Paper 077-2007

Predicting Software Outcomes Using Data Mining and Text Mining

Uzma Raja, University of Alabama, Tuscaloosa, AL
Marietta J. Tretter, Texas A&M University, College Station, TX

ABSTRACT

Organizations spend a major portion of their Information Technology budget on software maintenance. In this paper, we present a predictive model for the maintenance outcomes of the software projects. We also identify the factors that affect software maintenance outcomes. We build our model using Data Mining (DM) techniques on Open Source Software (OSS) project data. We use the public access to the data archives of over 100,000 projects hosted by SourceForge.net (SF). We use prior research in software engineering to identify the initial set of variables used in the model building process. We use multiple DM techniques available in SAS® Enterprise Miner™. We also create additional new variables from the textual data provided by SF, through SAS® Text Miner. The use of these new variables improves the model, significantly. The final model is selected based on domain knowledge and fit statistics of the models. Results indicate that end-user participation, product functionality, and usefulness of the project affect the software maintenance quality.

INTRODUCTION

Software maintenance accounts for a major portion of software life cycle costs. Researchers and practitioners have been in search of models to explain the factors that affect the outcome and quality of software maintenance tasks (Bennett and Rajlich 2002). Software maintenance tasks typically are related to detection and removal of errors. Much of maintenance costs are incurred on removal of errors. The unavailability of the software system due to occurrence of errors also accounts for a significant maintenance cost. In large organization, downtime for maintenance can translate into huge losses. Considering the impact of error detection and removal on software maintenance outcomes, it is considered an indicator of software maintenance quality. Software project that can efficiently detect and remove errors are considered high quality software systems.

Recently, there has been a shift towards the use of OSS projects. These projects are free to use and no costs are associated to acquiring them (Cearly, Fenn et al. 2005). They are developed through online community of volunteer programmers. The users are free to download the source code of the project. They can make changes/ updates at their own end, or make a request to the project team to make changes and add functionality. The free availability of OSS projects and interactive development teams makes them a very attractive choice for software users. Recent studies indicate that there has been an increase in the development and use of OSS projects. OSS projects are developed through a different methodology and philosophy compared to traditional software development. It is therefore important to study these projects and develop a better understanding of this new phenomenon. The differences in development and maintenance methods of OSS also encourage researchers and practitioners to develop new models that explain the behavior of OSS projects.

As mentioned earlier, the majority of software lifecycle costs are spent on the maintenance of software projects, not their development. Before companies adopt OSS projects, they need to evaluate these projects based on their operational performance and costs. OSS project maintenance is still being explored. It is important to examine the factors that impact software maintenance outcomes of these projects. Although OSS projects are free to use, the cost of maintaining such systems could be considerable, if they are of poor quality. Occurrence of errors in software systems is expected, but inability to remove these errors can translate to major IT expenses. We develop a model to predict the quality of an OSS project to the errors that occur during maintenance phase.

In this paper we develop a predictive model for OSS project maintenance outcomes, namely maintenance quality. In the next section we explain the model building process. We explain the dataset and variable selection process. Final model is selected based on fit statistics and domain knowledge. In the following section we discuss the final model that was selected. We explain the results and later conclude with future research directions.

MODEL BUILDING

There are several measures of software project quality used in prior research. We focus on maintenance costs and effort. Whenever an error occurs in software, a certain amount of time is needed to correctly identify, isolate and remove the error. The longer it takes to recover from occurrence of an error, the higher will be the costs associated to software maintenance. We therefore use the Mean Time to Recover (MTTR) from an error as a measure of the maintenance quality of an OSS project. Lower MTTR reflects that the project team responds immediate to the errors and removes them promptly. Such projects would be more desirable in corporate environment since this would account for less operational costs. On the other hand a high MTTR reflects that once an error occurs, either the team is slow to response, or the software quality is such that it takes longer to remove errors. In either case, the delay in removal of errors translates to lower maintenance phase quality.

