ISSN 2278-3091
Volume 10, No.3, May - June 2021
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse981032021.pdf
https://doi.org/10.30534/ijatcse/2021/991032021

Hybrid Optimization of Multiple Intelligent Recommendation Engines for Software Development Cycles

Ali Jaafar Meera Al-arkawazi[1], Abdullahi Abdu Ibrahim[2]
[1]Department of Computer and Electronics Engineering, Altinbaş University, Istanbul, Turkey, ali.alarkawazi@ogr.altinbas.edu.tr
[2] Department of Computer and Electronics Engineering, Altinbaş University, Istanbul, Turkey, abdullahi.ibrahim@altinbas.edu.tr

ABSTRACT

The primary purpose is to create a hybrid recommendation system approach to improve the performance of such systems. This recommendation system would typically be used to assign or suggest a small number of developers suitable for troubleshooting a bug report. For example, managing collections inside bug repositories is software developers' task to fix any bugs that have been identified. Unfortunately, bugs are often created, so the number of developers needed is high, so it's hard to decide how many to assign to specific tasks.

This better aims better to understand the outcomes of the latest scientific methods. We also addressed developer prioritization and how it can be used to determine the assignment of a problem to a developer. We have studied two aspects: first, selecting bug reports using hybrid machine learning methods, modeling prioritization in the bug repository, and supporting developer assignment tasks with our model. Second, we modeled the relevant objectives suggested by the developers' backgrounds based on proven knowledge and experience. The study focuses on two topers' experience with fixing bugs and developer rankings in the App Store. We've tried to take better assignments using developer prioritization in bug repositories, e.g., bug triage, severity identification, and re-opened bug prediction. We examine the output of the model in a representative sample of bug repositories. The results show that the prioritization of developers' prioritization triage worker and allow the program to solve the bugs more effectively in support of the software support has been clarified. The introduction article, section 2 on the literature and context, section 3 on the work contribution that will be made, section 4 on the methodology analysis and the expected outcomes will be explained, section 5 on the conclusion, and finally, on the potential aspects of this work.

Key words: Artificial neural networks; bug reports; developer prioritization; recommendation systems genetic algorithm; hybrid Intelligent optimization; naive Bayes; open-source repositories; random trees; simple logistic.

1. INTRODUCTION

In software development cycles, open-source bug reports are used in the software development system as many as 60 percent of all bugs. The '10x Rule' – outlier bugs are often much more complicated for locating and repair than other bugs. Still, some bugs can't be detected or patched at all – or at least not with the knowledge we have about recognizing and fixing bugs. Moreover, the number of software bugs has significantly increased. Additionally, more than 180 bugs were identified in Eclipse's bug tracking system in the development period.

Moreover, Debian created nearly 140 outstanding bug reports [25]. Bug reporting tools help developers identify bugs and work to fix them. In addition, for bug reports to be assigned as correctly as possible is a challenge for the software development industry.

Open-source project developers also use an open bug repository for all bugs. The error report must be delegated to team members who are taking responsibility for fixing the bug. When a new report comes in, there's a small group of developers who are expected to fix the bug. Therefore, this will help bug triage staff make decisions about how to prioritize fixing the bug study. Bug repositories are a form of issue tracking system that includes a database of any hardware, software, and programming issues. The open-source bug repositories, as opposed to the closed-source commercial ones, they are open to all for free access. These repositories play an important role in the collaboration between programmers and the project that enables it to work. We used datasets from the same open sourcing project as Eclipse, and we will apply a hybrid classifier recommendation engine with optimization using several machine learning algorithms such as J48, Neural Networks, and genetic algorithms. Some Scholars employed various Support Vector Machines (SVM) and tried using unsupervised learning methods. Besides, they used details from Firefox as an open-source project.

2. BACKGROUND

In this paragraph, we briefly discuss various strategies for fixing bugs suggested by multiple scholars. Numerous approaches have been suggested for discovering the best bug framework that recommends the most bug-fixing apps. Xuan addressed a social network approach to prioritization of bug reports in work on the Eclipse and Mozilla bug repositories. Shokripour et al. used a time-dependent bug detection algorithm. We looked at time metadata for each word in the database. The syntactic variation of words in documents is observed using the statistical technique TF-IDF. Using an improved Linear Discriminant Analysis model, Xia et al. classify bugs automatically. They defined the bug reports that would be acceptable based on the bug problem distribution and the similarities between the bug fix creator and the bug fix distribution. Our proposed algorithm would use developer demographics to suggest a community of potential developers who are involved in bug fixes and also have technical expertise. We studied the two different ways that these studies were performed (i.e., social network metrics and machine learning algorithms). Based on the parameters, we then describe the ways (number of programs, consider variables, and methods) for summarizing the comparisons. A summary of the current literature on problem triaging. Bug repositories are a form of issue tracking system that includes a database of any hardware, software, and programming issues. The open-source bug repositories, as opposed to the closed-source commercial ones, are open to all for free access. These repositories are essential to software development as they enable programmers to share progress during development.

