mühendislikdergisi
Dicle Üniversitesi Mühendislik Fakültesi

Dicle University Journal of Engineering (DUJE)

web: http://dergipark.gov.tr/dumf

Research Article

Hybrid Recommendation System Approach for appropriate developer selection in Bug Repositories

Mohanad Al-imari [1,*], Sefer Kurnaz [2], Jalal S. H. Al-bayati

[1] Altınbaş University, 0000-0001-6112-7206
[2] Altınbaş University.
[3] University of Baghdad. 0000-0001-7921-0489

ARTICLE INFO

Article history:

Received 29 October 2020
Received in revised form 7 March 2021
Accepted 18 Mart 2021
Available online 22 June 2021

Keywords:

Open source Bug Repositories; Hybrid machine learning approach; decision trees; Naive Bayes; Random Forest; Neural Networks; feature selection

Doi: 10.24012/dumf.818164

ABSTRACT

The essential destination of this research is to develop a hybrid recommendation system methodology to enhance the overall performance accuracy of such existed systems, this recommendation approach normally utilized to assign or propose a few counted numbers of programmers or developers that capable of resolving system's bug reports generated automatically from an open source bug repository, meaning the system decides which programmers or developers should be taken into account to be in charge of finding a solution the bugs mentioned in the bug's report. The definition of the bug selection problems in bug repositories are the activities that developers achieve within program maintenance to fix some specific bugs. Because of lot of bugs are created daily, many developers required are quite large, therefore it is difficult to specify the accurate programmers or developers to find a solution for the issues for specific bug inside the code. The article also aims to improve the accuracy results obtained than existed traditional approaches for this purpose. Besides, we have considered the case of prioritization of system developers, the case can be utilized to find an appropriate grade of developers' achievements as prior knowledge to assist the system in assigning of bug report issue. The results have found that the importance of developers could support the bug triage worker more and help software tasks to solve the bugs fast and within required time during development and support cycles of the software.

Introduction

In development of any software project including open source projects, there are several cycles, one of these cycles is the maintenance and support. With maintenance cycles, various maintenance bug reports are registered in the system which represents more than 70% of overall bugs. Programmers might loss more than half of their time for recognizing and resolve bugs. In addition to that, number of software bugs has become extremely high. For Instance, in eclipse project, more than twenty reports were sent to bug system in the release time. Also, project of Debian is sending approximately more than twenty reports were sent to bug system in the release time. Also, project of Debian is sending approximately 140 reports in a daily manner [1]. These types of systems should help developers to control and manage bug reports and resolve them faster and easily. On the other hand, the assignment of the best programmer to a specific bug is still in the focus of scholarly attention in software development researches. Open source software usually supports bug repository, this bug box contains reports that automatically sent to developers that should fix and solve bugs.

* Corresponding author
Mohanad, Al-imari
✉ mohanad.alimari@ogr.altinbas.edu.tr

Please cite this article in press as M. Al-imari, S. Kurnaz, J. S. H. Al-bayati “Hybrid Recommendation System Approach for appropriate developer selection in Bug Repositories”, DUJE, vol. 12, no. 3, pp. 471-477, June 2021.

The bug file has to send to the developer or category of specific developers, and they will be responsible for resolving and fixing the bug. Using the recommendation system, whenever a new bug appears in the system, the recommendation system's engine (classifier) indicates the category of convenient developers to resolve the bug. As a result, this potential advantage will help bug triage workers to decide who should fix a bug report [2]. These bug warehouses are indexed and referenced to a section of problems proceeding, which creates a big database of any issues reported by software designers or programmers in a project. The open bug repositories are called open source, in which anyone can access these reports. These repositories have an essential role in those projects due to full access permission of programmer communities to view, find, share, edit, and fix during the project's development progress. We have used a dataset from the same open-source project (eclipse project) to build a hybrid classifier recommendation system using Naive Bayes, decision trees, random forest, and neural networks. Some Scholars also used the Support Vector Machines method and tried some unsupervised learning algorithms. Besides that, they used datasets from the firefox open source project and GCC project.

