International Journal of Control Theory and Applications

ISSN : 0974–5572

Volume 9 • Number 40 • 2016

A Novel Multi-Ensemble based Feature Selection and Defect Prediction Model on Software Defect Projects

E. Sreedevi[a] and Y. Prasanth[a]

[a]*Department of Computer Science and Engineering, KLUniversity, Vaddeswaram, AP, India*

E-mail: sridevi_fed@kluniversity.in

Abstract : The prediction of software defects is an essential step before building high quality software defect classification models .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 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 open science software defect dataset. Relief F, 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.

1. 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. Unfortunaetly, 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.

Defect prediction provides an optimized way to find the vulnerabilities in the modern software modules, which occurs due to manual or automatic errors. As the dependency of software modules increasing, software quality is becoming more and more essential in present era. Software defects such as failures and faults may affect the quality of software which leads to customer dissatisfaction [1]. Due to the increasing of software constraints and modular complexity, it is too difficult to produce a quality end product.

Defects in software may cause loss of money and time, so it is necessary to predict bugs in advance for successful quality products and decision makers. As a result, these bug reports present in various bug tracking frameworks contains detailed information about the bugs along with the severity level[1-3]. Since the software

quality estimation model is based on the software metrics of the software modules, the selection of relevant metrics or dependency metrics becomes an integrated part of the model building process.

Traditional feature selection models can be classified into two groups, one is the feature subset selection and another one is feature ranking. In feature ranking models, each feature is assessed according to the computed measures and then an analyst selects relevant features for a given data set. A feature subset selection model extracts a subset of features from the large set of features using selection measures [2].

The software quality has been measured using static and defect analysis of the software. In order to predict the software defects, the static and dynamic code metrics should be understandable for structural design and complexity of the overall project[3]. In this static code metrics, some of them are highly relevant features with the defects but others have less number. So instead of using minimal defect features, the selected features should be detected and used in the prediction process as shown in Figure 1.

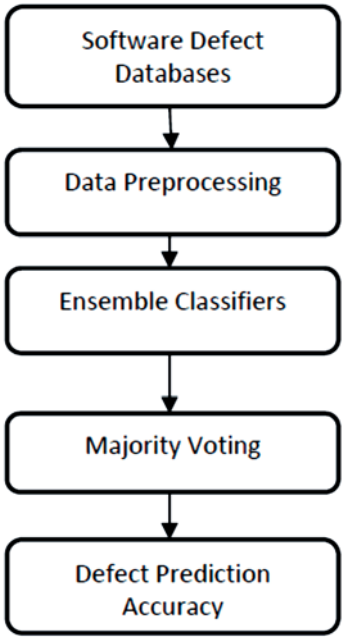

Figure 1: Traditional ensemble defect prediction process

Ensemble classification is defined as the training of multiple base classifiers to detect the software defects in the test data. The imbalanced property is a primary issue accounting for the poor performance of the traditional ensemble classification models, especially on the minority class attribute. Class imbalance and data uncertainty are the growing research direction in the software defect prediction that aims to discover better classification rate. Also, traditional software defect prediction models are evaluated on several semi-supervised classifiers when there are limited software attributes or limited datasize. In these models, metrics are considered as independent attributes and defect decision attribute is considered as dependent variable. The aim is to predict the defect class (either Non-defect or Defect) of the software modules for the newly designed software modules or trained data. In some software companies, software modules may not collect all defect data for their modules. In these situations, we need to develop powerful ensemble classifiers that can discover accurate software prediction on unbalance and uncertain data [4].

The rest of this paper is organized as follows; Section II describes the literature study of different defect selection and classification models. Section III describes the proposed ensemble feature selection and classification model, Section IV describes the experimental analysis and in Section V, we conclude with the model.

2. BACKGROUND WORK

Feature selection is an important step in defect prediction process and has been extensively studied in the field of machine learning. [5] Investigated different feature selection models that produce ranked list of features and applied them to UCI repository datasets. They concluded that wrapper is best model for limited data and limited feature sets. Software defect prediction models are commonly classified in the literature as Decision tree models, Support vector machine models, artificial neural network models and Bayesian models are summarized in the Table 1.

