Machine Learning Based Prediction of Change Request Severity Level: Experimental Results

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*— In the context of Change Request (CR) systems, the severity level of a change request is considered a critical variable when planning software maintenance activities, indicating how soon a CR needs to be addressed. 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 ongoing research aimed to predict the severity level of a CR by analyzing its long description, using text mining techniques and Machine Learning (ML) algorithms. Best results were obtained with a classifier based on the Random Forest ML algorithm. This classifier can predict whether the severity level will change (accuracy of 93.681%) and if it will increase or decrease (accuracy of 93.440%). However, preliminary results were not as good when predicting the final severity level in an imbalanced data scenario.**

***Keywords*-software maintenance; change request systems; machine learning; random forest.**

I. Introduction

Change Request (CR) systems 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 different levels of expertise spread out around the world, who might 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 an associated severity level (e.g. blocker, critical, major, minor and trivial) . 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.

The severity level information is recognized as a critical variable in the equation to estimate a prioritization of change request [3] . 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 OSS projects [5]. Severity level shifts throughout CR lifecycle (Figure 1) could cause relevants impacts on the maintenance activities planning and could drive the development team to solve the least significant change requests before the most important ones.

Machine Learning (ML) techniques have been applied successfully in solving real problems in many areas of knowledge, including those related to CR, such as duplication and assignment of CR [1]. However, the accuracy of ML algorithms may be affected by imbalance datasets [6] —a recurring critical problem in CR repositories [7]. For example, more than 60% of CRs may have a ”major” severity level.

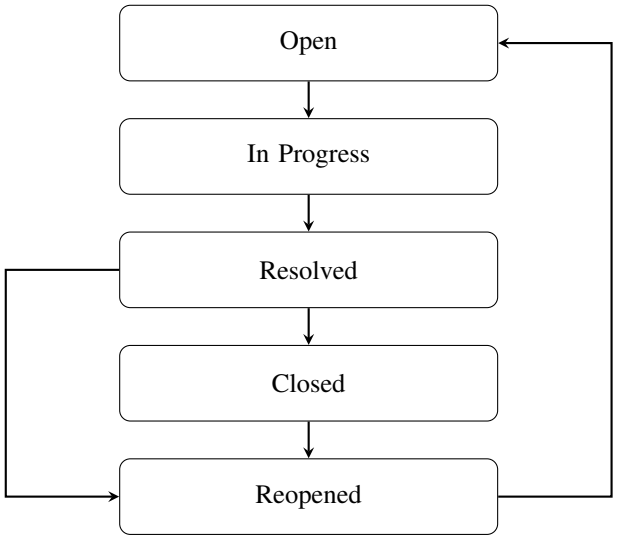

Fig. 1. CR life cycle [2].

In this context, our goal is to indicate the best ML classifier to predict the severity level of a CR in an imbalanced data scenario, based on the assessment the accuracy of three traditional ML algorithms: Neural Networks (NN), Random Forest (RF) and Support Vector Machine(SVM).

To systematize our evaluation, our experiments should answer the following research questions:

RQ 1: *Will the change request severity level change during its lifecycle?* we have investigated the accuracy of the classifiers to answering a binary response question: if CR will remain its severity level or it will change.

RQ 2: *Will the change request severity level increase, decrease or remain the same during its lifecycle*? we have investigated the accuracy of the classifiers to answering a ternary response question: if CR will remain, increase or decrease its severity level.

RQ 3: *What will the change request severity level at the end of its lifecycle?* Finally, we have investigated the accuracy of the classifiers to answering a question with more than three responses.

This paper aims to deliver the following contributions:

1) Indicate the performance of three ML algorithms to answering questions with two, three or more responses in an imbalanced scenario.
2) Suggest another way to measure the performance of ML algorithms in imbalanced data scenario besides the traditional forms used for this purpose.
3) Advance the researches on the topics discussed in this article.

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 [8] 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. [9] 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 three ML algorithms: Neural Networks, Random Forest and SVM.

A. Change Request Systems

Change Request systems [10] 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.

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).

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 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 [11]. It is composed of three basic activities [12]: 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 [13]. 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) Neural Networks: Neural Network is a learning algorithm that is inspired by the structure and functional aspects of biological neural networks [14]. Computations are structured regarding an interconnected group of artificial neurons, processing information using a connectionist approach to computation.

Modern neural networks are non-linear statistical data modeling tools. They are usually used to model complex relationships between inputs and outputs, to find patterns in data, or to capture the statistical structure in an unknown joint probability distribution between observed variables.