Literature Review

We briefly mention some approaches for recommendation systems for bug resolves proposed by some scholars for the background. Many models and simulations have been created and presented for finding the best developers for bug fixing in bug systems. Xuan et al. proposed a model that depends on social knowledge in the bug reports of Eclipse and Mozilla bug repositories utilizing the social network model [3]. Shokripour et al. utilized an automatic bug assignment using a time parameter. The approach considered time metadata for the weighted expression. The repetition of the word in reports is fixed by utilizing a technique called tf-idf [4]. Xia et al. have proposed an improved model of Linear Discriminant for bug assignment. They offered an incremental change approach as a learning approach called TopicMiner; this TopicMiner approach detects the appropriate bug reports for the proper developer concerning the bug's distribution and likeness. The likeness is created as a relationship between bug fix programmer and distribution [5].

These bug warehouses are indexed and referenced to the section of problems proceeding, which creates a big database of any issues reported by software designers or programmers in a project. The open bug repositories are called open source, in which anyone can access these reports. These repositories have an essential role in those projects due to full access permission of programmer communities to view, find, share, edit, and fix during the project's development progress. Yang G. et al. [15] proposed a semi-automatic bug triage based on the Topic model from 30,000 bug reports from several open-source projects, and they achieved practical analysis with approximated 52% accuracy.

Data Description and Features

The dataset had been extracted from the eclipse bug repository until the year 2009. The inputs of the system are the features of the dataset that comes from the eclipse platform bug measures dataset, which has (7700×50) numbers. The first column of the data contains the bug id number. The last column contains the class labels from 1 to 7. Each class of the dataset represents a group of developers according to their companies assigned before to resolve bugs in some components of the software project, which means we have 48 features. For these features, we have utilized many machine learning algorithms by using Matlab and weka software. For classifying these features according to their class labels, we have validated them by using cross-validation. The dataset is multivariate. Tab. 1 below presents developer emails in each category and instances in each category. Each type of dataset represents a class or group of developers according to their companies and the number of bugs they have fixed/resolved.

Table 1. Developers and Instances in each class

Class	**Number of developers and instances**	
	Number of developers	***Number of Instances***
1	10	1206

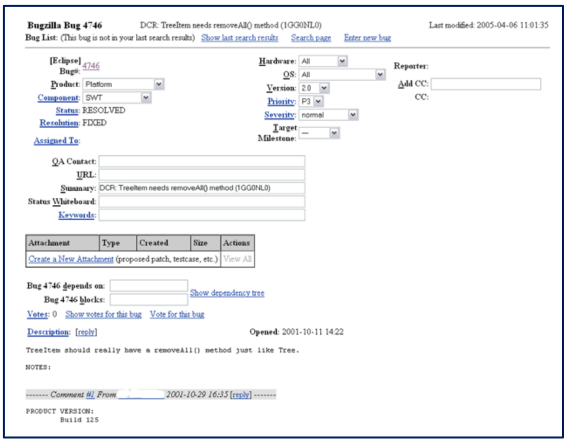
Class	Number of developers and instances	
	Number of developers	*Number of Instances*
2	10	1173
3	19	735
4	60	845
5	11	1611
6	20	1281
7	8	919
Total	138	7700

The classes of the dataset represent a group of developers based on their companies.

The Features of the Eclipse dataset are:

bugID; component;; assigneeEmail;os; platform; milestone; nrKeywords; nrDependentBugs; peopleCC;openedhoursOpenedBeforeNextRelease;lastModified;priority;severity;resolution;firstFix;lastFix;hoursLastFixBeforeNextRelease;hoursLastFixAfterPreviousRelease;status;firstActivity;nrActivities;lastResolution;nrComments;hoursToLastFix;hoursToLastResolution;monthOpened;yearOpened;monthYearOpened;monthYearLastFixed.

