Multivariate Logistic Regression Prediction of Fault-Proneness in Software Modules

Goran Mauša*, Tihana Galinac Grbac* and Bojana Dalbelo Bašić**
* Faculty of Engineering, University of Rijeka, Rijeka, Croatia
** Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia
goran.mausa@riteh.hr, tihana.galinac@riteh.hr, bojana.dalbelo@fer.hr

Abstract - This paper explores additional features, provided by stepwise logistic regression, which could further improve performance of fault predicting model. Three different models have been used to predict fault-proneness in NASA PROMISE data set and have been compared in terms of accuracy, sensitivity and false alarm rate: one with forward stepwise logistic regression, one with backward stepwise logistic regression and one without stepwise selection in logistic regression. Despite an obvious trade-off between sensitivity and false alarm rate, we can conclude that backward stepwise regression gives the best model.

I. INTRODUCTION
Considering the complexity of modern software products and the numerous constraints which follow its production, it is not unusual for the delivered software to have faults. Software quality models have the task to automatically predict fault prone modules and enable verification experts to concentrate on solving problem areas of the system under development. That is why applying software quality models in the early stages of software life cycle is essential. It provides an efficient defect removal procedure and results in delivering more reliable software products [1, 2].

Fault prediction modeling is an important area of research in software engineering. With overall testing costs estimated at 50% of entire development costs, testing consumes a lot of resources. Ideally, testing should be exhaustive in order to be confident that most faults are detected. In practice, however, due to many constraints, that is not possible and every additional save of resource is more than welcome. Fault prediction can be of assistance there, allowing software engineers to focus development activities on fault-prone code, improving software quality and making better use of resource.

Various techniques have been proposed for model building with logistic regression being among better ones. Among various statistical methods, machine learning methods, parametric models, and mixed algorithms, this paper explores the capabilities of logistic regression. The logistic regression has been recognized as one of the best methods used for fault-proneness prediction [1-5]. That is why the multivariate logistic regression is performed to build fault-proneness prediction model.

In this paper we are investigating following research questions:

1) Can we choose a smaller subset of independent variables in the fault prediction model using logistic regression to obtain better results?
2) Which static code attributes used as independent variables influence the model prediction performance?

Using too many independent variables can have negative effects on model’s fault-proneness prediction, making the model more dependent on the data set currently in use and therefore less general [6]. Selecting the appropriate measures to be used in the model requires a strategy of minimizing the number of independent variables in the model. This paper investigates the usage of static code attributes as independent variables and forward and backward stepwise selection principles in choosing variables. The model performances are evaluated using widely used performance measures such as accuracy, sensitivity and false alarm rate [1, 2]. A defect prediction model should identify as many fault prone modules as possible while avoiding false alarms [7].

A public domain NASA data set is used for building and testing the fault-proneness predicting model. The capabilities of logistic regression are tested on data set CM1, which contains certain parameters describing a program code written in C. The paper consists of following sections: the first section contains the description of whole case study process; the second section examines the potential threats to validity and the third section gives a conclusion based on conducted research.

II. CASE STUDY
A. Data set
CM1 is a public data set acquired from PROMISE (PRedictOr Models In Software Engineering) repository at http://promise.site.uottawa.ca/SERepository. The goal of PROMISE repository is to encourage repeatable, verifiable, refutable, and/or improvable predictive models in software engineering [8]. The CM1 data set’s creator is the NASA Metrics Data Program and the donor was Tim Menzies on December 2, 2004. The CM1 data is obtained from a spacecraft instrument, written in C, containing approximately 506 modules, structured as a matrix of 498 lines and 22 columns, where lines represent different software modules and columns represent different static code attributes. All but the last column describe the complexity of the software code and the last one gives the information weather there was a defect detected or not.
These data are used to evaluate the performance of the defect detecting models based on the Logistic regression. The 22 different code attributes from CM1 data set, along with its mean, standard deviation, minimum and maximum of each parameter calculated using Statistica are given in Table 1.

Halstead metrics mostly measure program size and time and effort put in to the process of programming. McCabe metrics, on the other hand, define complexity of the program code. Both metrics, along with total number and number of unique operators and operands and branch count help in identifying vulnerable code [5]. The last parameter is the one that tells us whether there was a defect in the software code. There are 449 software modules without defect and 49 modules with one or more defects.

