COST SENSITIVE SOFTWARE DEFECT PREDICTION TECHNIQUE USING GENETIC ALGORITHM

G N V Ramana Rao
Group Manager, Wipro Ltd.

Dr. VVSSS Balaram
Professor in IT, SNIST, Hyderabad

Dr. B. Vishnuvardhan
Professor in CSE, JNTUH, Hyderabad.

ABSTRACT

Research has yielded approaches to predict future defects in software artifacts based on historical information, thus assisting companies in effectively allocating limited development resources and developers in reviewing each other’s code reduces the cost of maintenance. Developers are unlikely to devote the same effort to inspect each software artifact predicted to contain defects, since the effort varies with the artifacts' size (the number of LOC and cost) of defects that it exhibits (effectiveness). We propose to use Genetic Algorithms (GAs) for training prediction models to maximize their cost-effectiveness. We evaluate the approach on two well-known models, Regression Tree and Generalized Linear Model, and predict defects between multiple releases of 5 open source projects. Our results show that regression models trained by GAs significantly outperform their conventional counterparts, improving the cost-effectiveness by up to 120%.

Key words: Machine Learning, Defect Prediction, Software Engineering, Statistical Methods, Expert Systems, Feature Selection, Regression Tree, Generalized Linear Model (GLM)

Cite this Article: G N V Ramana Rao, Dr. VVSSS Balaram and Dr. B. Vishnuvardhan, Cost Sensitive Software Defect Prediction Technique Using Genetic Algorithm, International Journal of Computer Engineering and Technology, 9(6), 2018, pp. (1)-(10).
http://www.iaeme.com/IJCET/issues.asp?JType=IJCET&VType=9&IType=6
1. INTRODUCTION

Statistical modeling has been often utilized in software engineering to assess quality of software projects. One of its frequent applications would be to produce prediction models to anticipate exactly where defects occur in a software system. Such models are actually valuable in several contexts: For instance, Kim et al. show the importance of theirs in effective API testing, where prediction models take the testing effectiveness in manufacturing environment [20]. Researchers and practitioners underline the importance of proper allocation of computing resources [11], for instance during code assessment [5], to save the appraisal cost of the code.

During initial research, researchers (e.g., [41]) had examined prediction models to make a binary distinction of each an application artifact: Likely or perhaps not very likely to incur in succeeding defects. Widely used evaluation metrics were precision as well as recall [41] or maybe the area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve. The AUC plots the classes properly classified as defective against individuals improperly classified as defective, as the prediction model’s discrimination threshold can vary.

Recently, scientists observed that the effort needed by designers towards inspecting artifacts recommended by binary distinction can vary based on the artifact [25]. Bigger and much more complicated software artifacts require more inspection effort, therefore hindering both the effectiveness and the usefulness of the prediction. As a solution, scientists have suggested to rethink prediction in conditions of cost effectiveness: Artifacts must be inspected in the order which maximizes the ratio between the amount of defects discovered as well as the effort spent (effort typically approximated by the dimensions of the artifacts) [11]. With this context, generally used evaluation metrics are: (i) the cost effective AUC (AUC CE) [11], that represented a weighted model of the more conventional AUC metric; and (ii) the P effort metric [11]. In every effort aware prediction model presented very far no matter the employed statistical mechanism the design is not immediately taught to get the best match to rank on the cost effectiveness, quite to foresee the raw selection of defects, i.e., an approximation of it. The concept we offer and assess in this paper is using

Genetic algorithms (GAs) to immediately tweak the coefficients of a prediction version like that the cost effectiveness on the instruction set is actually maximized. Menzies et al. [28] had been the first person to suggest the thought of self-tuning the inner parameters of a principle learner to obtain the correct settings, therefore leading to a learner which considerably outperforms regular learning strategies [28]; here, we wish to immediately train statistical models. We utilize GAs to evolve the coefficients of regression algorithms to create an unit optimizing the cost effectiveness on the instruction set, under the presumption that it’ll also predict cost effectiveness more efficiently on the examination set. We evaluate the idea of ours by applying it and doing an empirical analysis on a selection of unique application systems and releases. As the baseline of ours, we think about extensive statistical regression models (i.e., generalized linear regression version (GLM) as well as regression trees (RT)) as well as metrics (i.e., Kemerer and Chidamber (CK) metrics as well as Lines of Code (LOC)). Our results indicate that our strategy significantly outperforms traditional models.