OSS projects are developed online and the source code is publicly available. A significant amount of information on project development and maintenance is also accessible for these projects. We used the Source Forge.net repository to build our model. SF is the world's largest OSS project hosting community. It hosts over 150,000 projects and has over a million registered users. We extracted the data from a public repository of SF projects (Madey 2005). SF data warehouse has over 100 tables and thousands of variables. We studied the available information and used software engineering literature to identify the variables that could be useful in model building. Some new variables had to be cared from the available information. We used SQL queries to extract information from the warehouse.

We imported the data to SAS Enterprise Miner 5.3. The OSS projects vary in size and focus. Some projects maintain detailed data on all aspects of projects while others have very little data archives available. To build effective models it is critical to have a clean dataset with relevant variables. Therefore we examined the initial dataset for missing values. We identified the projects that were inactive and could not be used for analysis. OSS project hosting is free therefore a large number of project registrations were found to be inactive. Most of these projects either never started or became inactive after lack of volunteer participation. We removed all such projects from our datasets.

The next step was to separate the projects that were under development and the ones that were already complete and were operational. Software maintenance phase refers to the operational phase of software project lifecycle. Much of corporate use of OSS projects is restricted to mature and completed projects. Using the identifiers for project lifecycle phase, we extracted the projects in maintenance phase form the entire dataset.

Before model building, we had to complete the data extraction for all the variables needed. Using the data description of SF tables we identified a list of 36 variables from software engineering and OSS literature. We validated our data extraction against an independent third party data extraction, to ensure that we used the correct queries. We had to remove the projects that did not have an error-reporting database. In absence of error reporting data, it is impossible to compute the MTTR, which is our target variable. The resulting dataset had 4965 observations.

We explored the data and examined the distributions of all the variables. We also examined the distribution of the target variable MTTR. We created a binary variable reflecting MTTR, reflecting high or low MTTR. We selected a cutoff point based on the distribution to make the outcome binary (True/False). The data indicated that there was a cutoff at MTTR of one day. Therefore projects with MTTR less than one day have a low MTTR, i.e. high quality and vice versa. A high MTTR would indicate poor quality, since the project takes longer to remove errors occurring in it. On the other hand a low MTTR implies that errors are removed promptly and the software system recovery time is small.

Some of the variables were time dependent e.g. number of downloads. To account for the age of a project, these variables had to be normalized. We used the transformation node make needed transformations and to create new variables (e.g. defect density = SLOC / Errors). We also created the target variable. The cutoff value as suggested from the analysis using insight node was used to create the target variable. Projects with MTTR higher than the cutoff were labeled as "High" and the MTTR was coded as 1. The projects with MTTR lower than the cutoff were labeled as "Low" and coded as 0. Note that High and Low values are for MTTR, which is the inverse of maintenance quality, indicating that a project with a high MTTR has a low maintenance quality. The distribution of MTTR is shown in Figure 1. It can be seen that only 9.55% of the projects had a low MTTR i.e. high quality. The rest of the projects with high MTTR were categorized as low quality projects.

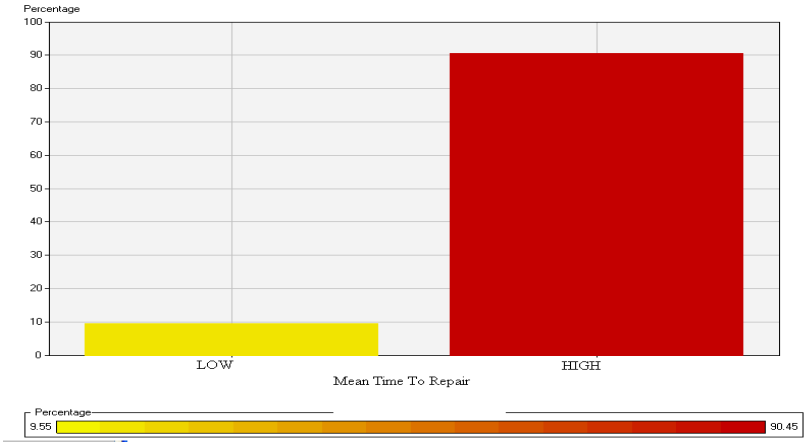

Figure 1: Distribution of MTTR

Besides the available variables and the computed variables form the SF data, we also wanted to categorize projects according to their type. SF projects vary in type and focus. There is categorization information available for projects, but it allows them to have multiple types. The issue of a single project belonging to multiple categories is not desirable in model building. We therefore decided to extract information about the type of project form the textual information available for each project. SF maintains a 200-word description for each project. This description explains the purpose of project, its target audience and its functionality. We performed text mining of the project description data for the OSS projects.

