Journal of Computers for Society
Journal homepage: https://ejournal.upi.edu/index.php/JCS
Vol. 1 No. 1, June-2020, pp. 24-34

Software Bug Prioritization in Beta Testing Using Machine Learning Techniques

Anum Waqar[a]

[a] *Department of Computer Software Engineering, UET Mardan, Pakistan*
anum.nwfpuet@yahoo.com[a]

Abstract

Testing in Software Development Life Cycle is one of the most crucial activities. Bug prioritization has been a manual process for long. Our paper provides a methodology for ease of bug prioritization in beta testing phase. In the methodology, data from various bug reports is supplied into a model and, through machine learning, the model outputs fairly accurate bug priority based on historical data.

Keywords: Bug prioritization, machine learning, linear regression, supervised learning

First draft received: May 21, 2020
Date Accepted: June 16, 2020
Final proof received: June 21, 2020

1. Introduction

During the last phase of software development lifecycle, just before the release, many software products undergo Beta Testing. Beta testing is a kind of User Acceptance Testing (UAT) which is performed to validate the business requirements [1]. UAT is the final stage of testing which is predominantly performed by Users/Clients/Customers before the software product is delivered. Beta testing is performed in the production environment. The software product is made available to the users. Users then have a certain amount of time during which they can use this product and get back to the developer in case of issues. Since beta testing is performed at user end, usability, efficiency, productivity and accuracy also get checked, hence, encompassing a larger range of requirements.

Normally, the bugs encountered are being prioritized manually. This is a cumbersome job and requires a lot of efforts. It is observed that dealing with bug priority quickly saves a lot of efforts and time since many bugs reported are not even bugs, or may be enhancements that could be worked on later. A solution to this issue is to automate the bug prioritization process. We have used Machine Learning technique to build a model capable of correctly calculating the priority of bugs provided enough input data. The following two questions have been addressed in this research:

i. Which procedure needs to be followed for calculating bugs priority that come across in Beta Testing phase?

ii. What are the most relevant parameters which estimate the priority of bugs?

We have used Multiple Linear Regression [2] which falls under the umbrella of Supervised Learning [3]. Linear regression, being a type of supervised learning, needs to be provided with training data set and correct answers. Machine learning algorithm infers knowledge and thus learns. We have provided some facts to our model which ensures the accurate predictability of bug priority. We have used Forward Selection which makes certain that just those predictor variables whose inclusion in the model will yield productive results are included in the model.

Rest of the paper is organized as follows. Section 2 is focused on methodology and technique, Section 3 and 4 present results and conclusion, respectively.

2. Methods

For performing the research, bug data have been collected from Eclipse projects. We have collected bugs containing columns namely “Bug ID”, “Product” in which bug was found, “Component” containing bug, “Assignee”, “Status” whether Open, Close etc, “Classification”, “Operating System”, “Resolution”, “Reporter” of the bug, “Hardware”, brief “Summary”, “Changed (Date)”, “Version”, “Priority” and “Severity”. Here, “Priority” is the response variable Y and others are taken as predictor variables X_1, X_2 … X_n. “Priority” can take five forms namely *Priority 1*, *Priority 2*, *Priority 3*, *Priority 4,* and *Priority 5*. *Priority 1* points to the most important bug.

The equation of Multiple Linear Regression then becomes [4] (see **Eq. 1)**

$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + e \quad (1)$$

In **Eq. 1**, β_0, β_1 and so on show regression coefficients for input variables x_1, x_2 … x_n and Y is the output or required variable.

In our research, we have collected year wise data. We have gathered bugs between years 2005 and 2017. The data is then assembled in tabular form. This can be done using either through some tool like Excel or can be put in Minitab projects. We have first filled excel sheets during our study. Our gathered data can be seen in **Table 1**.

Table 1. *Gathered data for our study.*

Bug Id	Product	Component	Assignee	Status of bug	Summary	Date of Change	Operating System	Priority
486823	JDT	Core	jdt-core-inbox	NEW	NPE logged as warning,error reporter does not pop up	29.1.16	PC Linux	3
468307	Platform	UI	Platform-UI-Inbox	NEW	[Preferences] Print margin's behavior is confusing to many users	27.1.17	All All	3
423715	Equinox	p2	mn	ASSIGNED	Add SHA256 to p2 metadata publishing (and prefer for consumption if available)	19.1.17	All All	3
366471	Platform	SWT	lshanmug	NEW	[Cocoa] Slow scrolling in editor on Mac OS X	09.2.17	PC Mac OS X	2
500758	Virgo	runtime	virgo-inbox	NEW	Virgo kernel (services) timeout intermittently	25.1.17	PC All	3

Further processing of these tables is done through a statistical software tool known as Minitab where we have applied Multiple Linear Regression. Minitab treats the data in the form of worksheets. Data can be input in multiple ways into Minitab project namely (i) Numeric Data or numbers, (ii) Textual data that may contain alphabets, special characters and (iii) spaces or Date/Time data.

