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

Author1
Affiliation1

Author2
Affiliation2

***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. This is specially true in OSS, 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 CRs [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 assigning the person responsible for the CR.

The severity level information is recognized as a critical variable in the equation to estimate a prioritization of CRs [3]. It defines how soon the CR needs to be addressed [4]. However, the severity level assignment remains mostly a manual process which relies only on the experience and expertise of the person who has opened the CR [1], [3], [4]. As a consequence, it is a process with high degree of subjectivity, and it may be quite error-prone.

The number of CRs in large and medium software OSS projects [5] is frequently very large. Severity level shifts throughout CR lifecycle may have an adverse effect on the planning of maintenance activities. For example, the maintenance team could be assigned to address less significant CRs before most important ones. There has been reports of efforts to implement intelligent software assistants to help developers and maintenance personnel in defining more accurately the field values in a CR form. Currently, Machine Learning techniques have become a popular method to address this issue and there is quite a few publications in this area in the literature [1].

Machine Learning (ML) techniques have been successfully applied in solving real problems in many areas of knowledge, including those related to CR systems, such as duplication and assignment of CR [1]. However, the accuracy of ML algorithms may be affected by imbalanced datasets [6] —a recurring critical problem in CR repositories [7]. For example, more than 60% of CRs may have a "major" severity level. In addition to this problem, most publications are still focused in predicting the severity level of CRs and none of them have been implemented into popular tools like as Bugzilla, Jira and Redmine. [1]. Furthermore, many have used proprietary and/or not public ML algorithms. Therefore, there is still a clear need of advances in this knowledge area, specially broadening the reach of research questions and including more popular and open OSS and ML algorithms.

In this context, the general purpose of our research is to develop an intelligent ML based assistant to help developers and maintenance personnel in the OSS maintenance activies. In this current article, our specific goals are:

G_1 : Evaluate the performance of traditional ML algorithms in the prediction of CR severity level;
G_2 : Identify a suitable algorithm to perform such prediction in a scenario where imbalanced data is natural;
G_3 : Propose a new metric to compare the performance of the software ML system and the user, in predicting or assigning the final CR Level.

In order to meet these goals, this research will work with the following research questions:

RQ_1 : *Will CR severity level change during its lifecycle?*
RQ_2 : *Will the CR severity level increase, decrease or remain*

the same during its lifecycle?

RQ_3 : *What will the CR severity level at the end of its lifecycle?*

The contributions of our reasearch are:

- Indicate the performance of three ML algorithms to answering questions with two, three or more responses in an imbalanced scenario.
- Propose a new way to measure the performance of ML algorithms in imbalanced data scenario besides the traditional forms used for this purpose.
- Advance the researches on the topics discussed in this article.

The article is organized as follows. Section II presents related work that are relevant to our research. Section III provides the information background about CR systems, text mining and machine learning techniques necessary to understand our approach. Section IV describes our work. Section V presents final findings and discussion. Finally, Sections VII and VIII present the threats to validity of this research, and conclusions and future work, respectively.

II. Related Work

This section presents relevant articles in the area of mining open system repositories, aiming at extracting data and using ML techniques to predict several maintenance properties.

Menzies [8] have developed a method, named SEVERIS (SEVERity ISsue assessment), for evaluating the severity of CRs. 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 predict CR severity level in five projects managed by the Project and Issue Tracking System (PITS), an issue tracker system used by NASA. The predictions had the following F-measures stratified by severity level (range in projects is shown): (2) 78%-86%; (3) 68%-98%; (4) 86%-92%. Severity level 1 and 5 were excluded from experiment.

Lamkanfi et al. [4] have developed an approach to predict if severity of bug report is non-severe (severity levels: 1 or 2) or severe (severity levels: 4 or 5) based on text mining algorithms (tokenization, stop word removal, stemming) and on the Naïve Bayes machine learning algorithm. They have validated their approach with data from three open source project (Mozilla, Eclipse, and GNOME). The article reports that a training set with approximately 500 CRs per severity level is sufficient to make predictions with reasonable accuracy (precision and recall in the range 0.65-0.75 with Mozilla and Eclipse; 0.70-0.85 with GNOME).