In earlier endeavors, researchers (e.g., [41]) had studied prediction models to present a binary classification of each software program artifact. Commonly used evaluation metrics were precision in addition to recall [41] or the area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve. The AUC plots the classes correctly classified as defective against those wrongly classified as defective, as the prediction model’s discrimination threshold may differ.

Not too long ago, researchers pointed out that the effort expected by developers towards inspecting artifacts indicated by binary classification models differs based on the artifact [25]. Larger and a lot more complex software artifacts require extra inspection effort, thereby
hindering both the usefulness and the effectiveness of the prediction. As a solution, researchers have recommended to rethink prediction in terms of cost-effectiveness: Artifacts have to be inspected in the order which often maximizes the ratio between the number of defects detected together with the effort spent (effort often approximated by the size of the artifacts) [11]. In this particular context, often used evaluation metrics are: (i) the cost-effective AUC (AUC-CE) [11], which in turn represents a weighted model of the more classic AUC metric; and (ii) the P effort metric [11]. In every effort aware prediction model presented very far no matter the employed statistical mechanism the design is not immediately taught to get the best match to rank on the cost effectiveness, quite to foresee the raw selection of defects, i.e., an approximation of it. The concept we offer and assess in this paper is using

Genetic Algorithms (GAs) to immediately tweak the coefficients of a prediction version like that the cost effectiveness on the instruction set is actually maximized. Menzies et al. [28] had been the first person to suggest the thought of self-tuning the inner parameters of a principle learner to obtain the correct settings, therefore leading to a learner which considerably outperforms regular learning strategies [28]; here, we wish to immediately train statistical models. We utilize GAs to evolve the coefficients of regression algorithms to create an unit optimizing the cost effectiveness on the instruction set, under the presumption that it'll also predict cost effectiveness more efficiently on the examination set. We evaluate the idea of ours by applying it and doing an empirical analysis on a selection of unique application systems and releases. As the baseline of ours, we think about extensive statistical regression (i.e., generalized linear regression version (GLM) as well as regression trees (RT)) as well as metrics (i.e., Kemerer and Chidamber (CK) metrics as well as Lines of Code (LOC)). Our results indicate that our strategy significantly outperforms conventional models.

2. EXISTING WORK
Researchers devised a number of defect prediction approaches to guide software maintenance as well as evolution by identifying more defect software artifacts [11]. These approaches are based on statistical models, whose main difference is actually the various sets of predicting metrics and also the underlying algorithms that learn from these metrics and make predictions [15,37]. Examples of metrics are the Chidamber and Kemerer's object oriented (CK) metrics [10,6], structural metrics [3] or process metrics [29]. Examples of algorithms are logistic regression used by Zimmermann et al. [40]; Multi-Layer Perceptron (MLP), radial foundation functionality (RBF), k nearest neighbor (KNN), regression tree (RT), dynamic evolving neuro-fuzzy inference body (DENFIS), and Support Vector Regression (SVR) used by Elish [fourteen]; Bayesian networks utilized by Bechta [31]; and Naive Bayes, J48, Alternative Decision Tree (ADTree), and One R believed by Nelson et al. [30]. Lately, various other researchers have suggested more advanced machine learning strategies, including ensemble learning [23], clustering algorithms [36], and combined techniques [thirty two]. Lessman et al. [22] evaluated twenty two classification models and showed that there's simply no statistical difference between the top-17 models when classifying software modules as defect susceptible. Meta-heuristics have been additionally investigated, such as using genetic algorithms (GAs) [13,19,23] or maybe genetic programming (GP) [1] to build prediction design aimed at optimizing standard performance metrics for classification problems, such as precision, recall, and f-measure.