Dynamic analysis extends traditional testing to check meaningful properties using intermediate state information in program executions. Dynamic analysis techniques share the limitations of testing inherently. Dynamic analysis cannot support complete analysis for target programs since it uses monitored partial behavior of the target programs. The other limitation is that dynamic analysis techniques are difficult to be applied unless target programs are complete. Dynamic analysis techniques require executable environments and test cases. However, these can be delivered only at later phase of software development especially for embedded software. Decision tree classification algorithm is widely used in statistics, data mining, machine learning. The goal is to create a decision tree model, which uses the given input data to predict the target data classification. For the nodes within the tree, we compare the attribute values. Each branch is a possible classification for the target data. Leaf node is the classification of the target data. In order to classify the unknown target data; target data attribute value judgment on the decision tree. The determination of the defect can be represented as the leaf node with the path, which corresponds to a classification rule.

Decision tree classification process consists of two phases: tree construction, tree pruning. Tree construction: this phase can be seen as the variables selection process, all the problems are basically partition down to two things: choose suitable variables as splitting nodes, and how to split, splitting rules.

3. PROPOSED SOLUTION

Table 1
Overview of Traditional Software Defect Prediction Models

Classification Model	*Imbalance Property*	*Training Data*	*Advantages*	*Disadvantages*
Decision Trees [5]	Affected	Adequate data required to avoid under fitting and over fitting problems.	Robust to noisy data; and decision rules evaluation.	Prone to over-fitting. Performance issue under imbalanced property.
Bayesian Models [6]	Affected	Required to find prior and posterior probabilities.	Robust to probabilistic predictions.	Requires domain expert for decision making. Computationally expensive.
Artificial neural networks [7]	Affected	Data required for training model.	Able to learn non-linear functions. Robust against errors.	Difficult to interpret results. Slow training and prediction process.
Support Vector Machines [8]	Affected	Data required for training model.	Best model for high dimensional datasets with complex kernel functions.	Slow processing Low performance under limited features.
Ensemble Models [9-10]	Affected	Data required for training model.	Best model for high dimensional datasets with complex feature selection models	Fast processing Low performance under imbalanced data and missing values.

In this paper, a novel dynamic multi-software based ensemble classifier was designed and implemented to address the problem of cross defect prediction. The key idea of our supervised classifier is to classify each software instance into "Defect" or "Not-Defect". Most of the the traditional single defect prediction models are supervised which considers defects information as trained data. Intuitively, larger the size of the feature set, lower the accuracy of defect detection. Also, as the size of the training dataset is small, defect prediction rate could be dramatically reduced due to class imbalance problem. Dynamic ensemble model is usually designed and implemented to improve the overall defect prediction rate on multiple projects. Dynamic ensemble model can be done in parallel to scale up prediction process. Figure 2, presents the overall proposed framework that describes the use of dynamic ensemble model for cross defect detection on multiple software projects. This framework has two main phases one is multi-project filtering model and the second one is dynamic ensemble learning model for defection prediction process. Generally, ensemble learning model is generated from a group of base classifiers to predict the software detection process. In our model, ensemble model is designed to predict the defects in multiple object oriented softwares.

In this framework, multiple software defects are analyzed using the proposed ensemble learner model with large number of feature set. Each software metrics are filtered using the proposed filtering methods. In our model, different base classifiers such as C4.5, Naïve Bayes, Random forest, Random Tree are used to test the performance of proposed model to the traditional models. Finally, dynamic test samples from each software projects are tested against the ensemble model for defect prediction.