3. DATA ANALYSIS

The dataset was collected from the error tracker on eclipse.org. The kit consists of the 7700 data points of the result of the Eclipse platform bug measures dataset. In the first column of the table, the bug ID is mentioned. Last, the given columns contain the 48 team labels from 1 to 7. Each team of the dataset represents a group of developers according to their companies that assigned before to fix bugs in certain components of the software project. For these features, we tested several machine learning algorithms by using Matlab and Weka to classify features according to team labels and rely on features. It has been achieved based on the 10-fold cross-validation. The dataset is multidimensional. The next graph displays the number of developers involved, along with the number of bugs. Each group of developers is defined by an organization and the number of bugs fixed by each team.

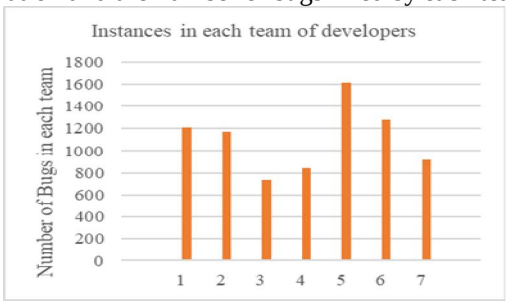

Figure 1: Instances of bugs in each class

The teams of the dataset represent the different companies that are developing the app, as well as their total number.

The Eclipse bug dataset can be found at https://github.com/logpai/bugrepo/tree/master/EclipsePlatform. 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 a sample of a bug report file.

Figure 2: Eclipse Bug Report Sample

4. OPTIMIZATION

The data set is used to train classifiers that can result in wrong classifications and may also take longer to train. The explanatory and clarifying features are redundant and contradictory and do not contribute to the classification. To boost classification accuracy, redundant and inconsistent features must be removed [1]. Use feature selection methods to classify combinations of features that maximize the amount of information that is collected. We omitted the final twenty features with the lowest correlation that could unduly influence our data set performance. Figure 2 displays the effects of the different features' correlations. These algorithms are constructed using natural selection dynamics and natural genetic mechanics. Genetic algorithms are used to solve string structures like biological structures, which develop in time by survival filtering using a randomized yet coordinated exchange of information. Thus, a new set of strings is created in each generation, with only the fittest individuals passing on from generation to generation. The Genetic Algorithm's fundamental characteristics are:

- The genetic algorithm operates with parameter set coding, not parameters themselves.
- The genetic algorithm begins looking from a point population, not a single point.
- The genetic algorithm uses payoff, not derivatives.
- Genetic algorithms use probabilistic, not deterministic, transformation laws.

A genetic algorithm is a type of stochastic algorithm that looks at probabilities. By applying a random search strategy to a step-by-step superstructure model, the search mechanism is determined. The optimum global solution can be reached at a chance of x% certainty. The search process is triggered by the selection of initial stochastic solutions called 'population.' These are called 'chromosomes.' 'Chromosomes' are made up of 'genes.' 'Gene' stands for the optimal variables in the heat exchanger network, such as the mass flow of cold streams and hot streams. For us, we used the GE algorithm to reduce the features used for classification by deleting those that don't have an impact on the result.

