past few years. Machine Learning (ML) techniques are most commonly used. In SBP, the ML techniques are widely used to forecast defect data and to predict new type of software defect using the training data.

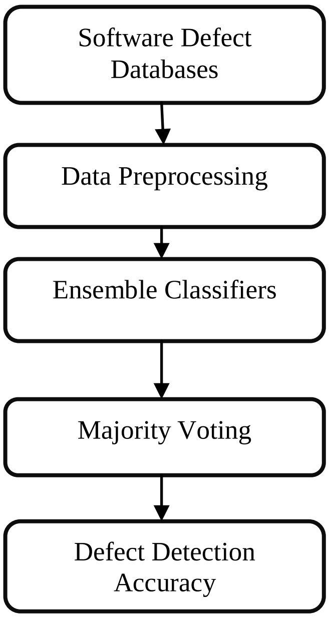

Figure. 1 Traditional ensemble defect prediction process

The functions selected for the predictive process as shown in Figure 1. Ensemble classification is defined as training multiple base classifier to detect software defects in the test data rather than using the minimal fault functionality. The imbalanced property is a major issue because of the poor performance of traditional ensemble classification models, particularly with respect to minority class characteristics.

Revised Manuscript Received on 30 May 2019.
* Correspondence Author

E.Sreedevi*, Research Scholar, Departement of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, A.P., India.

Dr.Y.Prasanth, Research Supervisor ,Departement of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, A.P., India.

Class imbalance and data uncertainty represent the growing research direction of the prediction of software defects aimed at finding better grading rates. In addition, traditional prediction models for software defects are evaluated on several semi-supervised classifiers, when software attributes or limited data size are limited. The metrics are considered in these models to be independent attributes and the attribute of a defect decision is considered dependent. The aim is to predict the defect (non-defect or defect) class of the software modules for the newly developed software modules or for the trained data. Software modules may not collect all defective data for their modules in some software companies. We have to develop powerful ensemble classifiers to find exact software predictions of imbalance and uncertain data in these situations [4].

In Section II, the literature study on the selections and classification of defects are discussed. Section III describes the proposed selection and classification model of the ensemble, section IV describes the experimental analysis, and section V concludes with the model.

II. BACKGROUND WORK

A survey will help developers to identify shortcomings with existing software defect methods, in particular Classification and software quality improvement, leading to a reduction in software development costs in the development and maintenance phase. Models which have been trained using ensemble methods generalize better than those which have been trained with standalone methods. In this paper, the ensemble methods of bagging, Adaboost, forest rotation and Random sub-space are implemented. Bagging (Bootstrap Aggregation) is designed to enhance machine learning algorithms' stability and accuracy.

Bagging forecasts multiple results from various training sets combined by uniform average or by voting. Adaboost performs multiple iterations with different weights, giving a final prediction by combining techniques. Rotation Forest uses a feature-extraction method to reconstruct a complete feature set in subsets of instances. Random Subspace creates a random forest of several decision-making bodies using a random selection approach. Some instances are randomly selected and allocated to the learning technique from the selected attributes. Another category of statistical techniques is neighboring (lazy-learning). Nearest neighbor students take more computing time during the testing stage and use hyperplanes to separate the two classes (i.e. defective or not), using the decision trees, neural networks, and bayesian networks. Each node of the decision tree is a feature in an instance which has to be classified, while each branch is represented for the hypothetical value of each node. Six different classifiers such as the Naive base, the Logistic regression, the Nearest Neighbor Instance, Bagging, J48, the Decision Tree and Random Forest were used for these purposes. Applied to five different open source software, this model identifies defects in 5885 classes based on object-oriented measurements. Of which only J48 and Bagging were the best. Feature selection is an important step in the process of defect prediction and is widely studied in the field of machine learning. They found wrapper to be the best model for limited data and limited set of features. Software defect prediction models are commonly classified as Decision tree, Support vector machine models, Artificial neural network models and Bayesian models in the literature as summed up in table 1.

Table. 1 Overview of Traditional Software Defect Prediction Models.

