Assessment Accuracy of Random Forest Classifier on Imbalanced Data to Predict Change Request Severity Level

Luiz Alberto Ferreira Gomes
Software Engineering Departament
Pontifical Catholic University of Minas Gerais - PUC MG
Poços de Caldas, Brazil
luizgomes@pucpcaldas.br

Mario Lúcio Côrtes
Institute of Computing
State University of Campinas - UNICAMP
Campinas, Brasil
cortes@ic.unicamp.br

***Abstract*—REMOVED: In the context de Change Request (CR) systems, the severity level of a change request is considered a critical variable in the planning of the software maintenance activities, indicating as soon a CR needs to be fixed. However, the severity level assignment remains a process primarily manual depending on the experience and expertise who has reported the CR. In this paper, we present the preliminary findings of research aim to predict the severity level of a CR by analyzing its long description using text mining and a classifier based on Random Forest algorithm. The results have evidenced that this classifier can predict the severity level will change and whether severity will increase or decrease with reasonable accuracy, around 94,12% and 93,86% respectively. However, it has provided poor accuracy (around 67,29%) to predict the final last severity level in imbalanced data scenario. NEW: In the context de Change Request (CR) systems, the severity level of a change request is considered a critical variable in the planning of the software maintenance activities, indicating how soon a CR needs to be fixed. However, the severity level assignment remains primarily a manual process, mostly depending on the experience and expertise of the person who has reported the CR. In this paper, we present preliminary findings of research aimed to predict the severity level of a CR by analyzing its long description, using text mining and a classifier based on Random Forest algorithm. The results have evidenced that this classifier can predict the severity level will change and whether severity will increase or decrease with good accuracy, around 94,12% and 93,86% respectively. However, it has provided poor accuracy (around 67,29%) to predict the final last severity level in imbalanced data scenario. MARIO: ultima frase: pensar outra abordagem. Está muito negativo. Dizer a mesma coisa de forma positiva. Comparar com a literatura? e sobre a nova medida? e comparativo com a literatura (no abstract)?**

***Keywords*-software maintenance; change request systems MARIO: Ver nome padrao na literatura; nao seria BTS bug tracking system?; machine learning; random forest.**

I. Introduction

Change Request (CR) systems REMOVED: has been performed a critical NEW: have played a major role in maintenance process in many software development settings, both in Close Source Software (CSS) and in Open Source Software (OSS) scenarios. Mainly in the latter, which is characterized by the existence of many of users and developers with REMOVED: distinct expertise levels NEW: different levels of expertise spread out around the world, who might REMOVED: register or deal with any amount of NEW: create or be responsible for dealing with several change requests [1].

A user interacts with a CR system often through a simple mechanism called CR form. This form enables him to request changes, to report bugs or to ask for support in a software product [2]. Initially, he or she should inform a short description, a long description, a type (e.g. bug, new feature, improvement, and task) and a NEW: associated severity level (e.g. blocker, critical, major, minor and trivial)REMOVED: for his or her solicitation MARIO: evite a palavra solicitation. Subsequently, a development team member will review this request and, case it is not refused for some reason (e.g. request duplication), he or she will complete the information in CR form, indicating, for example, its priority and the person responsible for accomplishing it. MARIO: importante usar terminologia e jargao do ramo: acho que aqui nao eh accomplish e sim assigned for the CR; verificar em outros artigos

The severity level information is recognized as a critical variable in the equation to estimate a prioritization of change request prioritization [3] MARIO: repetido. Consequently, it can be a decisive factor how soon it needs to be fixed [4]. However, the severity level assignment remains a mostly manual process which relies on experience and expertise of the person who has opened the CR [1], [3], [4]. So, it may allow a high degree of subjectivity and, consequently, it may be quite error-prone.

The number of CR made is frequently very high in large and medium software projects [5]. Severity level shifts throughout CR lifecycle (Figure 1) could cause substantial impacts on the maintenance activities planning and could drive the development team to solve the least significant change requests before the most important ones. In this scenario, the support of a computational tool to help the user to assign and the development team to verify the CR severity level well desirable.

RQ 1: *Is it possible to predict whether the change request severity degree will shift during its lifecycle?* Based on

long description filled by users in CR form, we have investigated if it is possible to predict using a Random Forest classifier, with a high degree of accuracy in an imbalanced data scenario, whether the severity level of the request will be changed.