The dataset of project description for each of the project in its development phase was used in SAS Text Miner to create the new variable, called “project-type”. Initially the default stop list was used on the dataset. The stop list contains the words that are ignored while the text analysis is performed. The default list contains most common occurring words that do not carry information regarding the text being analyzed. The initial run with the default stop list provided with a word frequency table. For such a large amount of data, performing an initial run with default stop list is beneficial. A new start list was created by removing the words that were not considered a project description or added no usefulness to the analysis e.g. frequent words like where, upon or abbreviation like en, dl etc. The new list with “keep terms” is saved as a new start list and a final analysis is performed based on this list.

The Text Miner node was set to cluster the terms automatically. The option to generate the Singular Value Decomposition (SVD) terms and perform clustering based on the SVD dimension was selected. A maximum number of 40 clusters were allowed. The term stemming option was set to “Yes”. The frequency weighing method was “Log” and the term weighting method was “Entropy”. The expectation maximization algorithm was used for clustering. This algorithm is best suited in cases where the expected number of categories is unknown.

The observations were classified into seven clusters. The resulting clusters and the descriptive terms along with percentages and frequencies are shown in Table 1.

Cluster	Percentage	Freq	Descriptive Terms
1	0.150252	746	+ library, c++, + class, python, + support
2	0.035851	178	+ game, + player, + play, game, + base
3	0.091037	452	+ file, into, + will, + program, + image
4	0.082578	410	php, + easy, mysql, web, + database
5	0.090634	450	+ framework, development, + application, java, web
6	0.338771	1682	+ server, + client, + allow, + have, + tool
7	0.210876	1047	+ code, + source, + project, java, + base

Table 1: Description terms, frequency and percentage of each cluster for project type

The textual data split into seven clusters. Cluster #1 contains terms associated to programming languages e.g. C++ and python. Cluster #2 is associated to games, Cluster #3 contains terms related to file and image programs, Cluster #4 has terms related to databases, Cluster #5 had terms related to JAVA and web applications, Cluster #6 has terms related to networking while Cluster #7 has terms related to general OSS projects. The segment profile of the clusters is shown in Figure 2. We can see that 33.9% of the OSS projects on SF are client server applications, 21.1% are JAVA and web applications and 15% projects are mostly programming languages related projects. This gives us a new look at OSS projects in general and on what kind of projects is more popular in OSS domain. The purpose of using textual information was to improve the predictive power of our final model with this additional information.

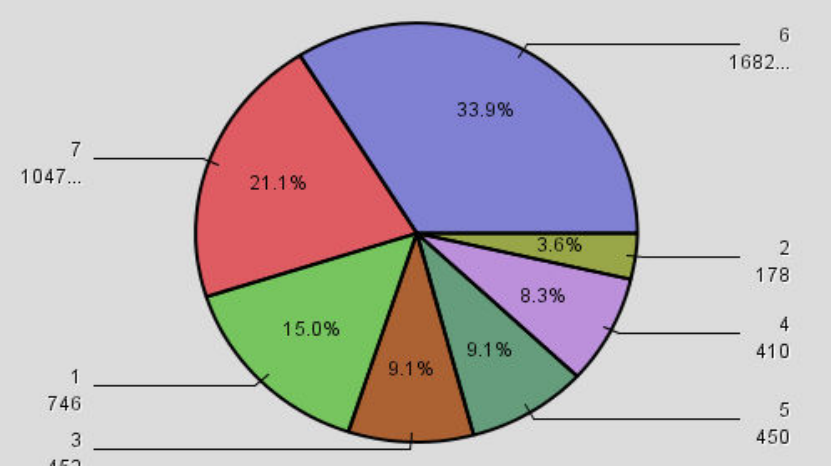

Figure 2: Segment Profile of Clusters for project type

The results of the text mining placed every project into one cluster. Each cluster identifies the project category it belongs to. We combined this information with all the other quantitative variables for each project through the merge node. The resulting dataset was now ready for model building using Data Mining. The process flow diagram for the model building process is shown in Figure 3.