These features represent the details and contents of each bug report. Below is an example of a bug report:



Bugzilla Bug 4746 DCR: TreeItem needs removeAll() method (1GG0NL0) Last modified: 2005-04-06 11:01:35

Bug List: (This bug is not in your last search results) Show last search results Search page Enter new bug

[Eclipse] Bug#: 4746
Product: Platform
Component: SWT
Status: RESOLVED
Resolution: FIXED
Assigned To:

Hardware: All
OS: All
Version: 2.0
Priority: P3
Severity: normal
Target Milestone: —

Reporter:
Add CC:
CC:

QA Contact:
URL:
Summary: DCR: TreeItem needs removeAll() method (1GG0NL0)
Status Whiteboard:
Keywords:

Attachment	Type	Created	Size	Actions
Create a New Attachment (proposed patch, testcase, etc.)				View All

Bug 4746 depends on:
Bug 4746 blocks:
Show dependency tree

Votes: 0 Show votes for this bug Vote for this bug

Description: [reply] Opened: 2001-10-11 14:22

```
TreeItem should really have a removeAll() method just like Tree.

NOTES:
```

------- *Comment #1 From 2001-10-29 16:35* [reply] -------

```
PRODUCT VERSION:
        Build 125
```

Figure 1. Bug report

Feature Selection

The datasets in which they are utilized for developers' assignments might give us inaccurate scores, and with time, it might overrun more project resources. So, we have to select a specific type of information which is called features in the machine learning community; this information should present the features in which they are redundant and inconsistent and affect the classification. In order to improve the accuracy of results, these redundant features should be removed [2]. Using feature selection approaches to check the datasets first, such as acquiring the information gain of these features, can be removed manually by eliminating the features with less information gain. In our case, we have eliminated the last 20 features (had lowest information gain) that had a negative effect on my classification results. Figure 2 shows the information gain results of the features.

```
=== Attribute selection 10 fold cross-validation (stratified), seed: 1 ===

average merit     average rank  attribute
 0.65  +- 0.003     1   +- 0       1 component3Days
 0.474 +- 0.004     2.3 +- 0.46    3 PredictedProbability_2
 0.474 +- 0.004     2.7 +- 0.46    2 PredictedProbability_1
 0.362 +- 0.002     4   +- 0       4 bugID
 0.241 +- 0.003     5.3 +- 0.46    5 os3Days
 0.239 +- 0.003     5.7 +- 0.46    6 PredictedProbability_1_1
 0.223 +- 0.003     7   +- 0       7 PredictedProbability_2_1
 0.119 +- 0.002     8   +- 0       8 status30Days
 0.076 +- 0.002     9.1 +- 0.3     9 nrActivities30Days
 0.072 +- 0.002    10.5 +- 0.81   10 nrActivities
 0.072 +- 0.001    10.6 +- 0.66   11 hoursLastFixAfterPreviousRelease
 0.07  +- 0.001    11.8 +- 0.4    12 nrActivities14Days
 0.066 +- 0.001    13   +- 0      13 nrActivities7Days
 0.059 +- 0.002    14.7 +- 0.9    14 status3Days
 0.057 +- 0.001    15   +- 0.77   16 nrActivities3Days
 0.058 +- 0.001    15.3 +- 0.64   15 hoursLastFixBeforeNextRelease
 0.045 +- 0.001    17   +- 0      17 nrActivities1Days
 0.032 +- 0.001    18   +- 0      18 hoursToLastFix
 0.027 +- 0.001    19   +- 0      19 monthOpened
 0.025 +- 0.001    20.1 +- 0.3    20 hoursToLastResolution
 0.022 +- 0.001    22   +- 1.1    22 peopleCC
 0.023 +- 0.001    22   +- 1.1    21 priority30Days
 0.022 +- 0.001    22.8 +- 1.6    23 filter_$
 0.021 +- 0.001    24.1 +- 1.3    24 nrPeopleCC30Days
 0.021 +- 0.001    24.9 +- 1.22   25 nrPeopleCC14Days
 0.02  +- 0.001    26.5 +- 1.8    26 hTLFix2Bins7Days
 0.019 +- 0.001    27.4 +- 1.62   27 hTLFix2Bins3Days
 0.019 +- 0.001    27.6 +- 1.43   28 nrPeopleCC7Days
 0.018 +- 0.001    29   +- 1.67   29 nrComments30Days
 0.018 +- 0.001    29.6 +- 1.36   30 nrPeopleCC3Days
 0.017 +- 0.001    30.9 +- 1.45   31 nrPeopleCC1Days
 0.016 +- 0.001    32   +- 1.1    33 hTLFix2Bins14Days
 0.016 +- 0.001    32.1 +- 1.04   32 platform3Days
 0.014 +- 0.001    34   +- 0      34 hTLFix2Bins1Days
 0.012 +- 0.001    35   +- 0      35 hTLFix2Bins30Days
 0.01  +- 0.001    36.6 +- 0.92   36 nrComments
 0.009 +- 0.001    37.1 +- 1.3    37 hTLFix2Bins0Days
 0.009 +- 0.001    37.9 +- 0.83   39 initPeopleCC
 0.009 +- 0.001    38.7 +- 0.46   38 nrPeopleCC0Days
 0.008 +- 0.001    39.8 +- 0.75   40 nrComments14Days
 0.007 +- 0.001    41.2 +- 0.6    41 severity3Days
 0.006 +- 0       41.7 +- 0.46   42 nrComments7Days
 0     +- 0       43.1 +- 0.3    47 nrDependentBugs
 0     +- 0       43.9 +- 0.3    45 nrComments1Days
 0     +- 0       45   +- 0      46 nrActivities0Days
 0     +- 0       46   +- 0      48 nrKeywords
 0     +- 0       47   +- 0      44 nrComments0Days
 0     +- 0       48   +- 0      43 nrComments3Days
 0     +- 0       49   +- 0      49 resolution3Days
```