2) Random Forest: The Random Forest algorithm [15] 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 [12]

- 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).

3) Support Vector Machine: Support Vector Machine is considered the most popular algorithm for supervised learning and an excellent first method to testing [14]. It is a set of related supervised learning methods used for classification and regression. Each example in a set of training data is marked as belonging to one of two categories an the SVM algorithm builds a model that predicts whether a new example falls into one category or the other.

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 [11].

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 [16]: 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 [17].

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 [16]: 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 [17].

F-measure. F-measure conveys the balance between the precision and the recall and combines the two measures in an ad hoc way [11], [17].F-measure can be calculated using the formula $2 * ((precision * recall))/(precision + recall)$.

IV. Experiment

In this section, we describe our approach which was conducted following the steps in Figure 2

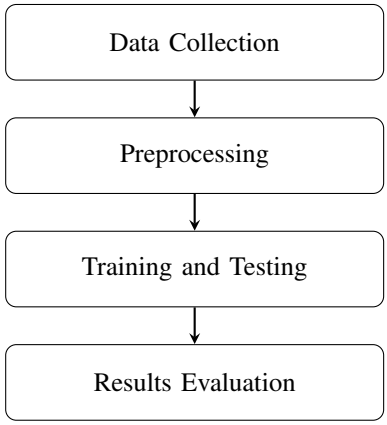

Fig. 2. Methodology used in this study.

A. Data Collection

. We have created our dataset to training and testing the machine learning from the changes requests data collected 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. It is considered a specialized and complex OSS project with many users with distinct expertise level. This fact has provided us access to change requests, in XML format, with different levels of information. From CR's long descriptions with few character lines to ones with many lines, including even code snippets and exception stack trace.

The users of HADOOP have registered and stored their CRs via Jira software[https://www.atlassian.com/software/jira]. To collect these data, we have organized its extraction into two steps: (i) the copying of CR basic data (e.g. status and resolution) in XML; and (ii) the copying of CR changes

histories in HTML, both from HADOOP web site[http://issues.apache.org]. About 10% of the CRs collected in this process changed at least once their level of severity.

We collected CR 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 7129 change requests. Figure IV-A shows three charts with requests distribution by severity level and severity level changes.

The graph (a) shows that of 7129 change requests, 3.6% had severity 1(trivial); 16.8% had severity 2(minor); 62.2% had severity 3(major), 4.0% had severity 4(critical), And 13.4% had severity 5(blocker). The graph (b) shows that 91.9% of them haven't changed their severities levels and 8.1% have changed. And the graph (c) reveals that of the 8.1% which changed their severity, 23.3% decreased it, and 76.7% increased it. We can note that dataset is clearly imbalanced.

B. Preprocessing

The CR data stored in XML and HTML files which we have copied from HADOOP website weren't in tidy data [18], i.e., it wasn't in appropriated format to train and test the machine learning. To convert it to this suitable format, we have coded and run scripts written in language R. These scripts execute the following tasks:

- Extraction from files many features: the key, the type, the status, the resolution and the long description of CR's;
- Filtering the only CR's with status equals to Closed and resolution equals to Fixed and Implemented.
- Merging the CR's features with history data to discover what CR's have changed its severity level and if there was increasing or decreasing in this level.
- Extraction from the long description initially the one hundred the most frequent words and convert them into features for each CR.
- Classifying the CR's in ascending chronological date.

C. Training and testing

.

To train and test de machine learning, we have partitioned the dataset generated after preprocessing in two distinct pieces: one part for training with 60% of the dataset and one part for testing with 40% of the dataset. In the training phase, we have used the 3-fold cross-validation technique to choose the best values for hyperparameters for each classifier algorithm to use them in the test phase.

V. Findings and Discussions

A. RQ1: Will the change request severity level change during its lifecycle?

The RQ1 is a simple binary problem, i.e., a question whose answer is true or false. The Table II shows the performance (in percentage) of the classifiers to predict the response to this issue.

We tested the classifiers with 2851(40% of 7129) change requests: 2620 have changed their severity level, and 231 haven't changed their severity level. We can observe that the three classifiers performed very closely. However, the Random Forest classifier have achieved a F1-Measure somewhat better than the two others.

TABLE II
Classifiers Performances on RQ1.

Classifier	Precision	Recall	F-Measurement
NN	93,381	98,015	95,642
RF	93,801	99,923	96,765
SVM	93,727	99,809	96,627

Regarding the Random Forest classifier we have done one step further, we have investigated its performance relating to the number of hits and errors made in answer to RQ1. The Figure V-A indicates that its accuracy was 93.681% (2676 divide by 2851).