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

average merit      average rank  attribute
 0.171 +- 0.001      1   +- 0        2 component3Days
 0.108 +- 0.001      2   +- 0        3 os3Days
 0.078 +- 0          3   +- 0        1 bugID
 0.07  +- 0          4.3 +- 0.46    46 PredictedProbability_1
 0.07  +- 0          4.7 +- 0.46    47 PredictedProbability_2
 0.065 +- 0.003      7.4 +- 1.2     14 nrPeopleCC30Days
 0.065 +- 0.001      7.6 +- 1.28    21 status30Days
 0.065 +- 0.002      7.7 +- 1.27     7 peopleCC
 0.063 +- 0.002      8.9 +- 1.7     20 status3Days
 0.064 +- 0.003      8.9 +- 1.37    13 nrPeopleCC14Days
 0.061 +- 0.003     11.3 +- 1.19    12 nrPeopleCC7Days
 0.06  +- 0.001     11.5 +- 1.28    19 hoursLastFixAfterPreviousRelease
 0.057 +- 0.001     13.7 +- 0.46    22 nrActivities
 0.057 +- 0.003     13.7 +- 1.62    11 nrPeopleCC3Days
 0.055 +- 0.001     14.8 +- 1.08    28 nrActivities30Days
 0.054 +- 0.002     16.5 +- 1.36    10 nrPeopleCC1Days
 0.054 +- 0.001     16.5 +- 0.67    39 filter_$
 0.051 +- 0.001     18.2 +- 0.87    36 hoursToLastFix
 0.051 +- 0.001     18.5 +- 0.81    42 hTLFix2Bins3Days
 0.049 +- 0.002     20.1 +- 0.54    43 hTLFix2Bins7Days
 0.047 +- 0.001     20.8 +- 0.75    49 PredictedProbability_2_1
 0.046 +- 0.001     22.6 +- 0.92    27 nrActivities14Days
 0.046 +- 0.001     23   +- 0.89    18 hoursLastFixBeforeNextRelease
 0.044 +- 0.001     24.8 +- 0.98    26 nrActivities7Days
 0.043 +- 0.001     26.3 +- 1.42    48 PredictedProbability_1_1
 0.043 +- 0.002     26.6 +- 3.01    15 priority30Days
 0.043 +- 0.001     26.8 +- 1.72    24 nrActivities1Days
 0.043 +- 0.002     27.3 +- 2.49     4 platform3Days
 0.043 +- 0.001     28.3 +- 2.05    41 hTLFix2Bins1Days
 0.046 +- 0.001     23   +- 0.89    18 hoursLastFixBeforeNextRelease
 0.044 +- 0.001     24.8 +- 0.98    26 nrActivities7Days
 0.043 +- 0.001     26.3 +- 1.42    48 PredictedProbability_1_1
 0.043 +- 0.002     26.6 +- 3.01    15 priority30Days
 0.043 +- 0.001     26.8 +- 1.72    24 nrActivities1Days
 0.043 +- 0.002     27.3 +- 2.49     4 platform3Days
 0.043 +- 0.001     28.3 +- 2.05    41 hTLFix2Bins1Days
 0.042 +- 0.001     28.9 +- 1.37    25 nrActivities3Days
 0.04  +- 0.001     31.9 +- 1.76    44 hTLFix2Bins14Days
 0.039 +- 0.001     33.3 +- 2.28    38 monthOpened
 0.039 +- 0.001     33.5 +- 2.62    40 hTLFix2Bins0Days
 0.038 +- 0.001     35.2 +- 1.72     8 initPeopleCC
 0.038 +- 0.002     35.3 +- 2.19    29 nrComments
 0.038 +- 0.001     35.8 +- 2.04     9 nrPeopleCC0Days
 0.038 +- 0.001     36.2 +- 2.52    37 hoursToLastResolution
 0.037 +- 0.002     36.9 +- 3.27    16 severity3Days
 0.037 +- 0.002     37.1 +- 2.3     34 nrComments14Days
 0.036 +- 0.001     39.3 +- 0.64    35 nrComments30Days
 0.034 +- 0.002     40.9 +- 0.54    33 nrComments7Days
 0.033 +- 0.001     41.9 +- 0.3     45 hTLFix2Bins30Days
 0     +- 0         44.5 +- 0.92    23 nrActivities0Days
 0     +- 0         44.6 +- 1.91    30 nrComments0Days
 0     +- 0         46   +- 1.84    32 nrComments3Days
 0     +- 0         46   +- 1.9     17 resolution3Days
 0     +- 0         46.1 +- 0.7      5 nrKeywords
 0     +- 0         47.1 +- 2.51    31 nrComments1Days
 0     +- 0         47.7 +- 1.19     6 nrDependentBugs
```

Figure 3: Attribute Correlation ranker

5. METHODOLOGY

Many machine learning algorithms, such as Naive Bayes, Decision Trees, and Support Vector Machines, will be used for implementing the recommendation method, along with some unsupervised machine learning algorithms, such as Expectation Maximization. First, we explored various machine learning algorithms such as J48, Random Trees, and artificial neural networks as recommendation systems. Then, feature selection algorithms with genetic algorithms have helped decide the best features and train the model. We have tried a bi-direction selection algorithm to choose the best features and applied the Naive Bayes algorithm [2]; we have utilized the machine learning algorithms for classification of the data set by using Matlab [3] and Weka [4].

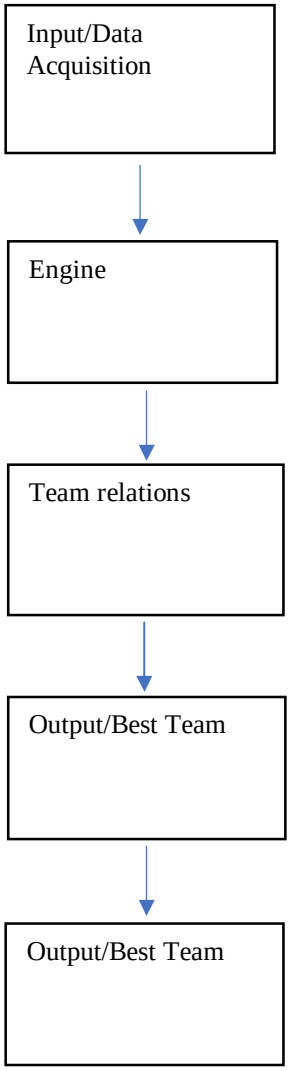

Figure 4: Methodology flowchart

- J48 with optimization

The functions used to classify the dataset in J48 for the 7 classes [2]. The J48 configuration is as follows:

- Random Trees with optimization

The new instance of something is gathered from all the trees that are grown in the forest. Each node generates a classification of new instances, which are reported as votes. Using majority vote, all the trees' votes are combined, and the class that receives the most votes is declared as the new case. As suggested by Breiman [4], in Random Forests, the majority voting mechanism acts as a voting mechanism for classification. Experiments are done related to how people vote.

The parameters that we will utilize of the Random Forest are:
• maxDepth = 0; The maximum depth of the trees, 0 for unlimited.
• numFeatures = 0; The number of attributes to be used in random selection; If < 1 (the default) will use logM+1, where M is the number of inputs. I tried different numbers of features like 10 or 20, but that did not have a real effect on accuracy results.
• numTrees = 150; The number of trees to be generated.

The random forest with thirty-five trees, each built using five features with a coefficient of variation of 0.3347. After creating the classifiers for each method, for each (Xi,Yi) in the original training set "T", pick all "Tk" which does not include "Tk" (Xi,Yi). It is a subset of a dataset in which no original record is present. These are the out-of-bag cases.
There are N such subsets for the data described in "T". OOB classifier is the aggregation of votes only over the popular terms "Tk". It does not include any part of the input (Xi,Yi). The out-of-bag error rate of the classifier on the training set, compared to the known error of the classifier on the training set.
The analysis of error estimation for bagged classifiers shows that the out-of-bag calculation is as reliable as using a test sample of the same size as the training set. Therefore, using the OOB error calculation eliminates the need for a separate test cycle.

- Simple Logistic with optimization

The logistic analysis is the sufficient regression analysis for an outcome that is binary (binary). The logistic regression model is a statistical analysis. Logistic regression is a statistical method that offers insight into the relationship between one dependent variable and one or more independent variables.

- Artificial Neural Networks with optimization

This approach was influenced by how brain cells interact and how brains function. This method was designed to learn how to perform various tasks by considering only samples of training data. In image recognition, the method learns to recognize images that have been identified as "keyboard" or "not keyboard" and is used to identify keyboards in other images. It does not require any prior knowledge or skills as it automatically produces and distinguishes characteristics from the samples entered.

The Neural Networks are built off the biological brain in the way that neurons are the connecting part. Like the neurons in a biological brain, each connected node can transmit a signal to other neurons. By way of similar (artificial) neurons, a signal is transmitted between neurons or nodes connected to them.

An implementation for neural networks uses real numbers to represent the signal and uses functions to represent the output of each node. The links, called edges, are called nodes. Nodes and edges are given weight. Weight adjustment during learning. Weight affects the strength of a connection, but the strength of that connection is dependent on how much weight a node can carry. Normally, a Node has a number of layers. Each layer contains many different nodes which perform operations on various inputs. Signals convoy from the first layer, which is known as the input layer, to the last layer, which is known as the output layer after visiting the in-between layers several times depending on the threshold and the precision of obtaining the best results for the training model. We also examined the model of neural networks as a training model and the hybrid method of decision tree and naïve Bayes in the study of the best results for the assignment of bug reports of open-source systems to the suitable developer.

6. METHODOLOGY

We computed the required time for each programmer in the training and testing sets and determine the demand of each programmer in bug's class by utilizing the following:

• Determine the time required for each programmer in the group.
• Find the differences in time.
• Arrange the programmers: the faster programmer have the highest rank.

7. FINDINGS

We achieved precision rates higher than 50 percent, and we assumed that the precision rates they registered are adequate to enable the bug triager to determine which developers are good enough to be assigned to a particular bug report [3]. We have used 10-fold cross-validation and attribute correlation to apply features and interpret data. In developing our bug measures, we utilized Naive Bayes, J48, Simplelogstic,

random tree, and artificial neural networks. By using an Intrinsic Network, we've obtained the best outcomes.

We also found that there are features not appropriate for the classification and made the results even worse, and others gave a higher rate to the classification accuracy. Therefore, we have optimized for accuracy by taking the best thirty features which have high attribute correlation values. Table 1 displays the model output of various machine learning algorithms like Naïve Bayes, random trees, Simple Logistics, and Artificial Neural Networks with a Learning rate: 0.2, momentum: 0.5, batch size: 100, and 500 iterations.