RQ 2: *Is it possible to predict whether the change request severity level will increase, decrease or remain the same during its lifecycle*? Based on long description filled by users in CR form, this question has enabled us to investigate if it is possible to predict using Random Forest classifier, with a high degree of accuracy in an imbalanced data scenario, whether the severity of the request will be increased, decreased or remained the same.

RQ 3: *Is it possible to predict the change request severity degree at the end of its lifecycle?* Finally, we have investigated if it is possible to predict using a Random Forest classifier, with a high degree of accuracy in an imbalanced data scenario, the severity level of a request based on the same information used in the two previous questions.

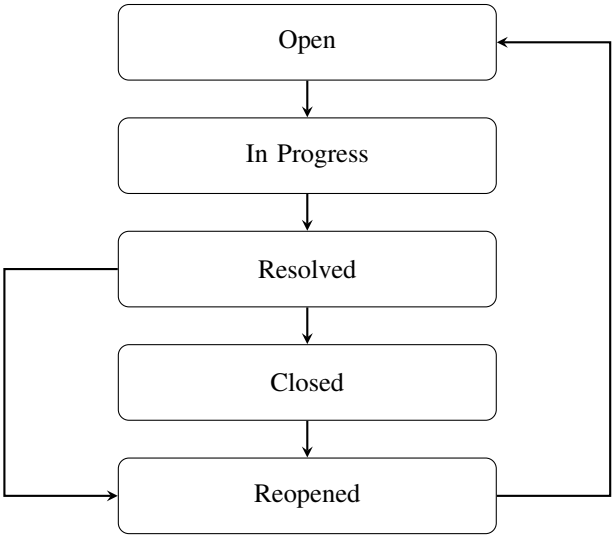

Fig. 1. CR life cycle [2].

This paper presents the preliminary results of research whose the objective is to implement an automatic mechanism, based on random forest machine algorithm, aim to aid users, developers and managers to assign the CR severity level more accurately. The contributions of this work are as follows

1) we...
2) we...
3) we...

The structure of this paper is as follows. In Section 2, we describe some related works. In Section 3, we provide the information background about CR systems, text mining and machine learning techniques necessary to understand our approach. Section 4 describe our work. Findings and discussions are presented in Section 5. Finally, we conclude and discuss future work in Section 6.

II. Related Works

Menzies [6] have developed a method, named SEVERIS (SEVERity ISsue assessment), for evaluating the severity of changes requests. SEVERIS is based on common text mining techniques (e.g. tokenization, stop word removal, stemming, Tf*Idf and InfoGain) and on the data mining techniques (e.g. RIPPER). The method was applied to five projects managed by the Project and Issue Tracking System (PITS). - an issue tracker system used by NASA.

Lamkanfi et al. [4] have developed an approach to predict severity level of a CR based on text mining algorithms (tokenization, stop word removal, stemming) and on the Naïve Bayes machine learning algorithm. They have been validated their approach over from three open source project Mozilla, Eclipse, and GNOME and they accomplished that a training set with approximately 500 change requests per severity degree are enough to predict it with a reasonable accuracy. In another following work, Lamkanfi et al. [5], these authors compared the accrual of four machine-learning algorithms (Naïve Bayes Multinomial, K-Nearest Neighbor, and Support Vector Machine). They have been concluded that Naïve Bayes Multinomial gave superior performance compared to the others proposed algorithms.

Valdivia et al. [7] have characterized blocking bugs in six open source projects and proposed a model to predict them. Their model was composed of 14 distinct factors or features (e.g. the textual description, location the bug is found in and the people involved with the bug). Based on these factors they build decision trees for each project to predict whether a bug will be a blocking bug or not. Then, they analyze these decision trees to determine which factors best indicate these blocking bugs.

III. BACKGROUND

In this section, we describe de change request process. Next we explain the common approach to pre-processing textual documents, and lastly, we highlight the Random Forest algorithm to classify and predict the CR severity level.

A. Change Request Systems

Change Request systems [8] are software employed to keep the recording and tracking information of requests for modifications, bug fixes, and support that could occur during the software life cycle.

Although there is no a common sense regarding the terminology or the amount of information that users must fill in to complete his requisition between popular CR systems (e.g. Bugzilla, Jira, and Redmine) [3], typically, they shall fill in a form containing at least the following attributes shown in Table I.

Once the request has been registered by the user, the development team will assess it and, if it not canceled for some reason (e.g. duplication), they will complement the information with, for example, the person responsible for

TABLE I
COMMOM ATTRIBUTES IN THE CR FORMS.