Figure 2. Features' Information gain

Methodology

Many machine learning algorithms will be utilized for implementing the recommendation system, such as Naive Bayes, Decision Trees, and Support Vector Machines. They might try some unsupervised machine learning algorithms like Expectation Maximization [2]. First, we have been attempting a hybrid of two machine learning algorithms as recommendation system engines such as Naive Bayes, Decision Trees. The Feature selection algorithms will be used on the training data set to get the best-related features. We have tried the forward selection algorithm to choose the best features and applied the Naive Bayes algorithm [6]. I used the Naive Bayes algorithm to classify the data set using MATLAB code [7] and WEKA software [8]. We also have applied the decision tree algorithm to the filtered dataset using the WEKA software with 0.25 confidence factor is utilized for pruning that lower values occur more pruning and two minNumObj is minimum instances per leaf and three numFolds that determines the amount of data utilized for minimizing the error in pruning. Only one fold is being used for pruning, and the others are utilized for growing the tree.

- *Classification Tree Rules*:

For node *m*, N_m examples in *m* domain, N^i_m for Class$_i$: [2]

$$\hat{P}(C_i \mid \mathbf{x}, m) \equiv p^i_m = \frac{N^i_m}{N_m} \tag{1}$$

If p^i_m is equal to zero or one, then Node *m* is called pure, and we mathematically calculate the Impurity in which it is entropy: [2]

$$\mathcal{I}_m = -\sum_{i=1}^{K} p^i_m \log_2 p^i_m \tag{2}$$

Create a leaf and stop if node *m* is pure, otherwise continue split recursively. Calculate the Impurity after split: N_{mj} from N_m taking branch *j*. N^i_{mj} was taken from C_i. Finding the variables and separating that minimum Impurity (among all variables and splitting positions for numeric variables) [6].

- *Random Forest*

This is an ensemble approach that receives a subset of dataset observations and a subset of variables as branches of its tree to create many decision trees (An ensemble depends on a set of one-by-one trained models (neural networks, support vector machines, decision trees as an example). In general, it gives us high classification accuracy than other models such as Naïve Bayes and support vector machines [9]. It creates multiple decision trees and merges them to improve accuracy results and stable prediction trying to avoid underfitting and overfitting issues. The outcome is taken from the maximum voting of independent judges list, and the final prediction should be better than the best individual judge.