As it is always recommended to plot the data, box and whisker plot was done as well. Box and whisker plot is able to graphically represent three different features. In large data sets, there is often the risk of misinterpreting descriptive statistic features such as range or mean due to the presence of outliers. That is why median is calculated instead of mean, and non-outlier range and interquartile range are calculated instead of range. As it can easily be noticed in table 1, the Halstead e parameter has much larger values comparing to other parameters and it stands in the way of appropriately examining all the parameters. Parameters Halstead t, Halstead v and Halstead n, though smaller in values than Halstead e, pose the same problem and that is why in the figure 1 all 4 parameters were omitted. On each box, the central mark is the median, the edges of the box are the interquartile range and the whiskers are the non-outlier range.

B. Logistic Regression

Logistic regression, in statistics, is a modeling method used for prediction of the probability of occurrence of an event. The independent variables are fitted to a logistic function and therefore the output can take on values between zero and one. Due to these reasons, it is suitable for building software classification models, especially for defect proneness prediction, because there are only two possible outcomes: either the software module is fault-prone or it is non-fault-prone. Logistic regression is related to some other statistical analysis techniques but it offers more flexibility and robustness [9, 10]. Logistic regression does not assume linear relationship between the
input and output variables, nor normal distribution and equal variance within input variables. The purpose of logistic regression is usually to model the probabilities of two different classes via linear functions in $x$, a posteriori. Moreover, it ensures the sum of probabilities equals 1 and all of them remain within usual range between zero and one, inclusive [11]. The model has the form:

$$\ln \left( \frac{\hat{Y}_i}{1 - \hat{Y}_i} \right) = u$$

(1)

where $\hat{Y}_i$ is the estimated probability within range $[0, 1]$ that the $i$-th case is in one of the categories and $u$ is the usual linear regression equation of $x$:

$$u = A + Bx$$

(2)

with constant $A$, coefficient $B$ and predictor $x$. With a simple calculation, this logit or log of the odds equation (1) gives the probability equation:

$$\hat{Y}_i = \frac{e^u}{1 + e^u}$$

(3)

The output of equation (3) is the probability of occurrence of an event and a cut-off percentage needs to be set in order to perform prediction based on given probability.

C. Multivariate Logistic Regression

In order to include more than just one predictor, a multivariate logistic regression model has to be used. Its functionality is based on equation (3) and performs according to equation:

$$\pi(X_1, X_2, ..., X_n) = \frac{e^{c_0 + c_1 X_1 + ... + c_n X_n}}{1 + e^{c_0 + c_1 X_1 + ... + c_n X_n}}$$

(4)

where $X_i$ are the numerous independent variables included in predicting model, $C_i$ are the regression coefficients of each independent variable $X_i$, and $\pi$ is the output probability. In our case, the independent variables are 21 features of software module and the output is probability weather the module is fault-prone.

Maximum likelihood is the procedure for estimating coefficients. The goal is to find the best linear combination of predictors to maximize the likelihood of obtaining the observed outcome values. Maximum likelihood estimation is an iterative procedure that starts with arbitrary values of coefficients for a set of predictors and determines the direction and size of change in the coefficients that will maximize the likelihood of obtaining the observed values. The residuals for the predictive model based on those coefficients are then tested and another determination of direction and size of change in coefficients is made. This procedure repeats until each coefficient converges to a steady value. In effect, maximum likelihood estimates are those parameter estimates that maximize the probability of finding the correct output data [5]. A great number of predictors that multivariate logistic regression tolerates does not always offer the best predicting performance. Including the predictors not related to the outcome may degrade that performance. One way to appropriately select predictors is to use stepwise selection procedure.

D. Stepwise Logistic Regression

Forward selection and backward elimination are the two main principles used in stepwise selection procedure. The general forward selection procedure initially has a model with the intercept coefficient only. Based on a certain statistical criteria, variables are selected one at a time for inclusion in the model, until a stopping criteria is fulfilled. The general backward elimination procedure, on the other hand, starts with a model that includes all the independent variables and, one at a time is to be removed from the model, until a stopping criteria is fulfilled. However, the two described principles are rarely used alone. Both forward stepwise and backward stepwise regression combine the principles of selecting and eliminating variables. The difference is that forward stepwise regression starts with only intercept included in the model and backward stepwise regression starts with all variables included in the model. Entry of a variable into the model is based on the p1-value, while removal of a previously entered variable is based on the p2-value.