Classification Model	Imbalance Property	Problems
Decision Trees[5]	Affected	Adequate data required to avoid under fitting and overfitting problems.
Bayesian Models [6]	Affected	Required to find prior and posterior probabilities.
Artificial neural networks [7]	Affected	Not supports multi-class defects
Support Vector Machines [8]	Affected	Problem of feature selection
Ensemble Models [9-10]	Affected	High true positive rate

III. PROPOSED SOLUTION

A novel dynamic multi-software ensemble classification was developed and implemented in this paper to address the issue of cross-defect prediction. Our supervised classifier's main idea is to classify every software instance as "defect" or "no defect." Intuitive, the larger the size of the function set, the less exact the detection of defects. In the traditional models, as the training dataset is small, the prediction rate of defects could also be dramatically reduced because of class imbalance problems. Usually, the dynamic ensemble model is designed and implemented to improve the overall defect prediction rate of several projects. Figure 2 provides the overall proposed framework describing the use of the dynamic ensemble model in multiple software projects for cross-default detection. This framework has two main phases: one is the multi-project filtering model. In general, an ensemble learning model is generated from a group of basic classifiers for the detection of software. The ensemble model is designed in our model to predict the defects in several object-oriented software's. Classification of data is a two-stage process. The first step is to construct a model that describes the default data classes or concepts. Each attribute has a predefined class. In the training data set, the model data tuples are analyzed. The training data set is called samples and selected randomly from the sample groups. This step is also referred to as supervised learning when a label is given to each sample of training (e.g. every sample has a target class). The second step is to classify the data using the model. Compare the known class label with the learning model to each test sample data.

If the model's accuracy is acceptable, it may be used for an unknown element of the label class or for the classification of objects.

In this context, multiple software defects are analyzed with a broad range of features in the proposed ensemble learner model. The proposed filtering methods are used to filter every software metric. Different basic classifiers like C4.5, Naïve Bayes, Random Forest, Random Tree are used in our model to test the performance of the proposed model with conventional models. Lastly, dynamic test samples from each software project are tested for defect prediction against the ensemble model.

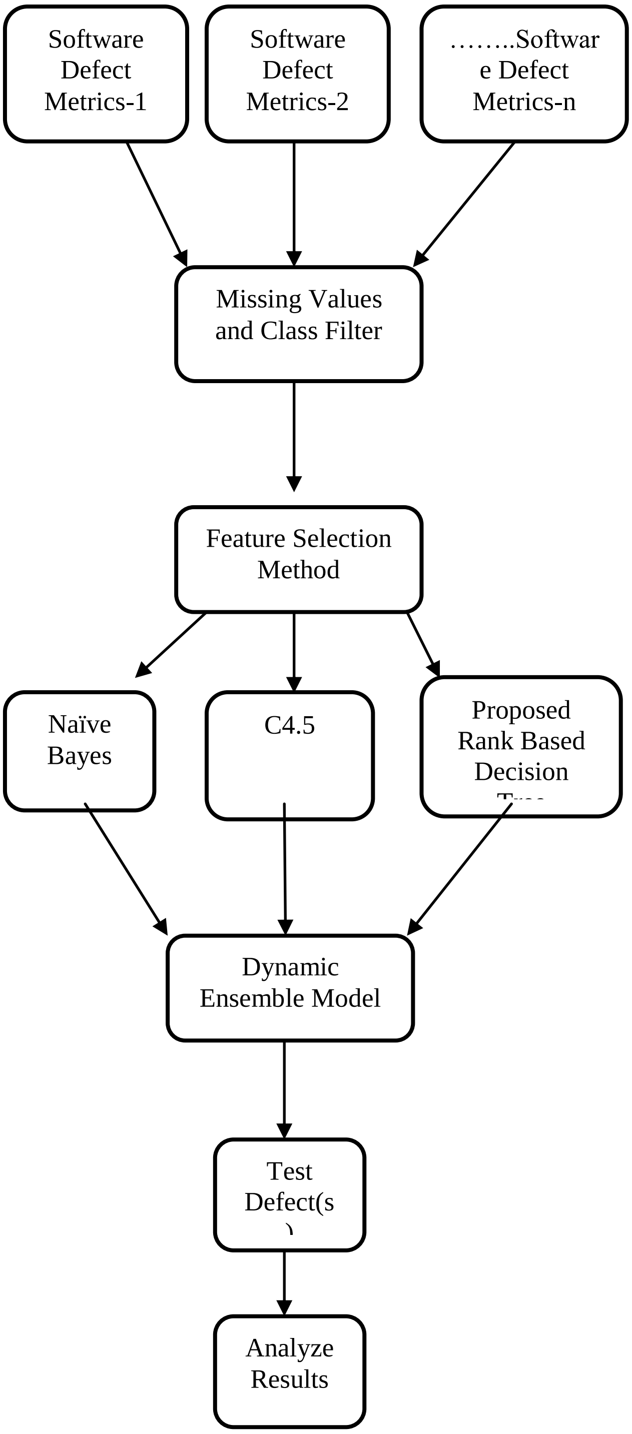
Software Defect Metrics-1
Software Defect Metrics-2
……..Softwar e Defect Metrics-n
Missing Values and Class Filter
Feature Selection Method
Naïve Bayes
C4.5
Proposed Rank Based Decision Tree
Dynamic Ensemble Model
Test Defect(s)
Analyze Results