Multiple Linear Regression algorithm is executed on the input data afterwards in Minitab. Certain configuration details are given to the software such as corresponding response which is Priority in our study. Continuous as well as categorical variables are also been highlighted. We have in our study assigned integer values to various priority levels. The sole reason for this is that Minitab can only take response variables as continuous. These integer values are as follows. *Priority 1* is assigned 1, 2 for *Priority 2, 3* for Priority level 3, *Priority 4* equals 4 and 5 represents *Priority 5*.

Our methodology in this study focuses on Forward Selection. Although it does count which methodology is being used by researchers, it is also vital that the study inspects individual variables so that the significance of variables be calculated. Also, it should be made sure that just those variables which have considerable significance on variance be included. A stepwise procedure should be followed where insignificant variables are excluded from the equation one after the other. Also, the equation must be calculated again after each insignificant variable is opted out. Our research uses Least Square Method for determining a line of best fit by trimming down the sum of squares where as a square is calculated by squaring the distance between a point on the plot and the regression line [5, 6].

Null Hypothesis: The null hypothesis states that the coefficient of predictor variable is 0. It implies that there is no association connecting the response and the predictor. The alpha value is 0.05 in our study. In our study the p-value has been kept as 0.05 for each coefficient of the predictor variable. Where the p-value is calculated to be <= alpha, we have considered the null hypothesis invalid.

3. Results and Discussion

Following are the results we got by applying Least Square method

Least Square Results-According to [7], range of *R-sq* lies between is 0 and 100%: a value equals to 0% is a sign that the model is not capable of explaining any of the variability of the response data with respect to mean. Whereas a 100% value symbolizes that the model encapsulates all the variability of the response data with respect to mean. In a broader spectrum, for higher values of *R-sq*, the model better fits the data.

Table 2. *Results for Least Square.*

S	R-sq	R-sq (adj)	R-sq (pred)
0.442041	85.16%	45.32%	*

We achieved in our study *R-sq* value equal to 85.16% as shown in **Table 2**. It is known that a priority of bugs varies. So, *R-sq* basically lets us know that 85.16% of the changeability in the priorities is clarified by the changeability in independent variables considered in the research. It implies that in our model, some of the invariability is still vague when it comes to the variation in priorities. There has to be some other explanation in addition to the predictor variables taken into account in this model that clarify the variability in response variable. The left over 14.84% is present in the residuals. The residuals are thought to be the vague portion of the model.

Standard Error of Regression, also known as Standard Error of the Estimate, corresponds to the average distance of observed data points calculated to the mean i.e. the regression line. The process followed was that actual values were shown on graphs as plots. A regression line was then drawn through them. Regression process then evaluates the estimated values. A comparison was made afterwards between actual and estimated values. This comparison was basically the calculated distance between the two values. This calculated distance then represents the Standard Error of Regression. Lesser values of Standard Error of Regression are considered better. Lower values illustrate the nearness between observed values and the fitted line [7].

Results of the predicted response-**Table 3** show the Priority column and the FIT column which are the actual and the estimated values respectively. Closeness of values is clearly evident in the Table between these two. The actual priority of bugs is presented in column “Priority” which was obtained from bug database of Eclipse projects. The “Fit” column in **Table 3** displays the predicted bug priority after

calculation through Machine Learning algorithm. As seen in the table, the difference, shown by "Resid" column, between predicted and actual priority is minimal. This shows the applicability of our model in case of bug prediction.

Table 3. *Actual and Predicted Price.*

Obs	Priority	Fit	Resid
1	3.000	3.000	0.000
2	3.000	3.000	0.000
6	3.000	3.000	0.000
10	5.000	5.000	-0.000
11	5.000	5.000	0.000
12	5.000	5.000	0.000
13	5.000	5.000	-0.000
14	5.000	5.000	-0.000
15	3.000	3.000	-0.000
18	3.000	3.000	0.000
21	3.000	3.000	0.000

Result of prediction interval and confidence interval-Predictions on new data can be done now by giving values as inputs in the model. Result of this activity gives some statistics like Predicted Interval (*PI*) and Confidence Interval (*CI*). These can be seen in **Table 4**.

Table 4. *PC & CI.*

Variable	**Setting**		
Status	**NEW**		
Reporter	**edward**		
Hardware	**PC**		
Fit	**SE Fit**	**95% CI**	**95% PI**
3	0.442041	(2.12244, 3.87756)	(1.75894, 4.24106)

The predicted priority of newly gathered data is shown in the column "Fit". In the table, column SE Fit represents Standard Error value of the Fit. It is the deviation in the anticipated mean response for a certain type of setting of independent variables.