Type	Type of request (e.g. bug, new feature, improvement, and new feature)
Title	Short description of request in one line.
Description	Long and detailed description of request in many lines. It could include source code snippets and stack tracing reports.
Severity	Level of severity of request (e.g. blocker, critical, major, minor and trivial).

handling this request. All these data are stored in a repository, keeping relevant historical data about a particular software.

B. Text Mining

Text mining is the process to convert unstructured text into a structure suited to analysis [9]. It is composed of three basic activities [10]: tokenization, stop word removal and stemming.

Tokenization is the action to parsing a character stream into a sequence of tokens by splitting the stream at delimiters. In this context, a token is defined as a block of text or a string of characters (without delimiters such as spaces and punctuation) that is recognized as a useful portion of the unstructured data.

Stop words eliminates commonly used words that do not provide relevant information to a particular context, including prepositions, conjunctions, articles, common verbs, nouns, pronouns, adverbs, and adjectives.

Stemming is the process stemming is the process of reducing or normalizing inflected (or sometimes derived) words to their word stem, base form—generally a written word form (e.g. "working" and worked into work).

C. Machine Learning

The machine learning is considered a part of artificial intelligence area whose the primary purpose is to resolve a given problem using experience or example data [11]. It can be seen as an improvement over a set of techniques or methodologies which can make a computer to learn by the study of data sets.

1) Random Forest: The Random Forest algorithm [12] relies on two core principles: (i) in the creation of hundreds of decision trees and the joining them into a single model; and (ii) in the closing decision based on the ruling of the majority of the forming trees which are treated as equals.

A random forest model is considered a suited alternative for model construction for a many of reasons [10]

- Requires little or no data preprocessing, no data normalization and it is resilient to outliers.
- Requires no variable selections because the algorithm does its own.
- Models resultants from each tree in the forest tend not to overfit to the training dataset because they are built using two levels of randomness (observations and variables).

D. Evaluation Metrics

From Information Retrieve(IR) discipline, the three most common performance measures for evaluating the accuracy of classification algorithms are precision, recall, and F-measure [9].

Recall. The recall for a class can be defined as the percentage of correctly classified observations among all observations belonging to that class. It can be thought of as a measure of a classifiers completeness. More formally [13]: the recall is the number of True Positives(TP) divided by the number of True Positives(TP) and the number of False Negatives(FN), where the TP and FN values are derived from the confusion matrix. A low recall indicates many False Negatives [14].

Precision. The precision is the percentage of correctly classified observations among all observations that were assigned to the class by the classifier. It can be thought of as a measure of classifier exactness. More formally [13]: the precision is the number of True Positives(TP) divided by the number of True Positives and False Positives(FP), as well as TP and FN, FP also comes from the confusion matrix. A low precision can also indicate a large number of False Positives [14].

F-measure. F-measure conveys the balance between the precision and the recall and combines the two measures in an ad hoc way [9], [14].F-measure can be calculated using the formula $2 * ((precision * recall))/(precision + recall)$.

IV. EXPERIMENT

In this section, we describe the datasets that we use in this study, followed by our methodology. We then present the measures used to evaluate the approaches.

A. Methodology

Our experiment was conducted following the steps have shown in Figure 1.

Data Extraction. The HADOOP CR are registered and stored via Jira software. The data extraction has occurred in two steps: (i) the change requests basic data were extracted and stored in XML format; and (ii) the modification histories in the change requests were extracted and stored in HTML file.The extraction in two-steps was needed because the XML file with the change requests basic data did not include the data about request modifications history.

PreprocessingThe pre-processing have applied the text mining on long description field to extract the most frequent terms in all CR and to count each term by CR and have merged the basic data of the documents with the history of change Priority.

Training and testing. Data were randomly partitioned at 80% for training and 20% for testing. The hyperparameter of the random forest algorithm was chosen through cross validation with three folds on the training data. The tests were then made with the most suitable parameters with the remaining 20%.

Analysis of results. The analysis of results based on precision, recall and F-measurement metrics to measure the accuracy.

B. Dataset

To evaluate and validate the proposed approach, we have utilized de change requests from Apache HADOOP project. According to [hadoop.apache.org], HADOOP is "framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models." The reason we choose this project were as follows: (i) It is complexity OSS with many modules; (ii) the change requests data are available and it can be easily extracted, both in the XML and JSON formats; and (iii) It has many users with varying experience level.