Figure. 2 Proposed Dynamic Ensemble Learning Framework

A. Proposed Filtering Method:

```
Input: Software Defect Data SD-1,-…SD-n
Output: Filtered Data
Procedure:
Read dataset SD-1..SD-n
For each instance I in the Dataset
do
For each attribute A in the instance I
do
if(Ai==null)   //attribute is null
    $A(I) = \sum_{j=1/j\in\{SD-SD-i\}}^{n} (SD_j(X_i^2 - \mu_A))/SD_j(Max_A - Min_A)$
    -                                                        ----(1)
  End if
if(isNumeric(Ai) AND Ai(I)==null) //numeric instance is null
then
    $A_i(I) = \sum_{i=1/i\neq j}^{n} (X_i^2 - \mu_{A_i})/(Max_{A_i} - Miin_{A_i})$ -
                                                             -----(2)
end if
if(isNominal(Ai) AND Ai(I)==null) //nominal instance is null
then
    $A_i(I) = \sum Prob(SD_j(A_i(I))/Prob(SD_j \cap SD_k); j \neq k$
                                                             ----(3)
end if
if(isClass(Ai) not nominal)
then
   Convert attribute to nominal;
   if(bug>0)
   then
      Class label "Defect"
   End if
   Else if(bug==0)
   Then
      Class label "Not-Defect"
   End if
End for
```

Each attribute is tested for missing values in this filtering method. Most traditional methods can extract missing values from training data, which store numerical attributes. Filtering makes modeling faster and more accurate. If the attribute is numerical and the value is null, it is replaced with an equation value (1). Similarly, if the attribute is numerical and the value is zero, the value is replaced with calculated equation value (2). In addition, if the attribute is nominal and has missing values, the calculated equation value is replaced (3). Lastly, if the class is numeric, it will be nominal and labeled Defect and Not Defect. Data preprocessing is essential to improve the accuracy of the dynamic ensemble model as illustrated in Figure 3.

B. Ensemble Feature Selection Methods:

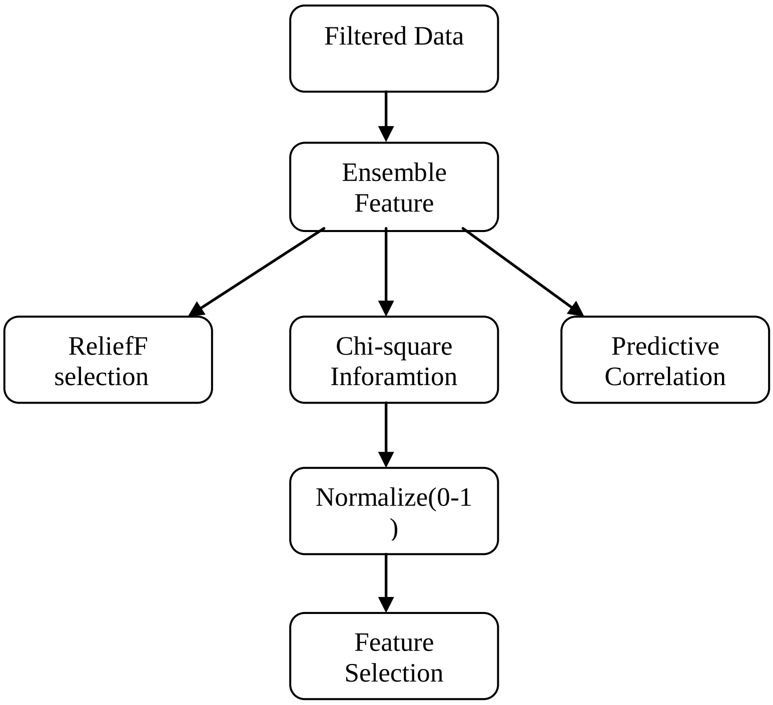

Figure. 3 Proposed Ensemble Feature Selection Model

C. ReliefF measure:

It is a traditional algorithm for feature selection. It is a relief algorithm optimization. In the first step, KNN instances are chosen from the training defect data, followed by the weight of the mean of every attribute. The second step is to calculate the correlation between each instance and the class. In the third stage, characteristics are sorted down by calculated weight to determine the relevance of each feature.

