Mining Severe Priority Bugs in Software Maintenance

Satish C J, Thendral Puyalnithi

Abstract: ***Maintenance of open source software is a hectic task as the number of bugs reported is huge. The number of projects, components and versions in an open source project also contribute to the number of bugs that are being reported. Classification of bugs based on priority and identification of the suitable engineers for assignment of bugs for such huge systems still remains a major challenge. Bugs that are misclassified or assigned to engineers who don't have the component expertise, drastically affect the time taken towards bug resolution. In this paper we have explored the usage of data mining techniques on the classification of bugs and assignment of bugs to engineers.Our focus was on classifying bugs as either severe or non-severe and identification of engineers who have the right expertise to fix the bugs. The prediction of bug severity and identification of engineers were done by mining bug reports from JIRA, an open source software bug tracking tool. The mining process yielded positive results and will be a decision enhancer for severe bugs in the maintenance phase.***

Keywords: Mining Software Repository, Data Mining, Software Maintenance.

I. INTRODUCTION

A software system undergoes lot of changes during the software maintenance phase. The changes can be either change requests or bugs. Bugs are usually classified as severe, high, medium or low priority bugs. The priority denotes the severity of the bug and its impact on the business. Severe bugs are those which act as a blocker to the system execution and they need an emergency fix. All bugs that are reported should be fixed by the maintenance engineers within a given time period as mentioned in a Service Level Agreement.

Bug tracking systems are used for managing bugs. Whenever a system user identifies a bug, the user reports the bug using the bug tracking system.Based on the severity of the bug the user assigns a priority to the bug. Bugs that are reported are later assigned to maintenance engineers by project managers who manage the project. Most often the users who report the bug report a bug as a severe bug when it's actually a medium or low priority bug. If a reported severe priority bug is found to be of a lower priority upon analysis, the maintenance engineers can downgrade the priority of the bug after informing the user who reported the bug. The problem with this approach is that engineers spend time analyzing a low priority bug as a severe bug and this hampers the fixing of actual severe bugs.Another important problem with respect to bug fixing is identifying the right person who can fix a bug. Most often the assignment of bugs to engineers is done based on the workload of each engineer. As severe bugs need an immediate fix, only engineers who have a good experience in handling such bugs can provide a quick resolution. Assignment of severe bugs to engineers who don't have the right experience levels can lead to poor fixes or reassignment of bugs[1]. A delay in fixing a severe priority bug has a very high level of impact on the business and can decrease customer satisfaction[2][3].

Bug classification and assignment has been addressed by researchers in the past[3][4][5].Such studies considered all bugs to be of equal importance and were not focused on mining severe bugs. Our approach deals with addressing these issues on severe priority bugs by mining bug reports. We have provided mining as an approach for classification of bugs as either severe or non-severe. Such a classification of reported bugs can prevent assignment of low priority bugs as severe priority. Identification of the best person to assign a severe bug is also achieved by mining the past history on bug fixes.

1. Mining Process

We have used bug reports on QT which is open source software. Bug management for QT is done using JIRA, a bug tracking tool developed by Australian Company Atlassian [6].The important steps in our mining process are given below

- Extraction of Bug Reports from JIRA
- Import of Bug Reports in SQL Server Database
- Preprocessing Bug Reports
- Creation of Mining Models using SQL Server Analysis Services.

The bugs were either classified as severe or non-severe. We have also identified the assignees that are best suitable for both severe and non-severe bugs. Bugs that are classified as severe will be assigned to assignees that have been identified for fixing severe bugs using our mining method. This approach enables the handling of severe bugs effectively without any delay by utilizing the best person to fix the bug. This will reduce the downtime of the system and improve customer satisfaction on the supported business.

2.1 Extraction of Bug Reports from JIRA

The bug reports were extracted from bug tracking tool JIRA as Excel Files.As Jira was configured to allow only downloads of thousand bugs per report, there was a need to download many reports. Our approach was to download bug reports for every month, as the number of bugs reported for every month was lesser than thousand. Reports for the last five years were extracted as excel files from JIRAtool. Using an excel macro to merge files; the individual reports were merged as one single excel file. The merged report had 43840 rows and 71 columns.

2.2 Import of Bug Reports in SQL Server Database

The import of the merged bug report in to SQL Server database was achieved using the SQL Server 2016 import/export wizard. The contents of the merged excel file was loaded in to the relational table QT on SQL Server.