The p-value is level of statistical significance. To assess the statistical significance of each independent variable in the model, a likelihood ratio chi-square test is used. Let $l$ be the log-likelihood of the model given in equation (4), and $l_i$ be the log-likelihood of the model without variable $X_i$. Assuming the null hypothesis that the true coefficient of $X_i$ is zero, the statistic $G = -2(l - l_i)$ follows a chi-square distribution and $p = P(\chi^2(l) > G)$ is tested. If p overcomes the usual level of significance of 0.05, the observed change caused by $X_i$ is not considered significant. However, if p is lower than mentioned level of significance, $X_i$ is considered to be significant and it becomes included in the logistic regression model [6]. In practice the p1-value is often set to 0.05 and the p2-value to 0.1.

E. Performance Evaluation Metrics

The most appropriate measure of overall fitness of an observed model is usually within interval from zero to one. Confusion matrix provides four scores that serve as basis for such an evaluation. A true positive (TP) score and a true negative (TN) score are counted for every correctly classified fault-prone module and non-fault-prone module, respectively. A false positive (FP) score and false negative (FN) score are counted for every misclassified non-fault-prone module and fault-prone module, respectively. Evaluation metrics used in this paper are:

- **Accuracy**, also known as precision as:

$$\text{ACC} = \frac{TP + TN}{TP + FP + TN + FN}$$

(5)

- **True positive rate (TPR)**, also known as sensitivity, rate of detection, hit rate or recall as:

$$\text{TPR} = \frac{TP}{TP + FN}$$

(6)

- **False positive rate (FPR)**, also known as false alarm rate or fallout as:

$$\text{FPR} = \frac{FP}{FP + TN}$$

(7)
Accuracy gives the percentage of accurately predicted modules in whole set of tested modules, evaluating the general appropriateness of the model. Sensitivity tells us the percentage of correctly classified fault-prone modules in whole set of faulty modules, indicating how many fault-prone modules we missed. False alarm rate shows the percentage of non-fault-prone modules that were misclassified as faulty in whole set of non-fault-prone, telling us the chance of focusing at dealing with fault that does not exist, which could prove to be a rather expensive mistake.

F. Research Method

Matlab 7.11.0. is used in research procedure. The procedure includes several steps:

1) Acquire the CM1 data set
2) Use descriptive statistics to examine the data set
3) Divide data randomly to learning and testing set
4) Apply forward and backward stepwise logistic regression to identify input variables
5) Build fault predicting models conducting the logistic regression upon the preselected parameters
   a) Build models on learning sample
   b) Test models on testing sample
6) Evaluate the results
7) Repeat from step 3) for iter times

We acquired the CM1 data set from (http://promise.site.uottawa.ca/SERepository/) and transferred it to Matlab where basic statistical features of each parameter in the data set were analyzed.

In order to automate the model building process we created a script in Matlab. The script is designed to choose independent variables according to forward and backward stepwise regression procedure, split the data set into a matrix of independent variables X and a vector matrix of dependent variable Y, to randomly split 67% of the data set as a learning test and to define the remaining 33% as a testing set, to include only the chosen parameters into the fault predicting model and finally to evaluate its performance.

Forward and backward stepwise regression were done using the stepwisefit function which by default performs forward stepwise procedure and needs the initial model that includes all of the parameters in order to perform backward stepwise procedure. When using both forward and backward stepwise model building p1-value is left at default value of 0.05 and the p2-value is left at 0.1. Maximum number of steps is left at default infinite value. The mnrfit function is used to calculate the coefficients of the multinominal logistic regression model. The matrix of independent variables from learning set is used as the first input variable and the vector matrix of dependent variable from learning set is used as the second input variable into the mnrfit function. It is important to notice that instead of using value of 0 for non-fault prone modules and 1 for fault-prone modules we had to increment their values to 1 and 2, respectively. That had been done in order to satisfy the mnrfit function demands (for example, if value of 3 had been used, the logistic regression model would try to fit the output data into 3 different categories). Coefficients are the output of the function and there is one coefficient more than the number of included variables. This is because mnrfit automatically includes a constant term in all models so it is not necessary to enter a column of 1s directly into X. Testing the acquired models is done in Matlab as well. The coefficients from the mnrfit function and the matrix of independent variables from testing set are the first and second input variables, respectively, of the mnrval function. The mnrval function computes predicted probabilities for the multinominal logistic regression model with chosen predictors and intercept and coefficient estimates returned by the mnrfit function. A cut-off percentage which determines if the program code is fault prone is set to the middle value, at 0.5. Evaluation metrics of accuracy, true positive rate and false positive rate are calculated separately for forward stepwise model, for backward stepwise model and for overall model (that includes all of the input variables). The whole process from randomly dividing sets to computing evaluation measures was repeated for 50 iterations.