Dif(Feature, Instance-1,Instance-2)

For discrete features

Dif(A,I-1,I-2)= 0 ; if value of both A(I-1)=A(I-2) //I-1 is instance one and I-2 is instance 2.

Dif(A,I-1,I-2)=1; otherwise

For continous attributes

Dif(A,I-1,I-2)=|Val(I-1)-Val(I-2)|/Max(A)-Min(A))

D. Chi-square based defect's selection:

Chisquare can evaluate the defect by computing the Chi-square statistical on the distribution of the defect class. This is a non-parametric statistical approach used to determine the difference between the defect distribution observed and the actual defect distribution. Chi-square applies only to nominal, but not to continuous attributes.

E. *Improved Predictive Correlation Measure:*

```
Step 1: Initialize all the defect features, F.
Step 2: For each feature F(i)
        Do
    Compute Predictive correlation between the two features as
    Predictive Correlation PC= Corr(F[i]mF[i+1])/$\sum_{i=1}^{n} Prob(F[i]/F[i+1])$.
    if(PC>thres)
    Then
    D'=addFeature(F[i],F[i+1],PC);
    End if
    Done
```

Compute the ReliefF measure, chi-square measure, and predictive correlation measure between the feature metrics.

F. Multi-Ensemble Based Defect Detection Model

Algorithm:

```
Input: Ranked FeaturesData as FData;
Output: Model Learning and Defect Detection Output;
Procedure:
Read defect data feature set as FData
For each FeatureFData[i] in FData
Do
For each instance I(Ai) in Ai do
Do
        For each attribute FData(Di) do
Divide the data instances of FA(Di) into 'k' independent sets.
            Select classifier Ci/i=1…m
While i<k
Do
If(Ci is MLE)
Then
    $O_{ens} = (1/\mathrm{N}_i) * \sum_{m,i=1}^{M} P_m(x_i)$
Where $O_m$ is the output and is the input vector with N subsets.
Else if(Ci is NaiveBayes)
Then
Let $V = \{V_1, V_2 ... V_n\}$ be the discrete or continuous random variables used in the Bayesian computation values for defect prediction model. The probability computation of is shown as P($V_i$ / $\mathrm{a}_x$ ) where represents the parent nodes of .Then the joint probability distribution of X can be given as
    $\mathrm{Prob}(V) = \mathrm{Prob}(V_1, V_2 ... V_n)$
    $\mathrm{Prob}(V_1 / V_2, ... V_n).\mathrm{Prob}(V_2 / V_3, ... V_n)...\mathrm{Prob}(V_{n-1} / V_n)\,\mathrm{Prob}(V_n)$
$= \prod_{i=1}^{n} \mathrm{Prob}(V_i / V_{i+1}, ... V_n)$
Elseif(C4.5 classifier)
DT(i);//decision tree
End if Proposed Model
Load training features and instances
Contruct the Dynamic Defect Classifier using the following procedure:
a) Construct N subset of trained data and N subset of test data sampling with replacement.
b) In the tree growing phase, each and every node select kfeatures at random from Ncompute for best split
```

computation.

Proposed Best Split Computation can be measured using Modified entropy and Defect ratio. Modified entropy is given as ModEnt(D)= $-prob_i\sum_{i=1}^{m} l\,og\sqrt[i]{prob_i}$ *,m different classes*

Where

$$prob_i = prob(\mathrm{i})/\mathrm{prob}(\mathrm{D}_i)//i=1..m(classes)$$

In case of three classes:

ModEnt (Di)= $-prob_i\sum_{i=1}^{3} l\,og\sqrt[i]{prob_i}$

$$= -prob_1\log\sqrt{prob_1}+prob_2\log\sqrt[2]{prob_2}+prob_3\log\sqrt[3]{prob_3}$$

Where $prob_1$ *indicates probability of the set of instances which belongs to target class -1 ,* $prob_2$ *indicates probability of set of instances which belongs to target class -2.* $prob_3$ *indicates probability of set of instances which belongs to target class -3. Defect ratio measure to each attribute is computed as*

$$DefectRatio_A(D_m) = prob(D_i / D)*\sum_{i=1}^{m}|D_i|/|D|\times ModEnt(D_i)$$

Done

Done.