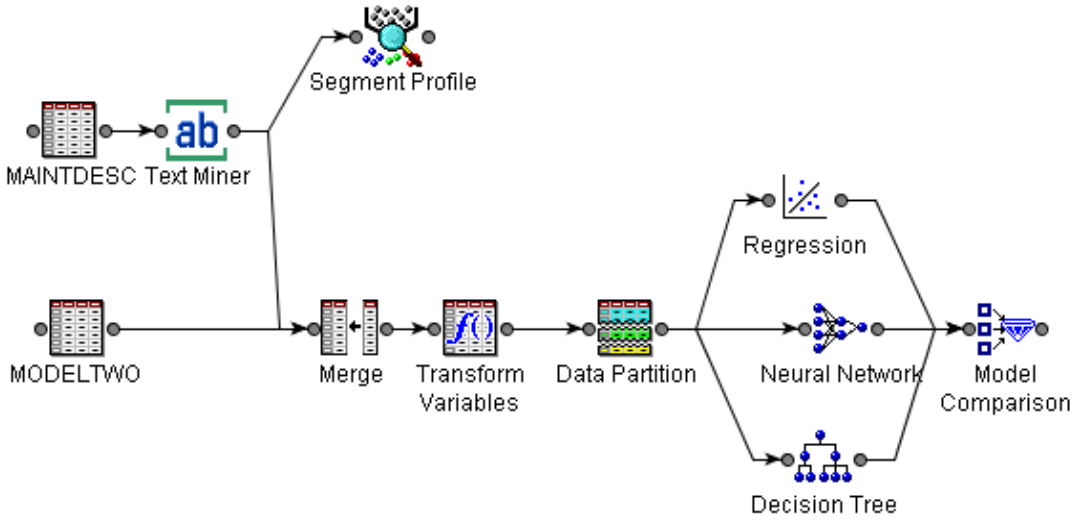

Figure 3: Process Flow diagram

Data Partitioning is very important step in Data Mining. We used the Sampling node to create the data splits. The data set was split into 40% training, 30% validation and 30% test sample. As seen earlier, the percentage of the observations with high quality was very small; therefore stratified random sampling was performed to ensure that each spilt is an accurate representation of the actual population. The sample used in model building is an accurate representative sample of the actual population.

We used Logistics Regression, Neural Networks and Decision Trees to build the model. Since the target is binary, Logistic Regression is suitable. We used the stepwise method for variable selection in the Logistic Regression analysis. Stepwise Logistics Regression is recommended method of variable selection for exploratory research (Hosmer and Lemeshow 2000). We create some interaction terms for Logistic Regression analysis as well.

Preliminary analysis indicated that Decision Trees was not a suitable technique for the given data. Several combinations of the number of terms in each leaf and splitting algorithms were used, but none gave reliable results. Therefore, the use of Decision Trees for the analysis was abandoned. The further comparisons and model building was performed using the Logistics Regression and Neural Networks. The outputs of the initial runs are shown in Figures 4 and 5.