G. Results

Before any stepwise logistic regression model bulding commenced, a simple modification had to be done. The Halstead e parameter that was problematic even for box and whisker plot due to its large range when comparing to other variables, proved to be problematic for stepwise regression as well. That is why it had been divided by a factor of 1000. That way the information this parameter holds is preserved and the problematic range is reduced.

While performing the forward stepwise and backward stepwise regression variable selecting process in Matlab, we counted how frequently each of the variables appears in chosen subset. Frequencies were counted in forward and backward stepwise regression procedure separately and along with their sum, they are given in table 2. With entrance p1-value set to 0.05 and exit p2-value set to 0.1, table 2 indicates some of the parameters that should be included in the logistic regression model and some that should be avoided as well. We can see that Halstead total operators + operands, Halstead "program length", Halstead "difficulty" and Halsteads count of blank lines have been included in the model for less than 10 out of 100 times. Since both forward and backward regression procedures often reject them, we can conclude these parameters, along with total operands, hold almost no information for fault prediction. On the other hand, Halstead count of lines of comments has been included almost every time, unique operators has been included in over 80% of the times, Halstead "intelligence", Halstead "effort" and LOCcode And Comment are selected approximately 50% of the times, indicating these variables are significant in fault predicting models.
Table 2  FREQUENCY OF CHOOSING EACH OF THE INDEPENDENT VARIABLES WITH STEPWISE LOGISTIC REGRESSION

<table>
<thead>
<tr>
<th>Method Level Features</th>
<th>Name</th>
<th>Forward</th>
<th>Backward</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>McCabes line count of code</td>
<td>11</td>
<td>18</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>McCabe &quot;cyclomatic complexity&quot;</td>
<td>6</td>
<td>37</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>McCabe &quot;essential complexity&quot;</td>
<td>14</td>
<td>8</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>McCabe &quot;design complexity&quot;</td>
<td>8</td>
<td>34</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>Halstead total operators+operands</td>
<td>0</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Halstead &quot;volume&quot;</td>
<td>2</td>
<td>16</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Halstead &quot;program length&quot;</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Halstead &quot;difficulty&quot;</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Halstead &quot;intelligence&quot;</td>
<td>18</td>
<td>36</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>Halstead &quot;effort&quot;</td>
<td>18</td>
<td>34</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>Halstead</td>
<td>4</td>
<td>23</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Halsteads time estimator</td>
<td>11</td>
<td>24</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>Halsteads line count</td>
<td>9</td>
<td>11</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Halsteads count of lines of comments</td>
<td>48</td>
<td>48</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>Halsteads count of blank lines</td>
<td>1</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>LOCode And Comment</td>
<td>18</td>
<td>29</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>unique operators</td>
<td>33</td>
<td>48</td>
<td>81</td>
</tr>
<tr>
<td></td>
<td>unique operands</td>
<td>2</td>
<td>28</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>total operators</td>
<td>0</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>total operands</td>
<td>1</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>branchCount of the flow graph</td>
<td>19</td>
<td>17</td>
<td>36</td>
</tr>
</tbody>
</table>

Table 3  EVALUATION AVERAGE RESULTS

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy (ACC)</th>
<th>Sensitivity (TPR)</th>
<th>False alarm rate (FPR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall (no selection)</td>
<td>0.852</td>
<td>0.927</td>
<td>0.834</td>
</tr>
<tr>
<td>Forward selection</td>
<td>0.883</td>
<td>0.971</td>
<td>0.929</td>
</tr>
<tr>
<td>Backward selection</td>
<td>0.880</td>
<td>0.963</td>
<td>0.875</td>
</tr>
</tbody>
</table>

After parameters selection had been done, the learning and testing procedure was performed. The script repeated from the 3rd step of the research procedure, described in previous subsection, 50 times over. Each time a different randomly chosen learning and testing data sets were extracted. In the end, the mean value of each evaluating metric was calculated. Performing such logistic regression model building, testing and evaluating process in Matlab, we obtained results given in figure 2.