In another work, Lamkanfi et al. [5], the authors compared the accuracy of four machine-learning algorithms (Naïve Bayes Multinomial, K-Nearest Neighbor, and Support Vector Machine) to predict if a bug report is severe or non-severe. According to the article, the Naïve Bayes Multinomial algorithm provided the best performance among the four studied algorithms (0.75-0.93 Area Under Curve (AUC)).

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 have build decision trees for each project to predict whether a bug will be a blocking bug or not (F-measures in the range 15-42%).

Tian et al. [3] have develop a method to predict the severity level of new CRs based on similar CRs reported in the past. The comparison between old and new CRs was implemented by the BM25 similarity function. This method was applied to Mozilla, Eclipse and OpenOffice projects over more than 250,000 CR extracted from Bugzilla (F-measure in the range 13.9-65.3% for Mozilla; 8.6-58% for Eclipse; and 12.3-74% for OpenOffice).

III. Background

This section briefly comments of basic concepts necessary to comprehend this research area, namely CR Systems, Text Mining, Machine Learning, and ML evaluation metrics.

Data used in this research area are usually extracted from the so called CR Systems, or Gug Tracking Systems. Popular CR Systems are Bugzilla, Jira, and Redmine [3]. Additional information can be found in [10].

Two techniques are frequently used in this research area: Text Mining [11] [12] and Machine Learning (ML) [12] [13] [14] [15]. Detailing of these techniques are outside the scope of this paper.

Finally, it is worth mentioning the specific metrics used in this research are for assessing prediction performance. The three most common performance measures for evaluating the accuracy of classification algorithms are precision, recall, and F-measure, described as follows.

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

This section describes the experiment conducted to address the Research Questions. As in typical methodologies used in ML studies, it comprises the followint steps: Data Collection (IV-A), Data Preprocessing (Section IV-B), and Training and Testing (Section IV-C).

A. Data Collection

This step in the experimental research encompasses sellecting a FLOSS to serve as the data source, studying and interpreting its data structure, and finally extracting relevant data from its repository (feature extraction). In this research, Hadoop, Linux, and Mozilla Open Systems were considered as potential Open Source Systems to study. In a first approximation, Hadoop was sellected as a data source of CR records, due to the fact it is open, well stablished, has a considerable number of CRs already registered, uses standard repositories, and was under study by other researchers in our research group. According to [hadoop.apache.org], HADOOP is a "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 different levels of expertise. Its CR repository allows for access to all CR contents in XML format. Everything is available (except change history), from CR long description field (with lines with few characters to ones with many lines), including code snippets and exception stack trace.

CRs in HADOOP are stored in a Jira based repository [https://www.atlassian.com/software/jira]. Two steps are used to perform data extraction from HADOOP web site[http://issues.apache.org]: (i) copying CR basic data (e.g. status and resolution) from XML contents; and (ii) copying CR changes history from external HTML pages (this may be important for learning). About 10% of the CRs collected in this process changed their level of severity at least once.

CR record data from February 01, 2006 to January 18, 2017 was collected. Only requests from the common module (identifier = HADOOP-*) were considered for retrieval. The total number of CR records retrieved after preprocessing was 7129.

Figure **??** shows how the 7129 retrieved CR records were distributed in terms of severity level and severity level change. Figure **??**(a) shows the severity level distribution: 3.6% have severity 1 (trivial); 16.8% have severity 2 (minor); 62.2% have severity 3 (major), 4.0% have severity 4 (critical), and 13.4% have severity 5 (blocker). Figure **??**(b) shows that only 8.1% have changed their severities levels during the CR lifecycle. Finally, Figure **??**(c) reveals that of these 8.1% CRs which changed their severity, 23.3% decreased it, and 76.7% increased it. One can see that this dataset is clearly imbalanced, posing addicional difficulty to the application of the ML methodology.