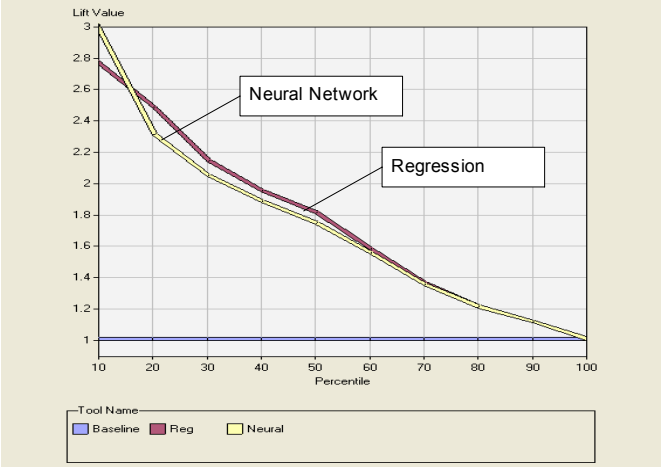

Figure 4: Lift Values for LR and NN nodes for MTTR

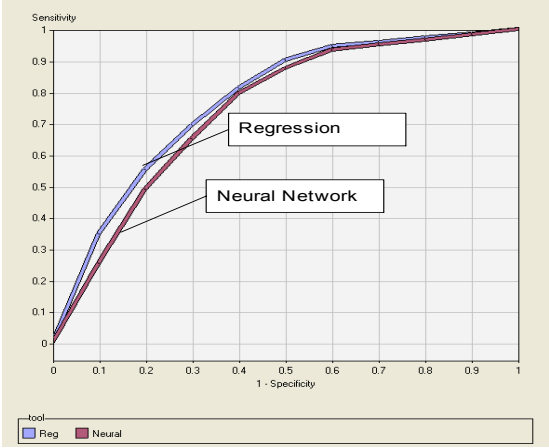

Figure 5: ROC Curves for LR and NN nodes for MTTR

We performed various comparative tests to select the best model. Based on the AIC, Lift and ROC Curve values, the Logistics Regression model was selected as the final model. The model was built using stepwise regression and possible interaction affects were examined. The model was evaluated performing diagnostic testing and Logistics Regression evaluation criteria.

To test the fit of the final model, the first step is to ensure that the model contains all required variables, entered in the correct functional form. The next step is to evaluate the affectivity of the model, i.e. the goodness-of-fit. This is to ensure that knowing the values of all the independent variables in the model allows an accurate prediction of resilience, better than the case of no information in the independent variables. The next step is to evaluate how well the group of independent variables explains the resilience. In Logistic Regression models, the Log Likelihood (LL) criteria are used to select model parameters. The values of -2LL of the model with and without the independent variables were used to check the model fit. The fit of the model is determined by the reduction in the value of -2LL with and without the covariates. The results of this test are shown in Table 2. The results showed that the model is significant at 5% significance level ($p < 0.0001$).

-2 Log Likelihood		Likelihood Ratio Chi-Square	DF	Pr> ChiSq
Intercept Only	Intercept & Covariates			
1168.711	972.835	195.8758	4	< 0.0001

Table 2: Likelihood Ratio Test for Global Null Hypothesis: BETA=0

The area under the ROC curve can test the accuracy of a Logistic Regression model. As a rule, area under the curve indicates how well the model provides discrimination between the high and low values of the target variable. For the final model the area under ROC curve was almost 0.8, which implies that the selected model for maintenance quality provides excellent discrimination between the projects of high and low MTTR. The ROC curve for the model is shown in Figure 6.

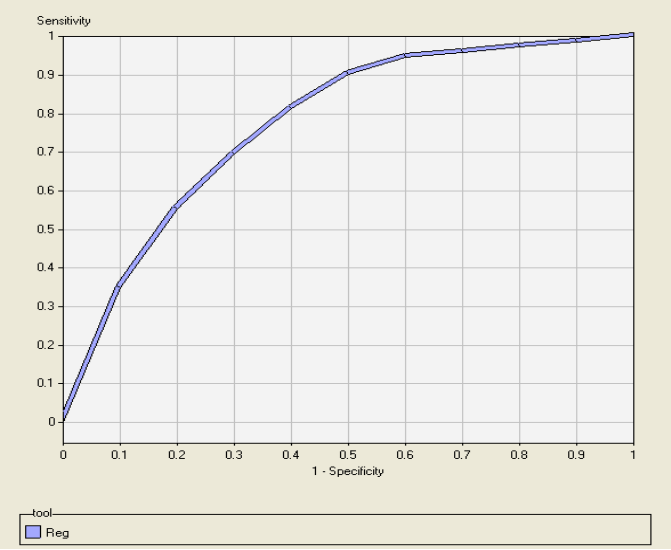

Figure 6: ROC Curve for the Final LR Model of Resilience

DISCUSSION

The final model that we selected was the Logistic Regression model. This model had the best-fit statistics and is the easiest to interpret. The misclassification rate for this model is at an acceptable 9%. The ROC indicates that using this model we can better predict the software maintenance quality of OSS projects. Therefore, before adoption decisions are made regarding use of an OSS project, use of this model can predict whether the project will have high or low maintenance quality. Based on this knowledge a better and informed decision can be made.