Mende et al. [25], Kamei et al. [18], Menzies et al. [28], and D 'Ambros et al. [11] are actually among the first to warn of the significance of taking into consideration the effort necessary to review the documents recommended by prediction models. Traditional performance metrics employed for binary predictions (precision, recollection, f-measure, AUC [32], error variance, median error, error sum, and correlation [11]) aren't well suited to assess
prediction since they provide exactly the same priority/importance to all defect prone software components. Rather, in useful scenario engineers would gain from determining those software components apt to contain far more defects earlier, or perhaps requiring lower inspection price at the exact same number of defects. Consequently, prediction techniques must be cost effective, the place that the effectiveness is actually number of defects to foresee as well as the assessment cost is actually approximated by the collections of code (LOC) metric, depending on the intuition that bigger files call for effort and time more to discuss than smaller files [11,25,18].Previous labor suggested performance metrics (e.g., AUC CE and Peffort) created for evaluating the cost effectiveness of prediction models [11, 34,17,32, 33,28]. Nevertheless, the designs were still made working with conventional knowledge algorithms; for instance, D'Ambros et al. [11] trained conventional linear regression models with the classical iteratively re weighted very least square algorithm; Rahman and Devanbu [33] used 4 different machine learning strategies (i.e., SVM, J48, Logistic Regression, and Naive Bayes) that had been used in utilizing the corresponding classical instruction algorithms.

3.EXISTING CHALLENGE
Even though previous work [25,18,11] proposes to assess prediction design by using new effort aware metrics grounded on the essential assessment hard work, these metrics haven't been accustomed to instruct the statistical versions. In reality, statistical and machine learning methods have been accustomed to find models which lessen the prediction error when computing the selection of defects in a software program artifact, not to optimize the cost effectiveness (which is actually a unique problem). Moreover, cost effective metrics are used to evaluate the final quality of a product (e.g., as being a post deployment process). Therefore, models are actually constructed onto a program set optimizing several functionality metric (e.g., family member error as well as accuracy with respect to selection of defects), though they're evaluated on the code set by using various metrics (e.g., AUC CE).

Menzies et al. [28] had been the first to suggest the thought of thinking about the effort (LOC) in the internal loop of a principle learner (i.e., WHICH). The WHICH framework parameters had been physically tuned to find the functions leading to the most desirable cost-effectiveness and this also lead WHICH to significantly outperform regular learning strategies [28]. In training, the proposed method to (i) physically tune the parameters as well as (ii) personally examine the outcomes on the training set to confirm whether the performance is actually enhanced with respect to basic learning techniques.

In this work, we wish at immediately discovering the inner parameter values which optimize the cost effectiveness, rather than self-tuning them. To this particular aim, we show a broad framework to train some regression trees(RT) with Genetic Algorithms to successfully examine the search space of potential parameters, the place that the quality of every parameter's environment (represented as a GA individual) is actually evaluated based on its cost effectiveness recognized on the defect data.

In the following we show the 2 statistical versions that we utilize to instantiate our normal framework (i.e., Regression Tree and GLM ).These statistical models make diverse assumptions over the instruction information and have been popular in a selection of defect prediction scenarios [11,21], therefore making them great candidates as subjects for the study.

4.GENERALIZED LINEAR MODEL(GLM) REGRESSION
A Generalized Linear Model (GLM) is a model, is basically a regression model with relaxed rules, consists of three components: (I) a link function (II) a variance function(μ) and (III) Independent Variables. The link relates the means of the observations to predictors: linearization (μ) The variance function relates the means to the variances. In our case the (i)
independent variables $I = \{I_1, I_2, \ldots, I_k\}$ are the software metrics used as explanatory variables of the scalar dependent variable $Y$, i.e., the number of defects. The linear function condenses the independent variable into a scalar value with a set of linear coefficients $\beta = \{\alpha, \beta_1, \beta_2, \ldots, \beta_k\}$ such that

$$\eta = \alpha + \beta_1 I_1 + \beta_2 I_2 + \cdots + \beta_k I_k$$

(1)

The link function $\psi$ that provides the relationship between the expectation of the outcome and the linear function.

$$\psi = \alpha + \beta_1 I_1 + \beta_2 I_2 + \cdots + \beta_k I_k$$

(2)

If the above function represents the variables from normal distribution then the equation corresponds to traditional multinomial regression. The General Linear Model (GLM) in Eq (2) finds the set of coefficients $\beta = \{\alpha, \beta_1, \beta_2, \ldots, \beta_k\}$ such that the corresponding model $\psi$ minimizes the Mean Squared Error (MSE) between predicted value and actual outcome.

4.1 Regression Tree

A regression tree is similar to a decision tree where internal decision nodes contain the decision rules on software metrics while leaf nodes are defect prediction outcomes. The decision rules are based on the software metric $M_i$ and coefficient $x_i$ verifies the test condition is reached or not, thus by partitioning the space into two branches (true or false).