B. RQ2: Will the change request severity level increase, decrease or remain the same during its lifecycle

The RQ2 poses a problem more difficult than a previous question. It is a question with three possible responses related to severity level: (-1) it has decreased; (0) it has remained; and (1) it has increased. The Table III shows the performance (in percentage) of the classifiers to predict the response to this issue.

TABLE III
Classifiers Performance on RQ2.

	Class	Precision	Recall	F-Measurement
NN	-1	3.703	28.571	6.557
	0	97.663	93.357	95.447
	1	22.598	38.461	28.469
	Average	41.311	53.463	43.491
RF	-1	13.111	85.714	19.672
	0	99.923	93.500	96.605
	1	28.598	90.652	46.199
	Average	47.210	89.955	54.158
SVM	-1	12,962	70,000	21,875
	0	99,923	93,902	96,819
	1	27,118	90,566	41,739
	Average	46,667	84,822	53,477

We have tested the classifiers with 2851(40% of 7129) change requests. Only now, we have three predicting situations: 2620 haven't changed their severity level, 177 have increased their severity level, and 54 have decreased their severity level. We can observe in the Table III which the classifiers also performed very closely as question 1. However, the Random Forest classifier have achieved F-Measure somewhat better than the two others.

Like as in the RQ1, we have done one step further regarding the best classifier, we have investigated its performance, observing the number of correct and incorrect answers on the test dataset as a whole. The Figure V-B indicates that its accuracy was 93.440% (2664 divide by 2851) in the RQ2 prediction.

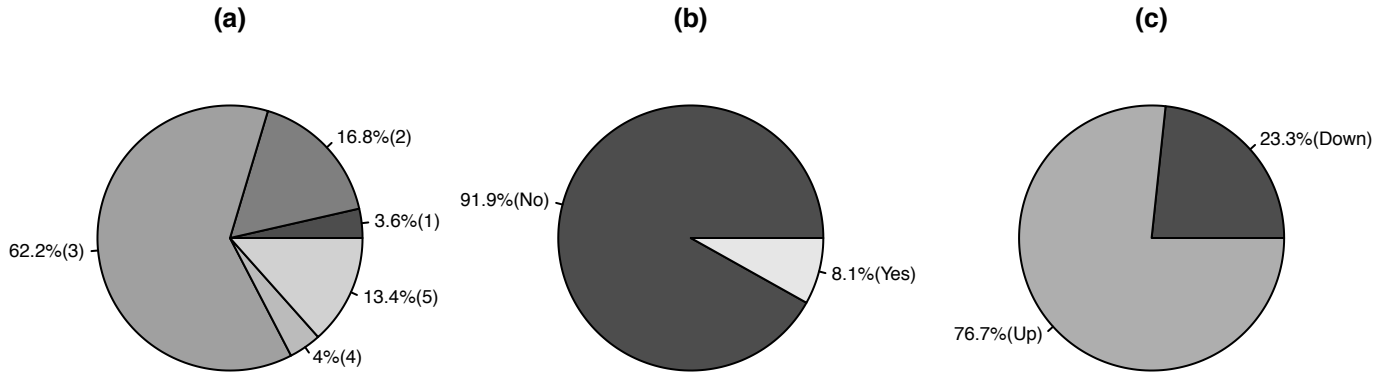

Fig. 3. Dataset distribution by severity level.

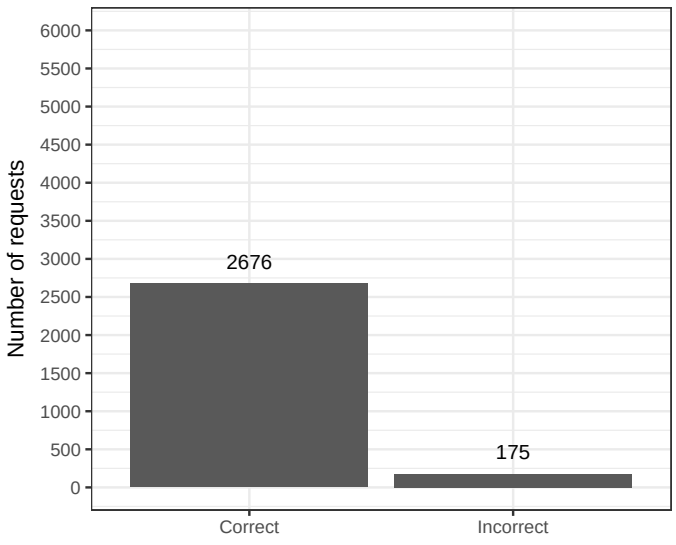

Fig. 4. Performance of Random Forest for RQ1.