In the final model the factors that impact the OSS maintenance quality are also of interest to us. Knowing these factors we can further explore how and why certain project characteristics impact the project outcomes. The project development teams can use these factors to control the quality of their project. They can improve the quality by controlling the factors that affect it.

We find that the errors reported by users have a positive impact on maintenance quality. This means that projects with higher end user activity in reporting errors have a greater probability of having low MTTR and high quality. This result appears to be counter intuitive in beginning because it relates the high number of errors to high quality. But deeper analysis reveals that it is the percentage of user-identified errors that have a positive affect on quality. The number of user identified errors, indicates that the end users of the project are involved in the maintenance process. It also indicates that they are responsible to report the errors they detect.

We find that the number of downloads has a positive impact of maintenance quality. This means that projects that are downloaded more frequently have a higher quality. Number of downloads can be used as an indicator of the popularity of the project. The more known a project is the chances of downloads become higher. It also indicates that the project is useful to the users. For example, OSS users will download projects that offer them the functionality they need. Therefore number of downloads also indicate how relevant or useful the project is to the end user community.

We find that the use of mail has a negative affect on maintenance quality. This result appears counter intuitive because use of mail messages should not adversely affect the maintenance process. One of the possible explanations can be that if a project offers mail messaging, then some users and developers might end up using the mail messaging system to report errors instead of the error repository. This could cause potential delays in the correct reporting of errors and therefore delay the process of error removal. This factor alone can be studied in detail, to discover the cause of this negative affect.

We find that the age of the project has a negative impact on maintenance quality. This implies that the newer projects have a higher quality than the older projects. In OSS this could be indicative of the evolution of the better maintenance practices. OSS phenomenon is quite new compared to traditional software development. There are new tools, methods and communities being developed rapidly. Therefore it is possible that the newer projects are more aligned to the needs and structure of OSS community. However, an alternative explanation comes from prior research in software engineering. According to the laws of software evolution as a software system ages, it becomes more complex and harder to maintain. The increased complexity can increase the difficulty in removing errors because of complex interfaces and spaghetti code. The affect of age is being used as a control variable in this study. Later studies can be done with focus at the affect of age on maintenance quality over the entire lifecycle of single project.

CONCLUSION

In this paper, we have presented the preliminary results of the model for OSS maintenance quality. OSS projects offer access to all aspects of their development and maintenance data sets. Therefore these projects are well suited for Data Mining studies. Robust models can be developed using these rich datasets. Our study is a step towards better understanding of OSS project maintenance and toward building robust models for OSS project selection for corporate use. We are further exploring the results and building a more descriptive model for OSS maintenance outcomes.

SAS Enterprise Miner and SAS Text Miner provide us with ability to create effective predictive models. An extension of this study analyses each error and the text associated to it, to predict the time taken to remove the error. The large amount of textual information for OSS projects offers a well-suited application for the combined power of Text and Data Mining.

REFERENCES

Bennett, K. H. and V. T. Rajlich (2002). Software Maintenance and Evolution: a roadmap. 22nd ICSE, Limrick, Ireland.

Cearly, D. W., J. Fenn, et al. (2005). Gartner's Position on the Five Hottest IT Topics and Trends in 2005.

Hosmer, D. W. and S. Lemeshow (2000). Applied Logistic Regression. New York, John Wiley & Son Inc.

Madey, G. (2005). SourceForge.net Research Data Archive, http://www.nd.edu/ ~oss/Data/data.html.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Uzma Raja, Ph.D.
Department of Information Systems, Statistics and Management Science
University of Alabama
AL, 35487
Work Phone: 205.348.7688
Fax: 205.348.0560
E-mail: uraja@cba.ua.edu
Web: mis.cba.ua.edu