Table 3 presents the average values of accuracy, sensitivity and false alarm rate of 50 iterations for model that includes all parameters, model that includes the parameters suggested by forward stepwise procedure and model that includes the parameters suggested by backward stepwise procedure.

Examining figure 2 and table 3, we notice that all three models have high score of accuracy and false alarm rate and even higher score of sensitivity. In term of accuracy the forward stepwise and backward stepwise selecting model have the best results, but the difference between the three models is insignificant. When it comes to observing sensitivity vs. false alarm rate, the trade-off between the two is obvious. Having higher sensitivity score comes with higher false alarm score and vice versa. False alarm rate offers more diversities and therefore easier comparison. Forward stepwise selecting model has worst results, while the other two models have similar results, with backward stepwise selecting model having chance of achieving lowest false alarm rate.

Fig. 2  BOX AND WHISPER PLOT OF EVALUATION RESULTS FOR 50 ITERATIONS
III. Threats to Validity

Every case study should be aware of potential threats to validity. Each stage of a case study research is prone to such threats and they should be pointed out [12]. In this research bias can be detected in choice of data set, choice of method for fault-proneness prediction model building method and choice of evaluation metrics.

We were aware that selection of data set can lead to bias of results and conclusions. That was one of the reasons for choosing the verified and trustworthy NASA PROMISE repository. Lack of data quality, a problem often present in data sets of unchecked origin was overcome. Another advantage of PROMISE repository is its public accessibility, which makes our results open to verification by the whole research community. However, limiting ourselves to only one company as source of data offers another threat to validity and we plan to include a greater number of data sets in our future work. The fact that only a small number of only 49 software modules with defects is present in the CM1 data set, has to be taken into consideration of threats to validity as well. Our decision to randomly split the data set into learning and testing set also leaves room for bias. In order to minimize that threat to validity, we conducted research process with 50 iterations. The mere choice of stepwise logistic regression method as means for finding which software modules’ features could be indicated as best suitable for defect proneness prediction is another threat to validity. Including other prediction and variable selection methods is planned for our future research. The evaluation metrics in this problem area are numerous and the once we used are one of the most commonly found in similar researches. They are also rather self-explanatory and offer a valuable insight into models’ performance. Inclusion of AUC as even more general evaluation metric is expected in future.

IV. Conclusion

In this paper, we investigated the capabilities of logistic regression based fault prediction model. An empirical comparison of two different approaches to selecting independent metrics for fault-proneness prediction model has been reported. Logistic regression was the tool and a public domain development data set CM1 from NASA PROMISE repository was the object of research. Three different models have been compared in terms of accuracy, sensitivity and false alarm rate: the one that includes all the independent variables, the one that includes only the independent variables suggested by forward stepwise logistic regression and the one that includes only the independent variables suggested by backward stepwise logistic regression.

The level of accuracy in all the models confirmed their general appropriateness in defect predicting task. The sensitivity is even higher in all the models, which is also advisable. The level of false alarm rate, on the other hand, is too high in all the models while it should be otherwise. It is difficult to say which model has best results because the differences are subtle. Forward model, however, could be indicated as one that achieves best results in sensitivity, but with too great a cost in high false alarm rate.

Backward stepwise selecting model has equally good results in accuracy as forward stepwise selecting model, better results in sensitivity than the overall model and, though in rare cases, it even achieves the lowest false alarm rates. These conclusions that we draw from the results have to be taken in respect to the threats to validity described in previous section.

Despite the limitations the chosen data set presents, we conclude that the backward stepwise regression proved to have great potential in choosing a smaller subset of independent variables for better results of logistic regression fault prediction model. We also found which static code attributes used as independent variables influence the model prediction performance. There are 2 features that must not be excluded from the model: Halstead count of lines of comments and unique operators and there are several features that can easily be omitted: Halstead total operators + operands, Halstead "program length", Halstead "difficulty", Halsteads count of blank lines and total operands.

REFERENCES