![Figure 1 Regression Tree](image)

For every class instance the classification is performed to predict the number of defects. The regression is trained with the training set data to predict the defects. The traditional tree models are used to find the minimum mean squared error (MSE).[35]

5. PROPOSED APPROACH

We propose that if the aim of the prediction model is cost effectiveness and evaluated differently with respect to traditional models then the models trained using different approach would show better results. Hence we propose to modify model training such that software engineering artifacts with higher defect density will be given higher priority.

In this regard, we wish to maximize the Mean Squared Error (MSE), which is ratio between cumulative number of defects and amount of code under inspection. Let $A = \{a_1, a_2, \ldots, a_k\}$ be the list of artifacts to be reviewed in the training set ordered by their predicted scores computed by the regression equation from Eq (2) where $\beta = \{\beta_1, \beta_2, \ldots, \beta_k\}$ is a set of regression coefficients. We hypothesize the defect prediction as:
To find the set of regression coefficients \( \beta = \{ \beta_1, \beta_2, \ldots, \beta_k \} \) from regression equation (2) such that coefficients maximizes the cost-effectiveness i.e maximizing the cumulative number of defects while inspecting the Lines of Code for the predicted artifacts in the order \( O = \{ o_1 o_2 \ldots o_k \} \)

\[
\varphi_{\text{max}}(f) = \sum_{i=2}^{n} \left( \sum_{j=1}^{i-1} \text{defects } O_j \times \text{LOC}(O_i) \right)
\]

Eq(3)

Where \( \sum_{j=1}^{i-1} \text{actual } O_j \) represents cumulative defects for the first (i-1) artifacts. Finally \( \text{LOC}(O_i) \) measures the lines of code(LOC) of i\(^{th}\) artifact as shown in following figure (Fig (2)).[25,4,11,8,32,9]

5.1 Training the Model with GA

To solve the maximization problem we apply nature inspired GA on the Regression Trees and GLM. The search space represents possible sets of linear combination of coefficients \( \beta = \{ \alpha, \beta_1, \ldots, \beta_n \} \) for GLM and \( X = \{ x_1, x_2, \ldots, x_n \} \) Decision coefficients of RT.

The algorithm starts with a randomly generated GLM or RT configurations. During the subsequent iterations, the population is evolved with genetic operators such as (i) selection (ii) Crossover and (iii) Mutation. During each iteration, the coefficients are evaluated according the function represented in equation (2). The best or fittest coefficients are selected using selection operator. During the same time new set of coefficients are generated by recombination technique using Crossover and Mutation operators. After each generation finally selected coefficients are used as starting point for next selection or iteration cycle.

6. EXPERIMENT AND RESULTS

In order to evaluate the cost – effectiveness in our approach compared to traditional approach we present the design of study as follows.

Data from open source, PROMISE repository, projects based on Java are selected for our study and details are mentioned on the following table. (ref Tab (1))
Table 1: Projects used for our study

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Release</th>
<th># Classes</th>
<th>% Defective Classes</th>
<th>Average No. Of Defects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Velocity</td>
<td>1.4</td>
<td>197</td>
<td>74.62</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td>1.5</td>
<td>215</td>
<td>66.05</td>
<td>2.3</td>
</tr>
<tr>
<td></td>
<td>1.6</td>
<td>237</td>
<td>33.91</td>
<td>2.4</td>
</tr>
<tr>
<td>Xlan</td>
<td>2.4</td>
<td>724</td>
<td>15.2</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td>2.4</td>
<td>804</td>
<td>48.1</td>
<td>1.3</td>
</tr>
<tr>
<td></td>
<td>2.6</td>
<td>886</td>
<td>46.4</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>2.7</td>
<td>910</td>
<td>98.7</td>
<td>1.4</td>
</tr>
<tr>
<td>Synapse</td>
<td>1.0</td>
<td>157</td>
<td>10.2</td>
<td>1.31</td>
</tr>
<tr>
<td></td>
<td>1.1</td>
<td>222</td>
<td>27.0</td>
<td>1.65</td>
</tr>
<tr>
<td></td>
<td>1.2</td>
<td>256</td>
<td>33.6</td>
<td>1.67</td>
</tr>
<tr>
<td>Lucene</td>
<td>2.0</td>
<td>196</td>
<td>46.4</td>
<td>2.9</td>
</tr>
<tr>
<td></td>
<td>2.2</td>
<td>248</td>
<td>58.1</td>
<td>2.8</td>
</tr>
<tr>
<td></td>
<td>2.4</td>
<td>341</td>
<td>59.53</td>
<td>3.1</td>
</tr>
<tr>
<td>Poi</td>
<td>1.0</td>
<td>237</td>
<td>59.5</td>
<td>2.4</td>
</tr>
<tr>
<td></td>
<td>1.5</td>
<td>315</td>
<td>11.8</td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td>2.0</td>
<td>387</td>
<td>64.4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2.5</td>
<td>443</td>
<td>65.6</td>
<td>1.8</td>
</tr>
</tbody>
</table>