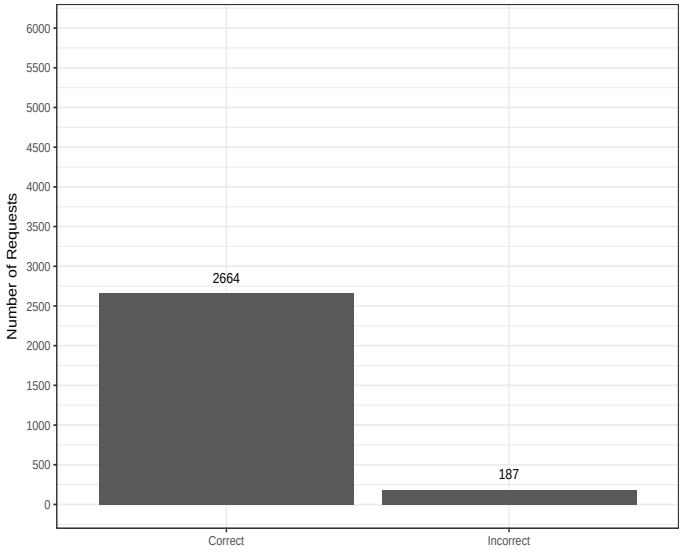

Fig. 5. Performance of Random Forest for RQ2.

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 related to severity level: (1) trivial; (2) minor; (3) major; (4) critical; and (5) blocker. The Table IV shows the performance (in percentage) of the classifiers to predict the response to this issue.

We have tested the classifiers with 2851(40% of 7129) change requests. Only now, we have six predicting situations: 101 are trivial; 479 are minor; 1774 are major; 112 are critical; 382 are a blocker. We can observe in the Table III which the classifiers also performed very closely as questions 1 and 2. However, the Random Forest classifier have achieved a F1-Measure somewhat better than the two others.

Like as in the RQ1 and RQ2, we have done one step further regarding the best classifier, we have investigated its performance, observing the number of correct and incorrect answers on the test dataset as a whole. 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 68,08% (1941 divide by 2851).

Although we can consider that accuracy a good value, the graph (a) shows that user accuracy was 91.18% and graph (c) also shows that 745 change requests, the PE was greater than UE. From a business vision, the predictor have not show suitable performance.

VI. Threats to Validity

Runeson [19] recommends that threats to validity should be considered under four aspects: construct validity; internal