B. Preprocessing

The raw data previously collected from the Hadoop CR Repository was not properly structured to serve as input to ML algorithms, it was in tidy data format [18]. The classical way to address this problem is to run preprocessing procedures to extract, organize and structure relevant information out of the raw data. Specific scripts were written in R language to acomplish this. Preprocessing tasks were executed as follows:

- Extraction of relevant features: key, type, status, resolution status, and long description of CRs;
- Filtering CRs with status equals to Closed and resolution equals to Fixed and Implemented.
- Merging CR features with their change history data. This additional information allows for the identification of CRs that have changed severity level during the CR lifecycle, and furthermore, if they have changed for better (decrease) or worse (increase).
- Performing text mining in the long description field to identify the 100 most frequent words. This information is then converted into features for each CR.
- Sorting the CRs in ascending chronological date.

C. Training and testing

.

Training and testing steps starts with partitioning the already preprocessed dataset in two disjoint subsets: a subset for training, with 60% of the CRs, and a subset for testing, with the remaing 40% of the CRs. In the training phase, we have used the cross-validation with 3-fold techniqueto 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 CR severity level change during its lifecycle?

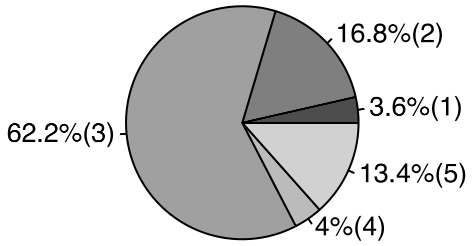
The RQ1 is a simple binary problem, i.e., a question whose answer is true or false. The Table I shows the performance (in percentage) of the classifiers to predict the response to this issue.

TABLE I: Classifiers Performances on RQ1.

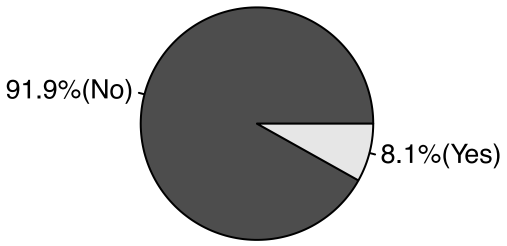
Classifier	Precision	Recall	F-Measure
NN	93,381	98,015	95,642
RF	93,801	99,923	96,765
SVM	93,727	99,809	96,627

We tested the classifiers with 2851 (40% of 7129) CR: 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.

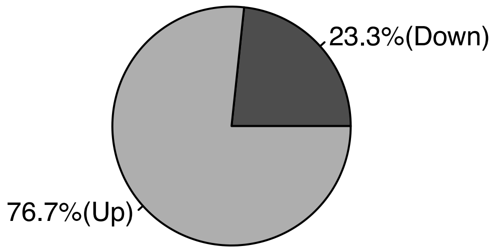
Regarding the Random Forest classifier we have done one step further, we have investigated its performance relating to the number of hits and errors made in answer to RQ1. The Figure **??** indicates that its accuracy was 93.681% (2676 divide by 2851).



(a) 1a



(b) 1b



(c) 1c

Fig. 1: plots of....

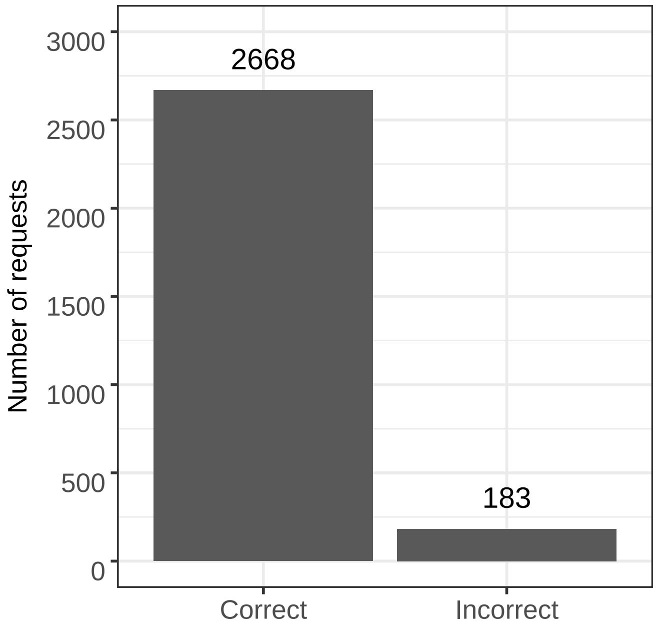


Fig. 2: Performance of Random Forest for RQ1.