If m classifiers are to be created, they are created in sequence so that one model is made in one iteration. For creating a classifier Ci model, for example, weights of training instances are updated depending on the accuracy results of classifier Ci1. The model classifiers created by boosting usually are dependent [10].

To classify a new sample, it is run towards the trees that created the forest. Each tree has some classification score for the new sample in which it will be registered as a vote. The votes from all trees are merged, and the class in which the maximum votes are considered, which is called majority voting, is stated as the classification result of this new sample. As proposed by Breiman [8], these random forest approaches utilize majority voting as the voting model for classification. All experiments are executed concerning the voting technique. We have been used the random forest model, and it should give us good classification accuracy results with respect to some articles [9]. The random forest model contains many tree classifier models like h1(x), h2(x), . . ., hK (x), K for tree number. Each classifier model is created utilizing a bootstrap replication of the training dataset and votes for one class. A test sample is classified by the top votes class's class label as it is called the majority voting [11].

The parameters that we will utilize of the Random Forest are:

- The tree's maximum depth, for unlimited, zero is put max depth = 0.

- numFeatures = 0; The number of features to be utilized randomly; If numFeatures< 1, logM+1 is used, M is the input's number. We have tried different numbers of features like 10 or 20, but that did not cause a real effect on accuracy results.

Numbers = 135; the number of trees to be created.

We have created a model of a Random forest, which consists of 135 trees. Each tree is constructed by taking into consideration five random features with an OOB error (out of the bag) is 0.3347. The Out of Bag is explained as follows: after implementing the classifiers model, which is five trees, for each x_i, y_i dimensions in the training setT_s, Choosing all T_K without including x_i, y_i. This small subset is a set of bootstrap sets that do not have a specific register from T_s Dataset, this set is called out of bag samples.

These sets are called N subsets, one for each data register in the training dataset T_s . Out of bag model classifier is the merge of votes only over T_K sets such that it does not include x_i, y_i. For the generalization error, Out of bag estimations error is the error ratio of the out-of-bag model classifier on the training set T_s as compare with known class labels y_i [13].

The study of error speculates for bagged model classifiers [13] gives empirical proof that the OOB presume it has good accuracy while utilizing a testing dataset with the same size as the training dataset. Thus, utilizing OOB error speculation eliminate the requirement of using a dataset apart from the test dataset.

- *Neural Networks*

This network-based approach was developed by the biological networks that model the attitude of the human brain system. This approach typically learns to achieve different tasks by considering samples of data called the training data without specific task regulations. For instance, in image analysis and detection, it knows to detect images that contain cars by training some sample images and labeled manually as "car" or "not car", and by utilizing the approach, it will output to detect cars from other unrelated images. It can be done without any prior knowledge. It automatically creates and identifies characteristics from the samples entered into the approach.

The neural networks contain connected nodes called neurons, which represent neurons in the human brain. Each node is synapses in a biological brain which transmits a signal to other neurons (nodes). In created artificial neuron (node), the same attitude is utilized by receiving a signal, processing it, and send a signal to nodes connected to each one of them.

For experimental setup, a signal in bio-neural networks is represented by artificial neural networks' real value. The output of nodes is calculated by utilizing nonlinear functions of the sum of its inputs. The lines between nodes are connections that are called node edges. Nodes and edges have a weight that is evaluated and adjusted through the learning stage. The weight changes the connection's strength; nodes have a activated threshold while they achieved that threshold. Usually, Nodes are divided into many layers. Each layer contains many different nodes, and it performs various transformations on its inputs. The signal transmits from the first input layer, which is known as the input layer, to the final output layer, which is known as the output layer after visiting the in-between layers, which are called hidden layers multiple times depending on the threshold and the accuracy of achieving the best results for the training model. We have investigated the model of neural networks as a training model with the hybrid approach of decision tree and naïve Bayes to analyze the best performance for bug reports of open-source systems to the suitable developer.