In the testing phase of the model proposed, the results of prediction can be verified by reading the values presented under these columns. The prediction value for some new bug will most likely to fall inside the interval called *PI* (Prediction Interval). *CI*

(Confidence Interval) gives the range in which the mean value is likely to fall within. The range covered by *PI* is greater than that of *CI*. This is due to the uncertainty occurring in prediction of a particular variable which isn't the case with measuring mean value.

***Results of regression plots*-Figure 1**, **Figure 2** and **Figure 3** are the results of passing data through the regression model. These graphs, in combination with some additional information, tell how much normally distributed are the residuals. In **Figure 1**, Frequency in Residual Histogram for Priority graph is the elevation of bars in the plot. Frequency is the measure of the number of observations. Density represents the area of each bar. It tells the quantity of the model observations. Most residuals lie between +0.75 and -0.75 as can be seen i.e. the graph is denser in the middle. 68% of the area of normal distribution is within one Standard Deviation of mean. 0 represents the mean. The histogram is normally spread around it. The mean is also known as the z-score. The distribution around mean indicates that majority of the residuals of the gathered data have been around the mean value of the residuals.

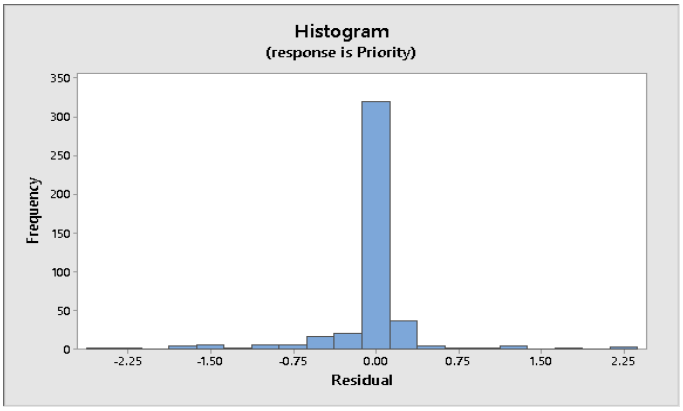

***Figure 1**. Frequency in Residual Histogram for Priority graph.*

In **Figure 2**, Residual symbolizes the error term while percent symbolizes the percentage of each observation computed after applying the model. As can be observed, most of the blue dots are crowded together about the red line. The curve is symmetrical about the mean. As there is linear relationship between Residual and percent, it can be

deduced that the residuals have a normal distribution. So, the postulation of normality is valid.

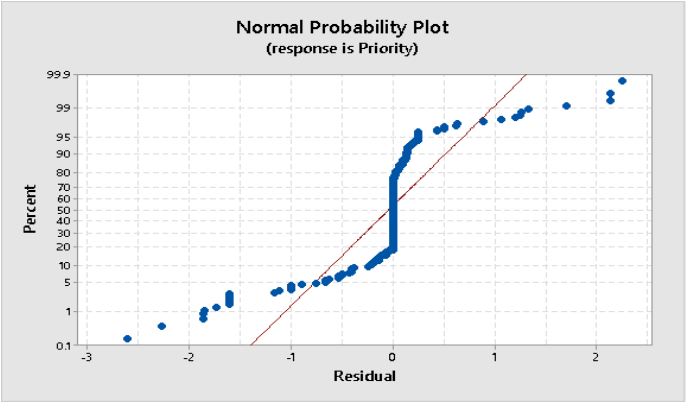

Figure 2. *Normal Probability Plot.*

In **Figure 3**, Observation Order represents the sequence of input data. Some additional information is being provided in this figure. Negative serial correlation is said to exist in plots where positive error in a single observed data unit add to the probability of a negative error in some other observed data unit and otherwise as well. As observed, there is negative serial correlation in Residual vs. Order for Priority graph. It can be deduced from these observations that the bug priority is most probably time bound. If we rephrase the sentence, it can be said that the priority has varied over years. These variations could be the result of variation in certain independent variables over the years. Some comparison can be done by investigating results produced by [8]. Moreover, others methods in machine learning, such as [9, 10, 11] can conducted to do some comparison analysis in more comprehensively. The improvement in term of computational time can be achieved by considering methods in parallel computing and Big Data, such as in [12, 13, 14].

4. Conclusion

An R-square value of 85.16% has been attained in our research. We can infer from this that our model is capable of explaining 85.16% of the variance in Priority which is the response variable. The left over portion which is 14.84% wasn't explained by the model. A possible reason for this lack of clarity could be some missing independent variables in the model. Due to which 100% variation in Priority could not be predicted. This can serve as food for future work.