We extracted change requests published from February 01, 2006 to January 18, 2017. We only retrieved requests from the common module which identifier started with HADOOP. The total number of records retrieved after preprocessing was 8858 change requests. Figure IV-B shows three chart with requests distribution by severity level or severity level changes.

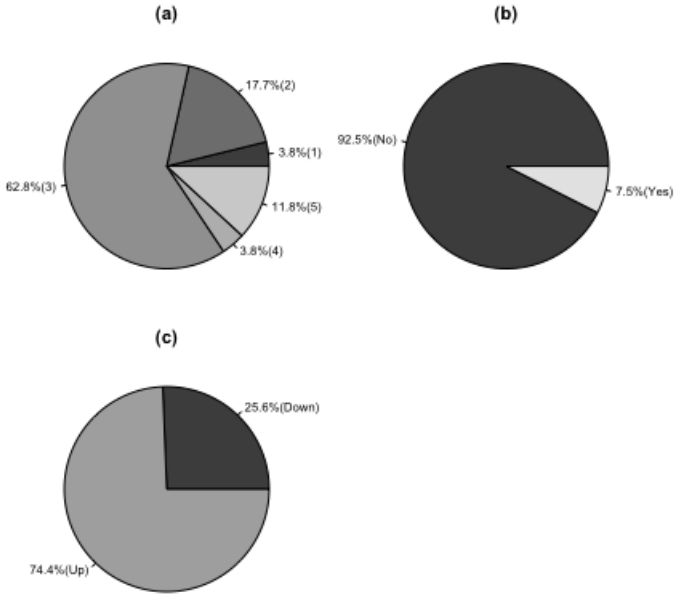

Fig. 2. Dataset distribution by severity level.

The graph (a) shows that of 8858 change requests, 3.8% had severity 1(trivial); 17.7% had severity 2(minor); 62.8% had severity 3(major), 3.8% had severity 4(critical), And 11.8% had severity 5(blocker). The graph (b) shows that 92,5% of them haven't changed their severities levels and 7,5% have changed. And the last chart reveals that of the 7.5% which changed their severity, 25,6% reduced it, and 74,4% increased it. We can note that dataset is clearly imbalanced.

V. Findings and Discussions

A. RQ1: Is it possible to predict whether the change request severity level will shift during its lifecycle?(RQ1)

The RQ1 is a simple binary problem or question whose answer is yes (positive class) or no (negative class). The Table II shows the performance of random forest to predict the answer to this question.

TABLE II
Random Forest Performance for RQ1.

Accuracy	Precision	Recall	F-Measurement
94,128%	95,032%	1,000%	96,924%

We tested the model with 3423(40% of 8858) change requests: 3167 have changed their severity level, and 256 haven't changed their severity level. The Figure V-A in terms of the amount of change requests whose predictions of severity level changes was correct or incorrect. We can note that the random forest performance was also 94,13%.

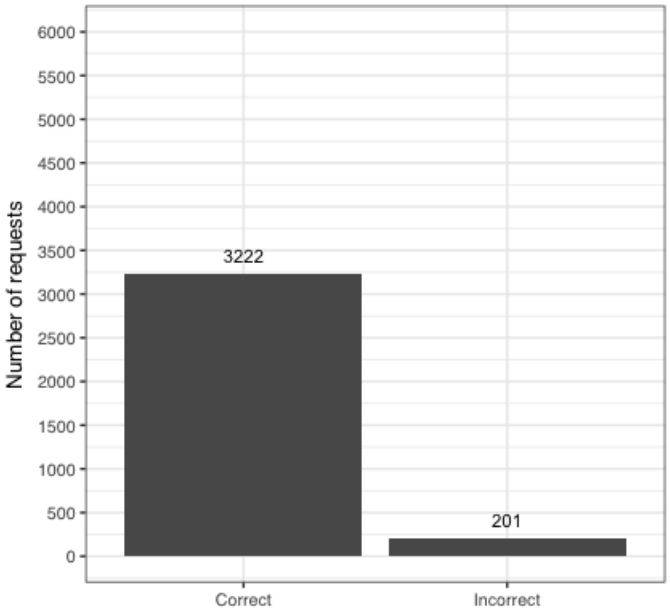

Fig. 3. Performance of Random Forest for RQ1.

B. RQ2: Is it possible to predict whether the change request severity level will increase, decrease or remain the same during its lifecycle

The RQ2 is a problem little harder than RQ1. It is a question with three responses classes related to severity: (-1) it was changed to down; (0) it was maintained; and (1) it was changed to up.The Table III shows the performance of random forest to predict the answer to this question.