IV. RESULTS AND ANALYSIS

We use defect metrics such as: recall, accuracy and truth positivity to measure defect prediction measurements. These measures are widely used for the assessment of true prediction of defects. In this investigation we have used software defect metrics oriented at an openscience object with a variety of feature sets such as a) camel b) jedit c) ant d) poi and e) lucene[2]. These projects and its defect rates are summarized below:

Table. 2 Multiple defect datasets and its defect status

Project	Avg_Bug_Rate	Avg_Files
Camel	21%	350
Jedit	19.8%	650
Lucene	36.4%	410
Poi	41%	420
Ant	14.1%	491

Table 2, describes the various defect datasets and its statistical defect analysis for defect prediction process.

From the table 2, it is observed that the accuracy rate improves on an average of 15% when preprocessed with the proposed ensemble feature selection model and classification model.

Table. 3 Performance analysis of proposed approach with traditional methods using defect prediction rate

Defect Projects	ANN	Naive Bayes	Ensemble C4.5	Ensemble Random Forest	Proposed Model
Camel	0.8374	0.8586	0.8374	0.8318	**0.9743**
JEdit	0.8125	0.8782	0.8643	0.9098	**0.9378**
Lucene	0.8438	0.8854	0.8788	0.9183	**0.9289**
Poi	0.8271	0.8485	0.9145	0.8879	**0.9517**
Ant	0.8468	0.8976	0.8659	0.9289	**0.9698**

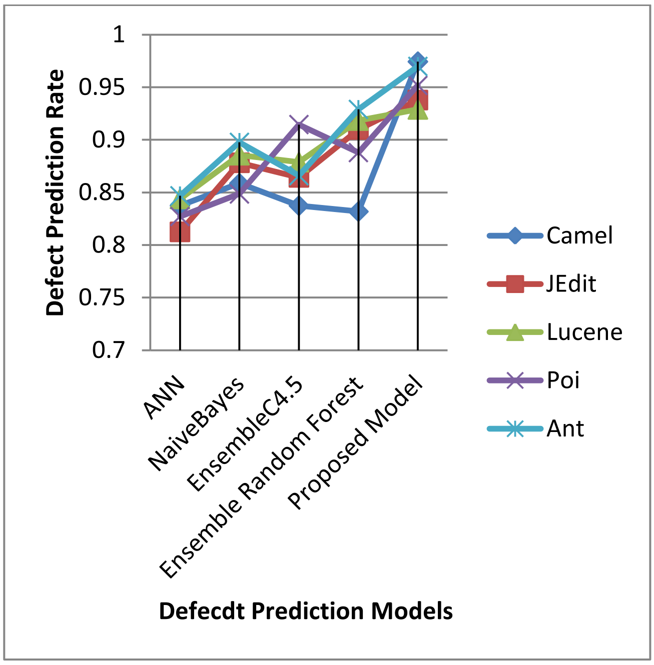

Figure. 4 Performance of the proposed model with the existing models

From the figure 4,it is observed that the accuracy rate improves on an average of 15% when preprocessed with the proposed ensemble feature selection model and classification model.

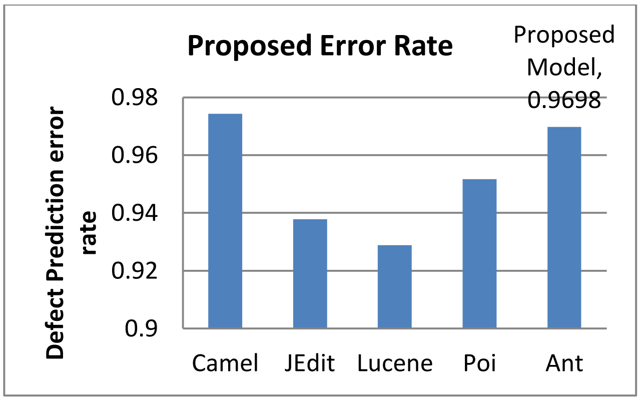

Figure. 5 Proposed model error rate on 24 features datasets.