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

$$A(I) = \sum_{j=1/\, je\{SD-SD-i\}}^{n} (SD_j(X_i^2 - \mu_A)) / SD_j(Max_A - Min_A) \quad (1)$$

End if

If (isNumeric(Ai) AND Ai(I)==null) //numeric instance is null

then

$$A_i(I) = \sum_{i=1/i\neq j}^{n} (X_i^2 - \mu_{A_i}) / (Max_{A_i} - Miin_{A_i}) \quad (2)$$

end if

if (isNominal(Ai) AND Ai(I)==null) //nominal instance is null

then

$$A_i(I) = \sum Prob(SD_j(A_i(I)) / Prob(SD_j \cap SD_k); j \neq k \quad (3)$$

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

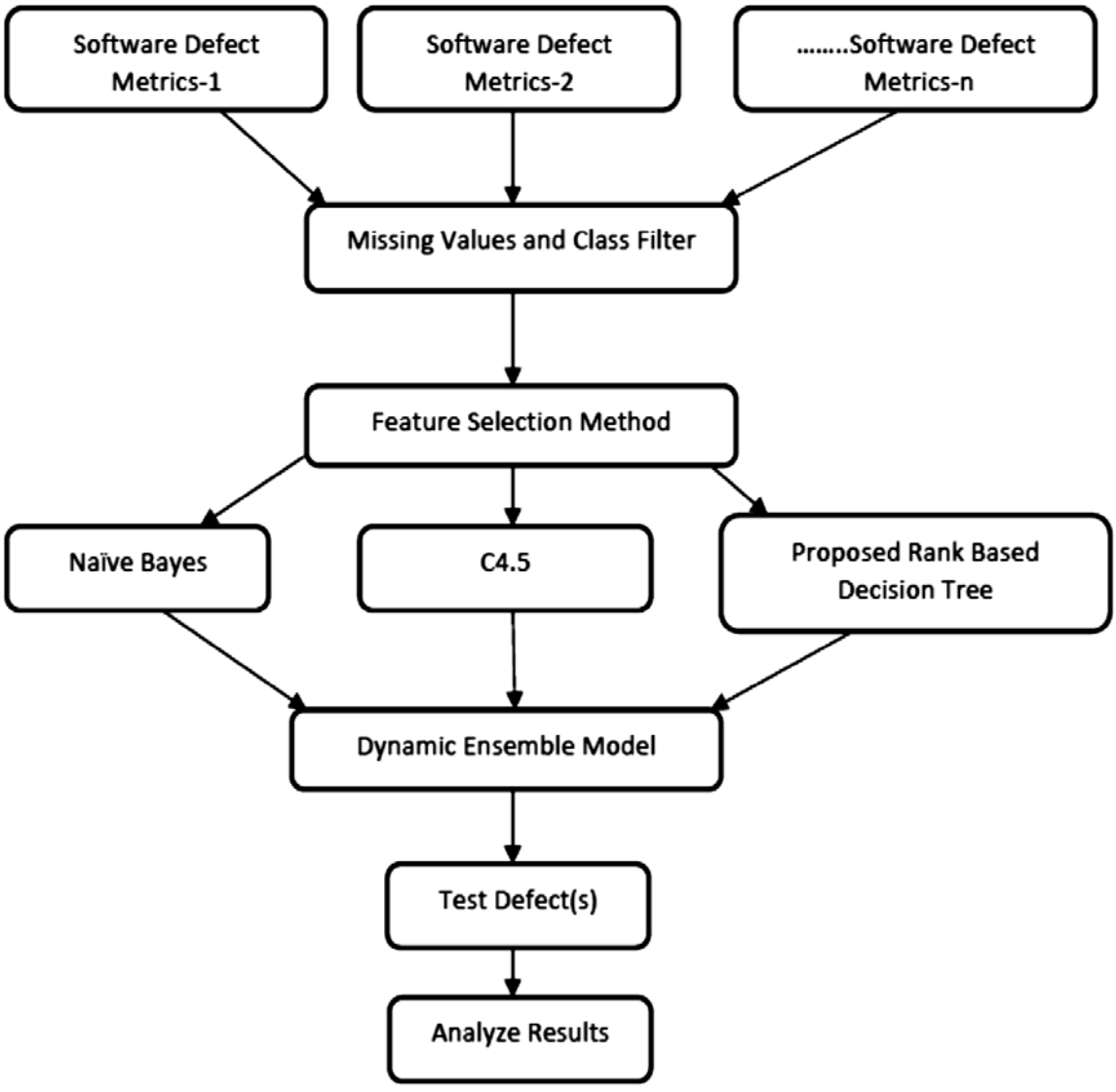

Figure 2: Proposed Dynamic Ensemble Learning Framework

In this filtering method, each attribute is tested for missing values. Most of the traditional methods are capable of extracting missing values from training data that store numeric attributes. Filtering makes models more accurate and faster.If the attribute is numerical and the value is null then it is replaced with computed value of equation (1). Similarly, if the attribute is numerical and the values are null, then it is replaced with computed value ofequation (2). Also, if the attribute is nominal and has missing values then it is replaced with computed value of equation (3). Finally, if the class is numeric then it is converted to nominal and labeled with Defect and Not-Defect. Preprocessing of the data is essential to improve the accuracy of the dynamic ensemble model as shown in Figure 3.

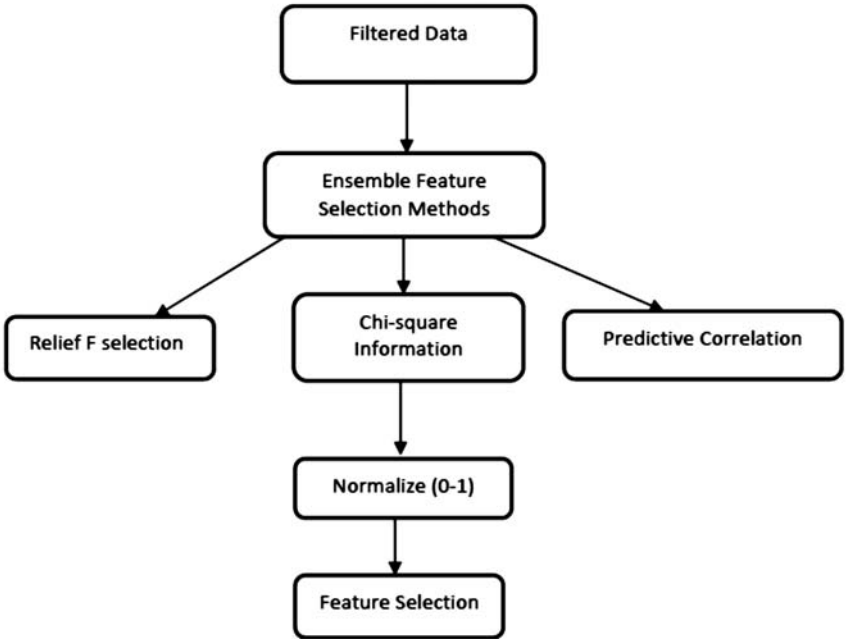

Figure 3: Proposed Ensemble Feature Selection Model

Ensemble Feature Selection Methods:

ReliefF measure: ReliefF [1] measure is one of the traditional feature selection algorithm. It is an optimization of Relief algorithm. In the first step, KNN instances are selected from the training defect data and then the mean of every attribute is selected as the weight. In the second step, correlation between the each instance and the class is computed. In the third step, features are sorted in descending order according to the computed weight to determine the importance of each feature.

Difif(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 continuous attributes

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

Chi-square based defect's selection: Chisquare can evaluate the defect by computing the chi-square statistic with respect to the defect class distribution.This is non-parametric statistical approach used to find the difference between the observed defect distribution to the actual defect distribution. Chi-square is only applicable to nominal attributes but not continuous attributes.

Improved Predictive Correlation Measure:

Step 1: Initialize all the defect features, F.

Step 2: For each feature F(i)

Do

Compute the ReliefFmeasure, chi-square measure, and predictive correlation mreasurebetween the feature metrics.

Compute Predictive correlation between the two features as

Predictive Correlation PC=Corr(F[i],F[i+1])/ $\sum_{i=i}^{N}$ Prob(F[i] / F[i+1])

If(PC>thres)

Then

D' =addFeature(F[i],F[i+1],PC);

End if

Done

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(Ciis MLE)

Then

$$O_{ens} = (1 / \mathrm{N}_i) * \sum_{m,i=1}^{M} P_m(x_i)$$

Where O_m is the output and x_i is the input vector with N subsets.