B. RQ2: Will the CR 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 II shows the performance (in percentage) of the classifiers to predict the response to this issue.

TABLE II: 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) CRs. 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 II 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 **??** indicates that its accuracy was 93.440% (2664 divide by 2851) in the RQ2 prediction.

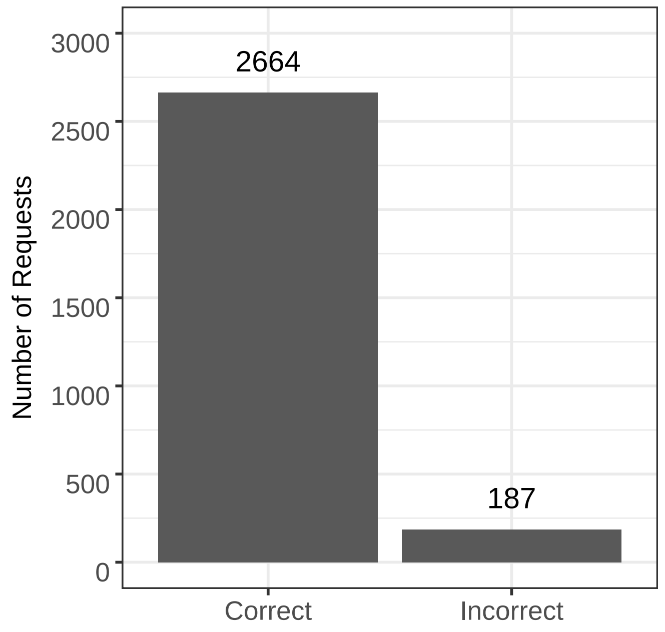


Fig. 3: Performance of Random Forest for RQ2.

C. RQ3: What will the CR 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.

TABLE III: 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

We have tested the classifiers with 2851 (40% of 7129) CRs. 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 II 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 **??** 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 CR 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 CRs, 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 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 CR. This assumption finds echo or support in [3], [4]

External validity. We have considered one single repository and we have extracted 8858 CRs 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