Revised Manuscript Received on December 05, 2019

Satish C J, School of Computer Science and Engineering, Vellore Institute of Technology, Vellore

Thendral Puyalnithi, Kalasalingam Academy of Research and Education, Krishnankoil

2.3 Preprocessing Bug Reports

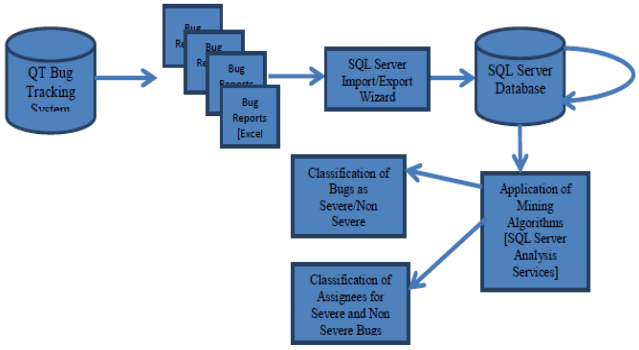
The report had 71 columns and many columns were filled with only null values.35 columns did not have any values and hence all the columns were dropped from our QT table.
The remaining columns were thoroughly scrutinized and only the columns mentioned in Table 1 were deemed relevant to the mining process

Table 1: Columns Identified for Mining

Column Name	Definition
ID	Primary Key
Project	Contains Project names of projects handled for QT framework like Qt Creator,Qt Installer Framework,Qt Mobility etc.
Reporter	Name of the person who reported the bug
Issue Type	Classifies the reported issue as bug, suggestion, task etc
Status	Holds statuses for the reported issue like open ,closed, in progress etc.
Priority	Holds the information on the severity of the issue.
Component	Holds the component names specific to the issue
Affects Version	Holds the version names for which the issue is reported
Assignee	Holds the name of the assignee that is assigned the bug

For the mining process we had to focus only on bugs that had the status has closed. All bugs that were not having the closed status were removed from the table. A total of 15474 bugs were removed from the table as they were not having the closed status.
Priority was maintained using the following priority levels in JIRA.

- P0: Blocker
- P1: Critical
- P2: Important
- P3: Somewhat important
- P4: Low



- P5: Not important

Figure 1: Mining Process

All bugs with priority P0 and P1 were updated with priority as Severe and all other bugs were updated with priority as Non-Severe in the QT
table on SQL Server Database. On further analysis for the null values on the shortlisted columns, component column was found to have 305 rows as null values and hence all the 305 rows were deleted from the table.
Thus as part of data preprocessing all the irrelevant columns and rows containing null values were completely removed. Only bugs with the closed status were retained.

2.4 Creation of Mining Models for classifying bugs as severe or non-severe.

SQL Server Analysis Services were used for generation of mining models using the QT table. The first mining process was focused on predicting priority of a reported bug. The input and output parameters selected for the mining process is given in table 2.

Table 2: Input and Output Parameters

Parameters	Input/output
ID	Key Column
Project	Input
Reporter	Input
Issue Type	Input
Priority	Predict [Output]
Component	Input
Affects Version	Input

Mining models were created using the following algorithms

- Decision Trees
- Naïve Bayes
- Clustering
- Association
- Neural Network

A generated decision tree for three levels is shown in figure 2

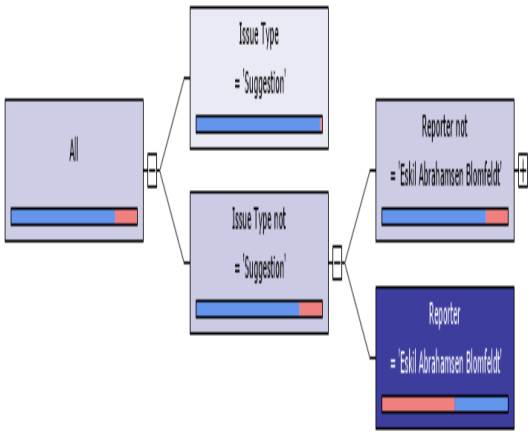


Figure 2: Decision Tree [3 Levels]

Decision Tree is also shown using Microsoft Generic content tree viewer in figure 3.

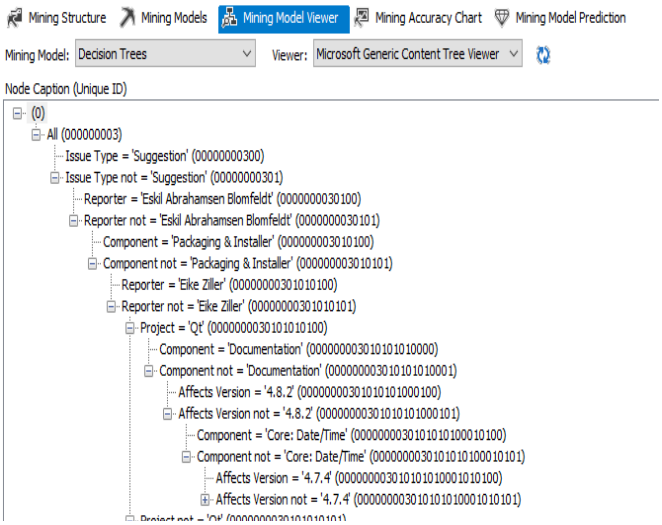


Figure 3: Decision Tree –Generic Content Tree Viewer

The comparison of all algorithms with respect to predicting severe bugsis shown in figure 4 as a lift chart. Figure 5 contains the mining legend of the comparison.