TABLE III
Evaluation metrics for RQ2.

Class	Precision	Recall	F-Measurement
-1	6,15%	100,00%	11,59%
0	100,00%	93,92%	96,86%
1	22,11%	91,30%	35,59%
Average	42,75%	95,07%	48,02%

We tested the model with 3423(40% of 8858) change requests: 3167 have changed their severity level, and 256 haven't changed their severity level. The Figure V-B in terms of the amount of change requests whose predictions of severity level changes was correct or incorrect. We can note that the random forest performance was also 93,86%.

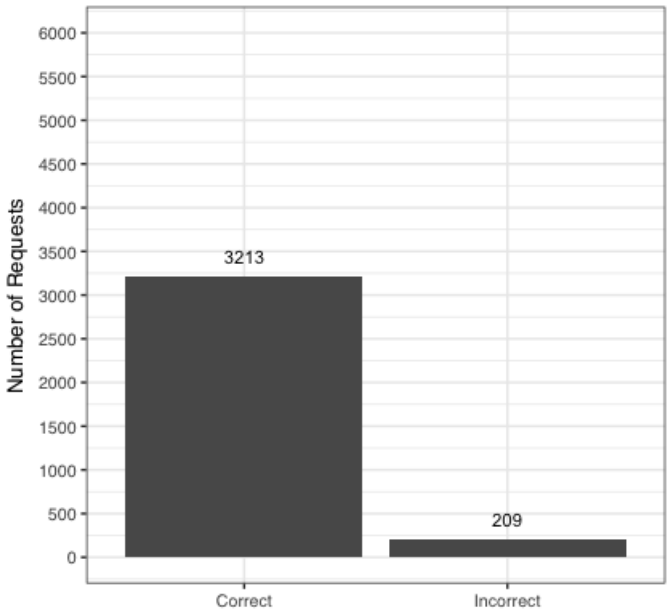

Fig. 4. Performance of Random Forest for RQ2.

C. RQ3: Is it possible to predict the change request severity level at the end of its lifecycle?

The RQ3 is a problem much harder than other two. It is a question with five responses classes related to severity: (1) trivial; (2) minor; (3) major; (4) critical; and (5) blocker. The Table III shows the performance of random forest to predict the answer to this question.

TABLE IV
EVALUATION METRICS FOR RQ3.

Class	Precision	Recall	F-Measurement
1	9,23%	100,00%	16,90%
2	13,86%	77,77%	23,53%
3	98,23%	67,34%	79,90%
4	28,12%	100,00%	43,90%
5	25,24%	80,95%	38,48%
Average	34,94%	85,21%	40,54%

We tested the model with 3423(40% of 8858) change requests: 130 are trivial; 606 are minor; 2151 are major; 129 are critical; 405 are a blocker. The Figure V-C shows three graphs. The graph (a) shows user range error in the assignment of severity level. The graph (b) shows the classifier error in the assignment of severity level. And the graph (c) compares de Predictor Error (PE) with User Error(UE) in terms of the amount of change requests whose predictions. We can note that the random forest performance was also 34,33%.

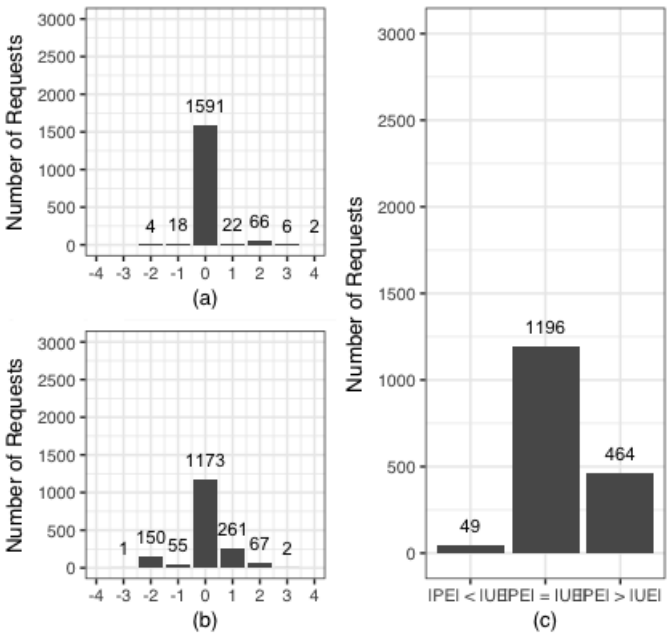

Fig. 5. Performance of Random Forest for RQ3.