V. CONCLUSION

As the size of software projects increases, the sparsity and uncertainty of the data increases, which affects the overall true positive rate of the defect prediction process? In this paper, a novel multi-ensemble feature selection and defect prediction model was designed and implemented on the openscience software defect dataset. ReliefF, Chi-square and improved predictive correlation measures are used in our ensemble feature selection process. Experimental results show that proposed model has high defect detection rate, recall and F-measure compared to the traditional software defect prediction models. In future, this work can be extended to improve the dynamic metric analysis in the web software metrics.

REFERENCES

1. T. Menzies, J. Greenwald, A. Frank, Data mining static code attributes to learn defect predictors, IEEE Trans. Softw. Eng. 33 (1) (2007) 2–13.
2. Z.A. Rana, S. Shamail, M.M. Awais, Towards a generic model for software quality prediction, in: Proceedings of the 6th International Workshop on Software Quality,WoSQ’08, ACM, 2008, pp. 35–40.
3. M. D'Ambros, M. Lanza, and R. Robbes, “Evaluating defect predictionapproaches: A benchmark and an extensive comparison,” Empiric.Softw. Eng., pp. 1–47, 2011.
4. S. Wang and X. Yao, “Using class imbalance learning for software defectprediction,” IEEE Trans. Rel., vol. 62, no. 2, pp. 434–443, Jun.2013.
5. E. J. Weyuker, T. J. Ostrand, and R. M. Bell, “Comparing the effectivenessof several modeling methods for fault prediction,” Empiric. Softw.Eng., vol. 15, no. 3, pp. 277–295, 2010.
6. Okutan, Ahmet and OlcayTanerYıldız. "Software Defect Prediction Using Bayesian Networks". Empirical Software Engineering 19.1 (2012): 154-181. Web. 17 Oct. 2016.
7. Jindal, Rajni, Ruchika Malhotra, and Abha Jain. "Software Defect Prediction Using Neural Networks". Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization (2014): n. pag. Web. 17 Oct. 2016.
8. Li, Feixiang, Xiaotao Rong, and Zhihua Cui. "A Hybrid CRBA-SVM Model For Software Defect Prediction". International Journal of Wireless and Mobile Computing 10.2 (2016): 191. Web. 17 Oct. 2016.
9. Wang, Shihai, He Ping, and Li Zelin. "An Enhanced Software Defect Prediction Model With Multiple Metrics And Learners". IJISE 22.3 (2016): 358. Web. 17 Oct. 2016.
10. Petric, Jean, Tracy Hall, and Nathan Baddoo. "Building An Ensemble For Software Defect Prediction Based On Diversity Selection". Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '16 (2016): n. pag. Web. 17 Oct. 2016

AUTHORS PROFILE

Sreedevi Emandi persuing her PhD from KLUniversity,Guntur(Dist)India.She received her M.Tech degree in computer Science and Engineeringfrom Acharya Nagarjuna University in 2010.She is an Assistant Professor in the department of computer Science and Engineering in KLUniversity from 2007 to till date.Till now she has published 10 papers in various International Journals.Her Research interests include software Engineering,Machine Learning.She taught several subjects like C Programming, Data Structures, Network Security and Python Programming.

Prasanth Yalla received his B.Tech Degree from Acharya Nagarjuna University, Guntur (Dist), India in 2001, M.Tech degree in Computer Science and Engineering from Acharya Nagarjuna University in 2004, and received his Ph.D. degree in CSE titled “A Generic Framework to identify and execute functional test cases for services based on Web Service Description Language” from Acharya Nagarjuna University, Guntur (Dist), India in April 2013. He was an associate professor, with Department of Information Science and Technology in KL University, from 2004 to 2010. Later he worked as Associate professor, with the department of Freshman Engineering from 2011 in KL University. Presently he is working as Professor in the department of Computer Science & Engineering in KL University and also Associate Dean (R&D) looking after the faculty publications. Till now he has published 28 papers in various international journals and 4 papers in conferences. His research interests include Software Engineering, Web services and SOA. He taught several subjects like Multimedia technologies, Distributed Systems, Advanced Software Engineering, Object Oriented Analysis and design, C programming, Object-Oriented programming with C++, Operating Systems , Database management systems, UML etc. He is the Life member of CSI and received “Active Participation- Young Member” Award on 13-12-13 from CSI. He has applied a project to SERB very recently.