We have selected the datasets which are having multiple releases to prove our approach. For each project we have considered LOC and CK metrics for as they are reliable and evaluated for cost–effectiveness in our approach. We have selected CK metrics as they are widely used for quality assessment of Object Oriented projects.

7. RESULTS

The following tabulated shows the result of \( P_{effort} \) scored by traditional Rregression Tree (RT) and Ga based RT. The \( P_{effort} \) achieved over 30 runs as well as corresponding standard deviation. From the results it is very clear that the GA based regression models significantly outperform their counterparts for predicting \( P_{effort} \).

Table 2: The average \( P_{effort} \) by RT model and GA trained RT model

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Training</th>
<th>Test</th>
<th>RT</th>
<th>RT-GA</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>St Dev</td>
<td>Mean</td>
<td>St Dev</td>
<td></td>
</tr>
<tr>
<td>Velocity</td>
<td>0.692</td>
<td>0.01</td>
<td>0.87</td>
<td>0.01</td>
<td>26%</td>
</tr>
<tr>
<td></td>
<td>0.45</td>
<td>0.03</td>
<td>0.73</td>
<td>0.09</td>
<td>80%</td>
</tr>
<tr>
<td>Xlan</td>
<td>0.43</td>
<td>0.006</td>
<td>0.81</td>
<td>0.03</td>
<td>47%</td>
</tr>
<tr>
<td></td>
<td>0.55</td>
<td>0.03</td>
<td>0.81</td>
<td>0.03</td>
<td>112%</td>
</tr>
<tr>
<td>Synapse</td>
<td>0.54</td>
<td>0.003</td>
<td>0.57</td>
<td>0.035</td>
<td>6%</td>
</tr>
<tr>
<td></td>
<td>0.64</td>
<td>0.017</td>
<td>0.68</td>
<td>0.037</td>
<td>35%</td>
</tr>
<tr>
<td>Lucene</td>
<td>0.39</td>
<td>0.024</td>
<td>0.53</td>
<td>0.025</td>
<td>36%</td>
</tr>
<tr>
<td></td>
<td>0.64</td>
<td>0.025</td>
<td>0.63</td>
<td>0.025</td>
<td>37%</td>
</tr>
<tr>
<td>Poi</td>
<td>0.46</td>
<td>0.025</td>
<td>0.63</td>
<td>0.025</td>
<td>37%</td>
</tr>
</tbody>
</table>

http://www.iaeme.com/IJMET/index.asp 7 editor@iaeme.com
Overall, as far as the effective resource allocation for maximizing the cost effectiveness our approach offers a stronger solution than the existing models.

8. CONCLUSION AND FUTURE WORK

In this paper we have hypothesized that the defect prediction may not reach its entitlement as they are trained on a task that is totally deferent from defect prediction cost-effectiveness. Current statistical based models are used to find the best fit to predict the artifacts having defects while our approach is to rank the artifacts based on defect density for most cost-effectiveness.

The proposed GA based approach is designed overcome the limitations of traditional methods so that artifacts exhibit more defects are given higher priority. Results from the Table (2) clearly suggest that the GA based approach is useful.

Future research need to investigate if our approach can be extended to other models such as Naïve Bayesian and Artificial Neural Networks as well as to other software metrics other than CK metrics. Further research is also required to study the influence of project size on the same.

REFERENCES

Cost Sensitive Software Defect Prediction Technique Using Genetic Algorithm