Else if(Ciis 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 V_i is shown as $P(V_i / a_x)$ where a_x represents the parent nodes of V_i .Then the joint probability distribution of X can be given as

$$\text{Prob}(V) = \text{Prob}(V_1, V_2 ... V_n)$$

$$\text{Prob}(V_1 / V_2, ... V_n).\text{Prob}(V_2 / V_3, ... V_n) ... \text{Prob}(V_{n-1} / V_n)\,\text{Prob}(V_n)$$

$$= \prod_{i=1}^{n} \text{Prob}(V_i / V_{i+1}, ... V_n)$$

Elseif(C4.5 classifier)

DT(i);//decision tree

End if Proposed Model

Load training features and instances

Construct the Dynamic Defect Classifier using the following procedure:

1. Construct N subset of trained data and N subset of test data sampling with replacement.
2. In the tree growing phase, each and every node select k features 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

$$\text{ModEnt(D)} = -prob_i \sum_{i=1}^{m} l\,\text{og} \sqrt[i]{prob_i} \text{ , m different classes}$$

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

In case of three classes:

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

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

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

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

The term prob(Di /D) acts as the weight of the subset data using conditional probability occurence.

Sort the k individual trees according to defects and not-defects .

Select the majority voting available in each tree using ensemble learning.

End while

Calculate misclassified error rate and statistical true positive rate;

Done

Done

4. RESULTS AND ANALYSIS

To measure the defect prediction measures, we use defect metrics such as : Recall , Precision and True positive rate. These measures are widely used to evaluate true defect prediction analysis. In this research we have used open science object oriented software defect metrics with different values of the feature sets such as a) Camel b) Jedit c) ant d) poi and e) lucene[2]. These projects along with 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

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

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.

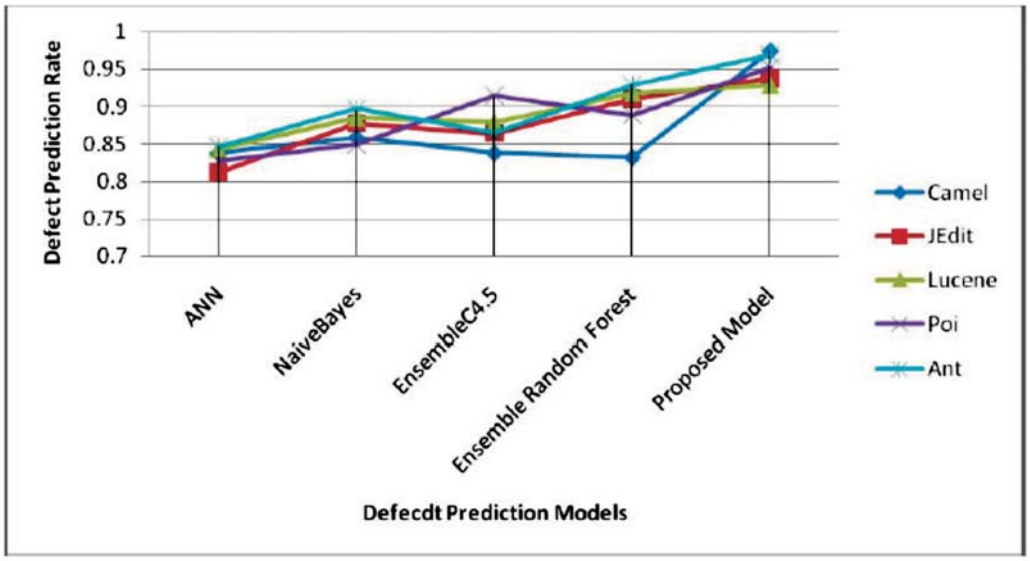

Figure 4: Performance of the proposed model with the existing models

5. 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 resuts 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, RuchikaMalhotra, 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, XiaotaoRong, 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.