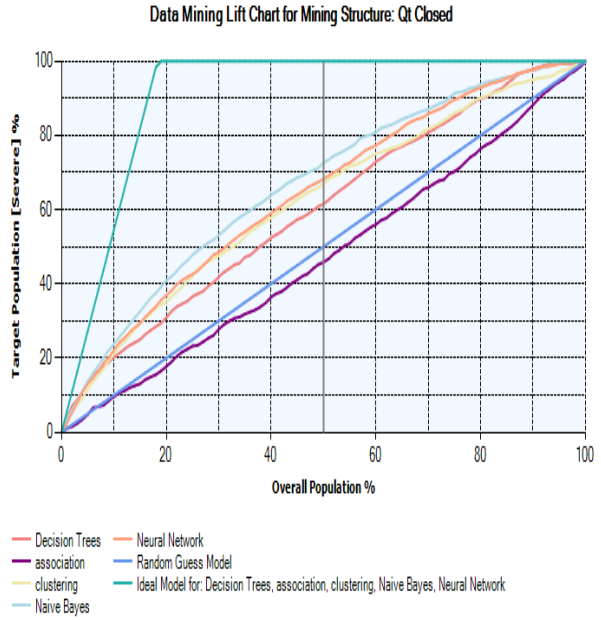
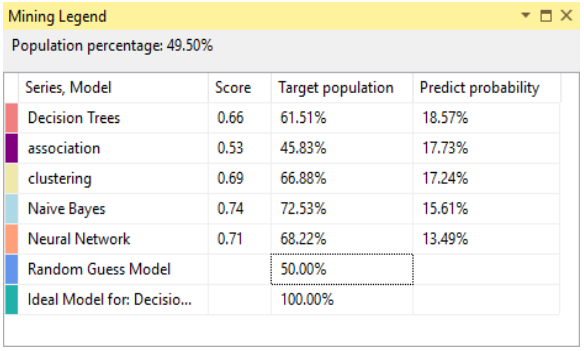


Figure 4: Lift Chart [Severe Bugs]



Series, Model	Score	Target population	Predict probability
Decision Trees	0.66	61.51%	18.57%
association	0.53	45.83%	17.73%
clustering	0.69	66.88%	17.24%
Naive Bayes	0.74	72.53%	15.61%
Neural Network	0.71	68.22%	13.49%
Random Guess Model		50.00%	
Ideal Model for: Decisio...		100.00%	

Figure 5: Mining Legend [Severe Bugs]

The comparison of all algorithms with respect to predicting non-severe bugs is shown in figure 6 as a lift chart and figure 7 depicts the mining legend of the comparison

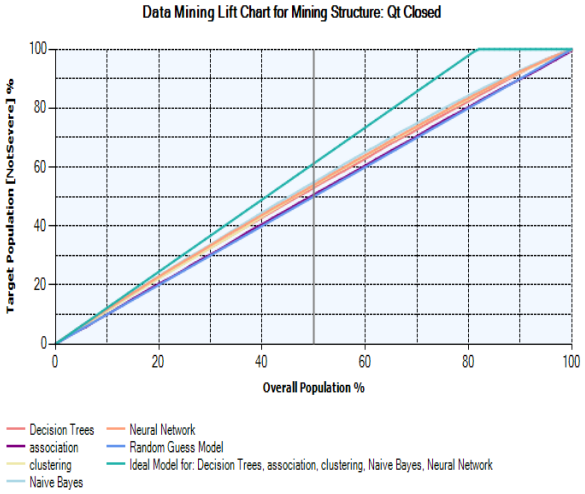
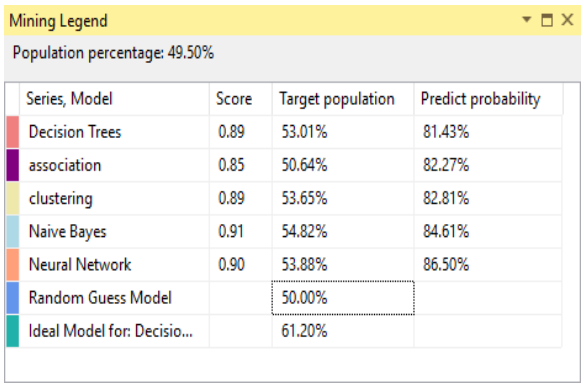