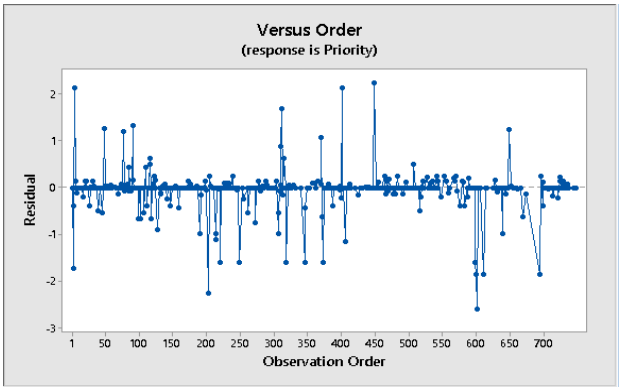

***Figure 3**. Residual vs Order.*

Through the Histogram and Normal Probability Plot, it is apparent that our data s normally distributed. On the other hand, rather strange fact has been observed in the Residual vs. Order graph. This plot contains some negative serial correlation. This could mean that the data may be dependent on time factor as well. Time dependency could be the reason of variation in independent variables with respect to time.

Our research shows that Multiple Linear Regression for bug prioritization is a good option for prediction. The model proposed by our research is suitable for prediction in bug priority as seen by the statistics observed. We have used Forward Selection which eliminates insignificant Predictors successfully.

Another future step for research can also be finding association between bug reports and time frame. Extension of our research can also be advanced in an area where more independent variables and their effect on response variable i.e. bug priority can be

assessed. Apart from broad-spectrum applications, our research can be narrowed down to extracting models for special purpose applications and their bugs.

References

[1] Hambling, B., & Van Goethem, P. (2013). *User acceptance testing: a step-by-step guide*. BCS Learning & Development.

[2] Andrews, D. F. (1974). A robust method for multiple linear regression. *Technometrics*, 16(4), 523-531.

[3] Alpaydin, E. (2020). *Introduction to machine learning*. MIT press.

[4] Kutlubay, O., & Bener, A. (2005). *A machine learning based model for software defect prediction*. Working paper, Boaziçi University, Computer Engineering Department.

[5] Yang, B., & Balanis, C. A. (2006). *Least square method to optimize the coefficients of complex finite-difference space stencils*. IEEE Antennas and Wireless Propagation Letters, 5, 450-453.

[6] Singh, Y., Bhatia, P. K., & Sangwan, O. (2007). A review of studies on machine learning techniques. *International Journal of Computer Science and Security*, 1(1), 70-84.

[7] Frost, J. (2014). *Regression Analysis: How to Interpret S, the Standard Error of the Regression*. The Minitab Blog. http://blog.minitab.com/blog/adventures-in-statistics/regression-analysis-how-to-interpret-s-the-standard-error-of-the-regression

[8] Nazir, S., Shahzad, S., & Riza, L. S. (2017). Birthmark-based software classification using rough sets. *Arabian Journal for Science and Engineering*, 42(2), 859-871.

[9] Riza, L. S., Janusz, A., Bergmeir, C., Cornelis, C., Herrera, F., Ślezak, D., & Benítez, J. M. (2014). Implementing algorithms of rough set theory and fuzzy

rough set theory in the R package “RoughSets”. *Information Sciences*, 287, 68-89.

[10] Riza, L. S., Nasrulloh, I. F., Junaeti, E., Zain, R., & Nandiyanto, A. B. D. (2016). gradDescentR: An R package implementing gradient descent and its variants for regression tasks. In *2016 1st International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)* (pp. 125-129). IEEE.

[11] Mulyani, Y., Rahman, E. F., & Riza, L. S. (2016). A new approach on prediction of fever disease by using a combination of Dempster Shafer and Naïve bayes. In *2016 2nd International Conference on Science in Information Technology (ICSITech)* (pp. 367-371). IEEE.

[12] Riza, L. S., Pratama, F. D., Piantari, E., & Fashi, M. (2020). Genomic repeats detection using Boyer-Moore algorithm on Apache Spark Streaming. *Telkomnika*, 18(2), 783-791.

[13] Riza, L. S., Rachmat, A. B., Munir, T. H., & Nazir, S. (2019). Genomic Repeat Detection Using the Knuth-Morris-Pratt Algorithm on R High-Performance-Computing Package. *Int. J. Advance Soft Compu. Appl*, 11(1), 94-111.

[14] Riza, L. S., Dhiba, T. F., Setiawan, W., Hidayat, T., & Fahsi, M. (2019). Parallel random projection using R high performance computing for planted motif search. *Telkomnika*, 17(3), 1352-1359.