- *Priority of Developers*

Several methods have been tested to calculate the hours needed for each developer in the training dataset and testing dataset and check the priority of each developer in the group using the following steps:

- Calculate the hours needed for each developer in the same class in the training dataset and testing dataset.
- Take the differences between the hours in the training dataset and the testing dataset.
- The accuracy results are sorted as the required average hours for the developer as the highest priority in this group; the

most significant hours needed is the lowest priority in the same group.

Results

We have achieved precision rates of greater than 50%, and they believed that the precision rates they reported are sufficient to help the bug triager decide which developers are good enough to be assigned to a specific bug report [7]. We used 10-fold cross-validation, applied the information gain feature selection algorithm, discretized the data,, and used the Naive Bayes algorithm using MATLAB code and the WEKA software on the eclipse platform bug measures dataset. We have got the same results in MATLAB and WEKA software with 50.001% of correctly classified instances, so it was nearly the same rates that the paper achieved. Using the decision trees, we have got 62.7413% of correctly classified samples, and with using random forest, 66.0746% classification rate has been performed.

We have noticed that some features are not good or made the classification even worse, and others gave a higher rate to the classification accuracy. We have got the results using the 30 features that had high information gain. The 30 features of the dataset made 50.001% classification accuracy in Naive Bayes, and these features made 62.7413% classification accuracy using the decision tree algorithm. Using Random Forest with several trees equal to 135 (the best result we have got while using 135 trees), We have found that the classification accuracy increased by 3.33% than the decision tree achieving 66.0746%.

Table 2 shows the results of different machine learning algorithms we used like Naïve Bayes, Decision Trees, Random Forest, NBTree, Simple Logistic, and Neural Network (Learning rate: 0.3, batchsize:100).

Table 2. Classification Accuracy Results

Algorithm	*Classification accuracy (%)*
Naïve Bayes	*50.001*
Decision Trees	*62.7413*
Random Forest (25 Tress)	*64.9807*
Random Forest (50 Tress)	*65.2767*
Random Forest (75 Tress)	*65.7529*
Random Forest (100 Tress)	*65.9588*
Random Forest (135 Tress)	*66.0746*
Random Forest (200 Tress)	*65.9846*
Random Forest (500 Tress)	*65.9846*
NBTree	*60.1802*
SimpleLogistic	*60.514*
Neural Network	*66.514*

Concerning the trees count in random forest approach, the accuracy results acquired have shown occasionally that whenever several trees increase in a forest only increase its computational time and cost and have no considerable changes in performance gain and this is what occurred in our datasets when we have used more than 135 trees. We have tried 200 trees and 500 trees, and no performance gain is acquired [13]. So, we have tried several methods to improve the classification accuracy and the performance of the algorithms, and we have found that we should make some changes to the class labels like normalization and clustering, as is seen in the next section (clustering results). For each developer's priority in the same group, we calculated the hours needed for each developer using the last hours' feature. Table 3 shows the effects of hybrid utilizing of Naïve Bayes algorithm and DT together in the training dataset (50%-50%) and the usage of RF with Neural Networks (50%-50%) in the training dataset, we could declare as it is seen from results that it has better accuracy results than the usage of each one separately.

Table 3. Hybrid Accuracy Results

Methodology	*Classification accuracy (%)*
Naïve Bayes with decision Trees	*57.3811*
Random Forest with Neural Networks	*67.7473*

Conclusion

The semi-automatic assignment approach is an excellent method to assist the bug triage worker in deciding depending on the specific category of bugs concerning the developer's knowledge and experience. By utilizing an open-source software project (bugs repository) to classify the bug reports dataset. We have been used hybrid machine learning models: random forest, neural networks, and naive Bayes. For the feature selection, we have used the information gain values to detect which features are useful for classification, and we have eliminated 20 features because they had low information gain. We have done the classification based on the best 30 features and one feature for each class label.