Figure 6: Lift Chart [Non Severe Bugs]



Series, Model	Score	Target population	Predict probability
Decision Trees	0.89	53.01%	81.43%
association	0.85	50.64%	82.27%
clustering	0.89	53.65%	82.81%
Naive Bayes	0.91	54.82%	84.61%
Neural Network	0.90	53.88%	86.50%
Random Guess Model		50.00%	
Ideal Model for: Decisio...		61.20%	

Figure 7: Mining Legend [Non Severe Bugs]

2.5 Creation of Mining Models for classifying assignees for Severe and Non-Severe Bugs

The main objective behind this mining is the identification of Assignees that can work on a severe bug or a non-severe bug We have applied Association Rule mining to generate all the associations between Assignees, component, project and priority. For severe bugs the support count is maintained at 40 and minimum probability is 0.42. The list of rules that satisfy the minimum support provide us with details of experienced assignee for each component in every project. Whenever a bug is classified as severe based on our previous analysis then that bug can be assigned to an assignee using the association rules generated in this step. A part of the association rules generated for severe bugs is given in figure 8.

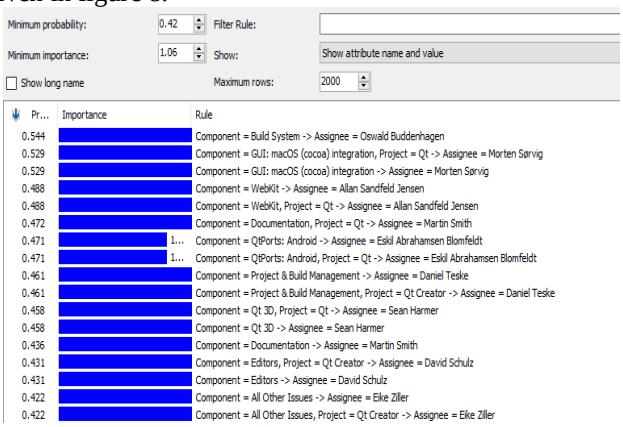


Figure 8: Association Rule Viewer

Association rules between component, project assignee and non-severe bugs can be identified and used for assigning classified non-severe bugs.

II. DISCUSSION

The decision making process on severe bugs in the software maintenance phase will be enhanced by the mining of bug reports. Our objective was to improve the handling of severe bugs in software maintenance so that system downtime is reduced by classifying and assigning bugs to the best suitable engineers.

The mining models used for such classification provided us with a greater insight on classification of bugs. Severity of a bug depends on the project, component, version, issue type andreporter. It is observed that certain components always receive non-severe bugs and certain reporters only report non-severe bugs. When bugs are reported for such components or raised by such reporters it can be directed to engineers who are identified to work on non-severe bugs. Whereas the bugs arising out of components and reporters who report a majority of the severe bugs can be assigned to engineers who are identified to work on severe bugs.

The classification models have given us an opportunity to explore the hidden knowledge on the factors that affect the severity of a bug. The comparison of the various algorithms reveals that Naïve Bayes is more effective in classification of severe bugs in the test data and the mining legend shows that the classification was good for almost 74% of the targeted population.

III. CONCLUSION

The use of datamining technique for classification and assignment of severe bugs not only aids the decision making process but also uncovers a lot of factors that contribute to the severity levels of a bug. Such a mining model not only helps in classification of bugs but also gives us information on the projects, components, versions that contribute to the maximum number of severe bugs. Corrective action could be taken to reduce the number of severe bugs on such projects. The mining models also enable the identification of the pool of resources that are skilled in fixing severe bugs for each version, each component in every project. Thus application of mining models in software maintenance for mining severe priority bugs brings us the promise of system down time reduction through effective resource utilization.

REFERENCES

1. Shao, Qihong, et al. "Efficient ticket routing by resolution sequence mining." Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2008.
2. Ohira, Masao, et al. "A dataset of high impact bugs: manually-classified issue reports." 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories. IEEE, 2015.
3. Lamkanfi, Ahmed, and Serge Demeyer. "Predicting reassignments of bug reports-an exploratory investigation." Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on. IEEE, 2013.
4. Zhou, Yu, et al. "Combining text mining and data mining for bug report classification." Journal of Software: Evolution and Process (2016).
5. Tian, Yuan, David Lo, and Chengnian Sun. "Drone: Predicting priority of reported bugs by multi-factor analysis." (2013): 200.
6. QT issues download page
7. https://bugreports.qt.io/browse/QTWEBSITE-745?jql=.