VI. THREATS TO VALIDITY

Runeson [15] recommends that threats to validity should be considered under four aspects: construct validity; internal validity; external validity; and reliability.

Construct validity. Despite existing others metrics to evaluate classifiers [13], which could be more suitable than precision, recall and F1-measure, we prefer to use them because they have been used satisfactorily in related works [4]–[7].

Internal validity. We assume that level of severity assignment by the user is correct and that there is an intimate relationship between it and the long description of the change request. This assumption finds echo or support in [3], [4]

External validity. We have considered one single repository and we have extracted 8858 change requests from it. Although we can't generalize the results to others, the characteristics presented by HADOOP repository, particularly regarding the balance of the data, are similars to those shown in the repositories studied [3]–[5], [7].

Reliability. The code developed in the Java language and the R language for preprocessing, training, testing and analysis of results have been carefully checked may contain bugs. To minimize this problem, we rely heavily on libraries offered by them such as XStream (the XML parser) and randomForest (a random forest implementation).

VII. CONCLUSION AND FUTURE WORK

In this paper, we assess the Random Forest, a popular machine learning algorithm to CR severity level in imbalanced data scenario. We have considered the CR long description

as the main factor to this prediction. The features of machine learning were derived from words(token) from this description. The results on a dataset consisting more than 8,000 CR from Hadoop have shown that random forest performed well to predict the severity level will change and whether severity will increase or decrease with reasonable accuracy, around 94,12% and 93,86% respectively. However, it has provided poor accuracy (around 67,29%) to predict the final last severity level in imbalanced data scenario.

ACKNOWLEDGMENT

This work has been carried out in the context of Ph.D program of Computing Institute at State University of Campinas (UNICAMP), Brazil. Additional sponsoring by Permanent Professor Preparation Program(PPPP) of Pontifical Catholic University of Minas Gerais (PUC MG).

REFERENCES

[1] Y. C. Cavalcanti, P. A. da Mota Silveira Neto, I. d. C. Machado, T. F. Vale, E. S. de Almeida, and S. R. d. L. Meira, "Challenges and opportunities for software change request repositories: a systematic mapping study," *Journal of Software: Evolution and Process*, vol. 26, no. 7, pp. 620–653, jul 2014.

[2] I. Sommerville, *Software Engineering*, 2010.

[3] Y. Tian, D. Lo, and C. Sun, "Information Retrieval Based Nearest Neighbor Classification for Fine-Grained Bug Severity Prediction," in *2012 19th Working Conference on Reverse Engineering*, oct 2012, pp. 215–224.

[4] A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals, "Predicting the severity of a reported bug," *Proceedings - International Conference on Software Engineering*, pp. 1–10, 2010.

[5] A. Lamkanfi, S. Demeyer, Q. D. Soetens, and T. Verdonckz, "Comparing mining algorithms for predicting the severity of a reported bug," *Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR*, pp. 249–258, 2011.

[6] T. Menzies and A. Marcus, "Automated severity assessment of software defect reports," *IEEE International Conference on Software Maintenance, 2008. ICSM 2008*, pp. 346–355, 2008.

[7] H. Valdivia Garcia, E. Shihab, and H. V. Garcia, "Characterizing and predicting blocking bugs in open source projects," *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, pp. 72–81, 2014.

[8] R. S. Pressman, *Software Engineering A Practitioner's Approach 7th Ed - Roger S. Pressman*, 2009.

[9] R. Feldman and J. Sanger, *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, 2007.

[10] G. Williams, *Data Mining with Rattle and R: The Art of Excavating Data for Knowledge Discovery*, 2011. [Online]. Available: http://books.google.com/books?id=mDs7OXj03V0C

[11] K. Surya, R. Nithin, and R. Venkatesan, "A Comprehensive Study on Machine Learning Concepts for Text Mining," *International Conference on Circuit, Power and Computing Technologies [ICCPCT] A*, vol. 3, no. 1, pp. 1–5, 2016.

[12] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[13] K. Facelli, A. C. Lorena, J. Gama, and A. Carvallho, *Inteligência Artifical: uma abordagem de aprendizado de máquina*. Rio de Janeiro: LTC, 2015.

[14] Y. Zhao and Y. Cen, *Data Mining Applications with R*, 1st ed. Academic Press, 2013.

[15] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, 2009.