The classification results have shown why we have selected algorithms such as random forest and neural networks to classify the bug reports. To support my conclusion, many research papers are approved that suggestion [9] and the results that we have obtained are improved. In general, it will give more accuracy than traditional decision trees and support vector machines. Two advantages that let us select random forest for our dataset. The first advantage is that they do not anticipate linear features or linearly interacted features. The branch of the tree is just a decision point of entire trees combined; this can deal with the dataset correctly. The other advantage is that, because of how they are constructed, this approach deals correctly with more than two-dimensional spaces and large datasets of training samples [14].

For random forest parameters, we have found that the right numbers of random trees to use are 135 trees; the classification accuracy of a random forest relies on the durability of the individual tree model classifiers and dependency between trees [8]. The trees are genuinely classified by the category label of the overcome category (majority voting) [11]. Sometimes, increasing the trees in the random forest technique increases its computational time and cost and has no significant performance gain results [13]. The hybrid approach of utilizing random forest and neural networks, decision trees, and naïve Bayes has created high bug assignment classification accuracy.

Many future enhancements and practical system possibilities can be considered, considering that it is always required that the system should behave automatically for analyzing, recognizing, and appropriate feature selection, for instance, code checking and time while executing different tasks for adaption and resolving incoming new various tasks automatically.

References

1. Wu, W.; Zhang, W.; Yang, Y.; Wang, Q. Time series analysis for bug number prediction. *In Proceedings of the 2nd International Conference on Software Engineering and Data Mining*, Chengdu, China, 23–25 June 2010, 589–596.

2. B.Azhagusundari; Thanamani A.S. Feature Selection based on Information Gain. *IJITEE,* 2013, 2, 19-21.

3. Xuan, J.; Jiang, H.; Ren, Z.; Zou, W. Developer prioritization in bug repositories. *In Proceedings of the 2012 34th International Conference on Software Engineering (ICSE)*, Zurich, Switzerland, 2–9 June 2012, 25–35.

4. Shokripour, R.; Anvik, J.; Kasirun, Z.M.; Zammani, S. A time-based approach to automatic bug report assignment. *J. Syst. Softw.* 2015, *102*, 109–122.

5. Xia, X.; Lo, D.; Ding, Y.; Al-Kofahi, J.; Nguyen, T. Improving automated bug triaging with the specialized topic model. *IEEE Trans. Softw. Eng.* 2016, *43*, 272–297.

6. Ethem Alpaydin, Introduction to Machine Learning, 2nd edition,*MIT press*, 2010, London, England.

7. Anvik J.; Hiewand L.; Murphy G. Who Should Fix this Bug, *ICSE*, 2006, Shanghai, China, 20-28.

8. Breiman L. Random Forests, *Springer Machine Learning*, 2001, 45, 5-32.

9. Liuac M.; Wangb M.; Wanga J.; Lic D. ,Comparison of random forest, support vector machine and back propagation neural network for electronic tongue data classification: Application to the recognition of orange beverage and Chinese vinegar., *Elsevier ,Sensors and Actuators*, 2013, 177, 970–980.

10. Kulkarni V. Y.; Sinha P.K. Random Forest Classifiers :A Survey and Future Research Directions, *International Journal of Advanced Computing*, 2013, 36, 1144-1153.

11.Yan M.; Guo L.; Cukic B. A statistical framework for the prediction of fault-proneness." *Advances in Machine Learning Applications in Software Engineering*. IGI Global, 2007, 237-263.

12. Breiman L. OUT-OF-BAG ESTIMATION, Statistics Department, 1996, University of California, USA.

13. Oshiro T. M.; Perez P. S.; Baranauskas J.A. How Many Trees in a Random Forest, Department of Computer Science and Mathematics, University of Sao Paulo, Lecture Notes in Computer Science, 2012, 7376.

14. Amatriain X., Pompeu Fabra University. Associate Professor in Computer Science, 2019, VP of Engineering at Quora.

15. G. Yang, T. Zhang and B. Lee, "Towards Semi-automatic Bug Triage and Severity Prediction Based on Topic Model and Multi-feature of Bug Reports," *2014 IEEE 38th Annual Computer Software and Applications Conference*, Vasteras, Sweden, 2014, pp. 97-106.