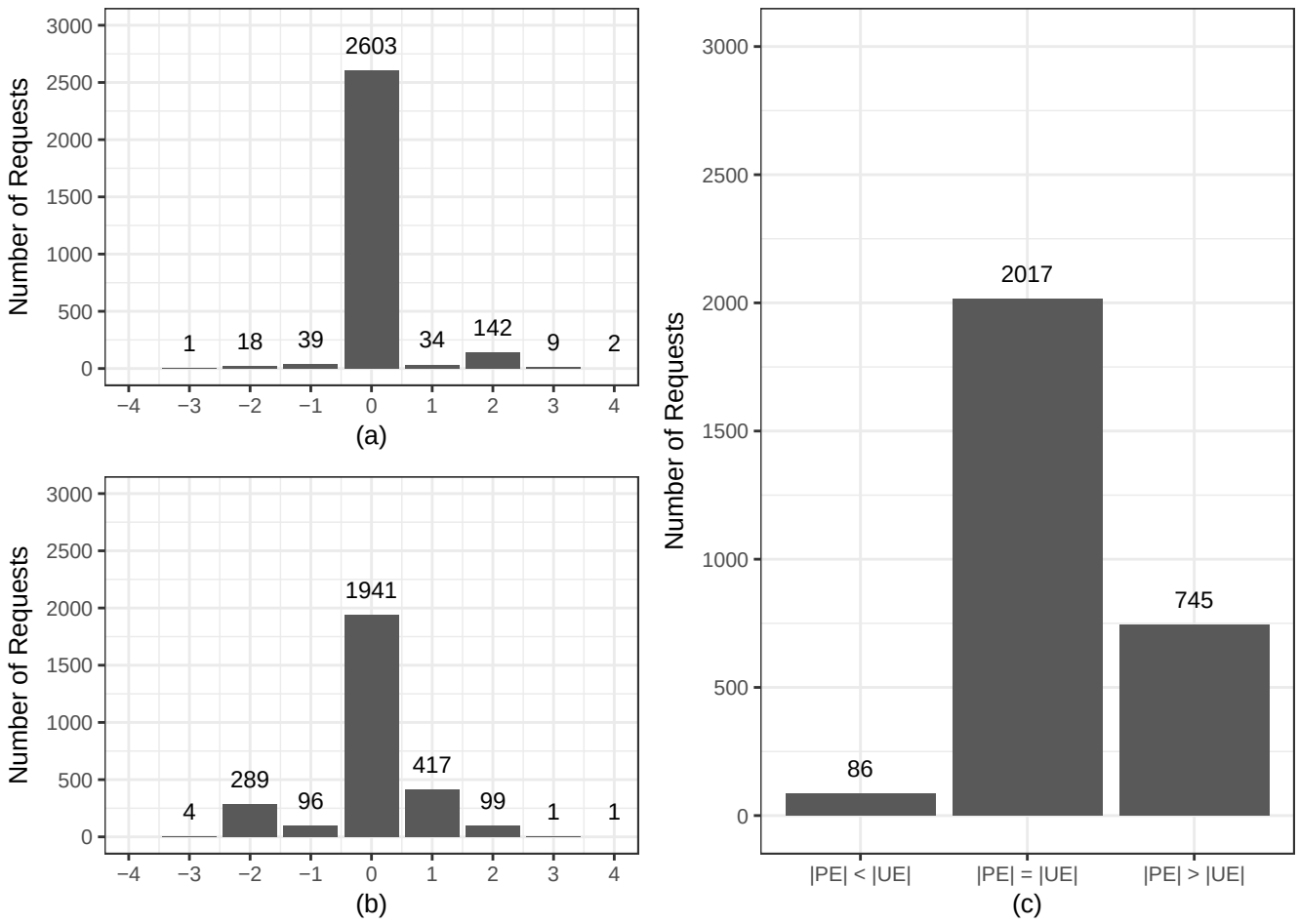

Fig. 6. Performance of Random Forest for RQ3.

validity; external validity; and reliability.

Construct validity. Despite existing others metrics to evaluate classifiers [16], 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], [5], [8], [9].

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], [9].

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), nnet (a neural network R implementation), randomForest (a random forest R implementation) and e1071 (a SVM R implentation).

VII. Conclusion and Future Work

In this paper, we assess the Neural Network, Random Forest and Support Vector Machine, three popular ML algorithms 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 93.681% and 93.440% respectively. However, it has provided the accuracy (around 68,08%) to predict the final last severity level on 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(PPP) of Pontifical Catholic University of Minas Gerais (PUC MG).

TABLE IV
CLASSIFIERS PERFORMANCE ON RQ3.

	Class	Precision	Recall	F-Measurement
NN	1	4.950	29.411	8.474
	2	18.997	34.469	24.495
	3	88.726	66.273	75.873
	4	16.964	37.254	23.312
	5	17.015	46.099	24.856
	Average	29.330	39.606	31.402
RF	1	4.950	83.333	9.345
	2	16.283	76.470	26.850
	3	98.308	67.387	79.963
	4	30.357	100.000	46.575
	5	25.130	81.355	38.400
	Average	35.005	81.709	40.226
SVM	1	6.930	70,000	12.612
	2	16.283	77.227	26.896
	3	95.478	67.166	78.857
	4	30.357	97.142	46.258
	5	23.036	87.128	36.438
	Average	34.416	79.7326	40.212

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] N. V. Chawla, "Data Mining for Imbalanced Datasets: An Overview," in *Data Mining and Knowledge Discovery Handbook*. Boston, MA: Springer US, 2009, pp. 875–886.

[7] Y. Tian, D. Lo, X. Xia, and C. Sun, "Automated prediction of bug report priority using multi-factor analysis," *Empirical Software Engineering*, vol. 20, no. 5, pp. 1354–1383, oct 2015.

[8] 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.

[9] 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.

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

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

[12] 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

[13] 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.

[14] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2010.

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

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

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

[18] E. de Jonge and M. van der Loo, "An introduction to data cleaning with R," *Statistics Netherlands*, p. 53, 2013. [Online]. Available: http://cran.r-project.org/doc/contrib/de{_}Jonge+van{_}der{_}Loo-Introduction{_}to{_}data{_}cleaning{_}with{_}R.pdf

[19] 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.