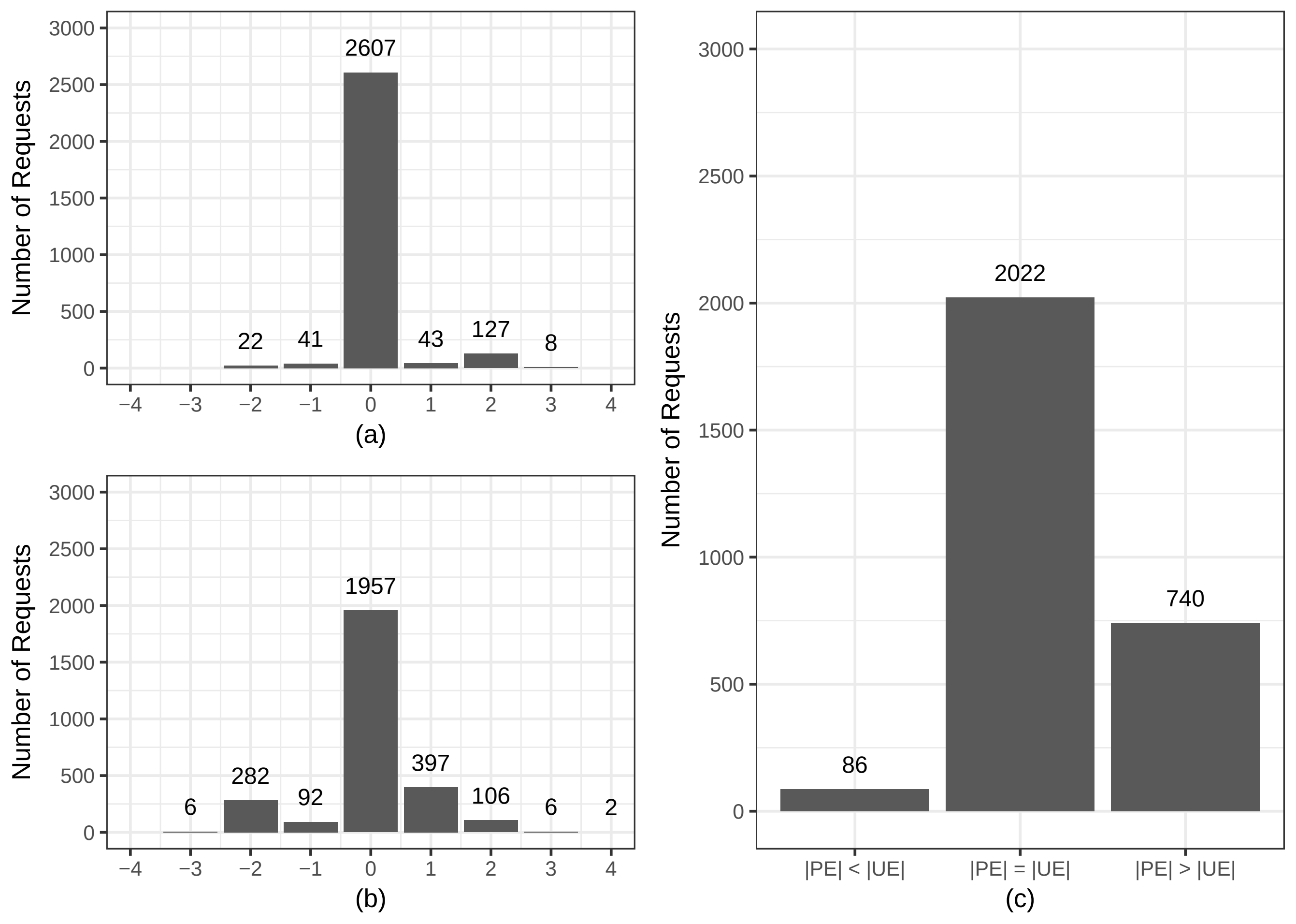


Fig. 4: Performance of Random Forest for RQ3.

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).

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.

TABLE IV: Classifiers Performance on RQ3.

	Research Questions	Repositories	F-measure	Algorithms	Observations
Menzies [8]	NA	pitsA	14-71	RIPPER	
		pitsB	42-90	RIPPER	
		pitsC	53-92	RIPPER	
		pitsD	87-99	RIPPER	
		pitsE	8-80	RIPPER	
Lamkanfi [4]		4.950	83.333	9.345	0
		16.283	76.470	26.850	0
		98.308	67.387	79.963	0
		30.357	100.000	46.575	0
		25.130	81.355	38.400	0
Lamkanfi [5]		6.930	70,000	12.612	0
		16.283	77.227	26.896	0
		95.478	67.166	78.857	0
		30.357	97.142	46.258	0
		23.036	87.128	36.438	0
Valdivia [9]		6.930	70,000	12.612	0
		16.283	77.227	26.896	0
		95.478	67.166	78.857	0
		30.357	97.142	46.258	0
		23.036	87.128	36.438	0
Tian [3]		6.930	70,000	12.612	0
		16.283	77.227	26.896	0
		95.478	67.166	78.857	0
		30.357	97.142	46.258	0
		23.036	87.128	36.438	0
Ours		6.930	70,000	12.612	0
		16.283	77.227	26.896	0
		95.478	67.166	78.857	0
		30.357	97.142	46.258	0
		23.036	87.128	36.438	0

[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.