Regarding the number of trees in the random forest, the results obtained suggest that a more significant number of trees in a forest just raises its computational cost and has no meaningful output advantage, which is what happened in the dataset. When we utilized more than 150 trees. We tried 400 and 600 trees, but no significant changes are observed. We have tried to find a way to boost the classification accuracy and the efficiency of the algorithms, which is why we used optimization. The modification of class labels in the Regarding the number of trees in the random forest, the results obtained suggest that a more significant number of trees in a forest just raises its computational cost and has no meaningful output advantage, which is what happened in the dataset. When we utilized more than 150 trees. We tried 400 and 600 trees, but no significant changes are observed. We have tried to find a way to boost the classification accuracy and the efficiency of the algorithms, which is why we used optimization. The modification of class labels in the clustering algorithm created better outcomes. Meanwhile, the demand of each programmer in the same class is determined by the utilization of the last hours' features, and the results are shown in Table 3. Table 1 shows the effects of optimization and hybrid use of Naïve Bayes algorithm and ANN together in the training set and the utilization of J48 with Simple logistic (50 percent -50 percent) in the training set, and we can declare it has shown better results than the usage of each one alone approximately 7.42 percent improved results in optimization, 8.21 percent for hybrid optimization performance.

Table 1: Accuracy findings

Algorithm	Classification accuracy (%)
J48 Trees	61.24
Random Trees	51.39
SimpleLogistic	61.27
Naïve Bayes	49.08
Artificial Neural Networks	63.12
J48 Trees with Optimization	65.59
Random Trees with Optimization	55.21
SimpleLogistic with Optimization	65.82
Naïve Bayes with Optimization	52.72
Artificial Neural Networks with Optimization	67.81
Hybrid O-J48 with-SimpleLogistic	66.59
Hybrid O-ANN with O-J48	67.59
Hybrid O-Naïve Bayes with O-ANN	60.98
Hybrid O-ANN with Random Trees	62.24

8. CLUSTERING FINDINGS

The system efficiency can be enhanced by modifying the programmer's classes or sets using a K-means algorithm with K=1000. We will have seven clusters (classes) with the closest class for each programmer, the programmer with the closest mean value will belong to that class. In table 2, the examples of programmers and the clusters they belong to are shown.

Table 2: Instances in each new cluster

Cluster	Instances
1	1572
2	1223
3	1180
4	1089
5	949
6	902
7	855

Table 3: Accuracy findings with clustering

Algorithm	Classification accuracy (%)
J48 Trees	67.45
Random Trees	58.81
SimpleLogistic	67.21
Naïve Bayes	56.72
Artificial Neural Networks	71.06
J48 Trees with Optimization	72.27
Random Trees with Optimization	63.17
SimpleLogistic with Optimization	72.19
Naïve Bayes with Optimization	60.93
Artificial Neural Networks with Optimization	75.87
Hybrid O-J48 withO-SimpleLogistic	74.77
Hybrid O-ANN with O-J48	74.97
Hybrid O-Naïve Bayes with O-ANN	68.51
Hybrid O-ANN with Random Trees	62.24

After applying the new classes, we have utilized the same algorithms with 10-fold cross-validation, and we have made better results (approx. 12.1% improvement).

9. CONLUSION

The proposed framework methodology would assist the bug triage programmers in administering and selecting a functioning utility to fix a particular form of bug reports using the open-source bugs repository to assign bug reports. We used machine learning statistical approaches such as Naïve Bayes, J48, random forests, and Simple logistic.

For the feature collection, we have used information benefit values to decide which features make decisions, and we have excluded the 20 least insightful ones. Then, based on the most important 30 features, we will use one function for each of the classes.

Results obtained show that we have chosen to identify bug reports using algorithms such as random forest and neural

networks. To support my conclusion, several research papers seem to indicate [7]. It is more accurate than both conventional decision trees and helps vector machines for most classifications. Two advantages that should inspire us to choose random forest. One major benefit of using this model is that it does not require interactive features. Tree Ensembles combine the strength of Decision Trees, which function well for large datasets. The other key benefit of deep learning is that they are well-suited for solving high-dimensional problems and manage vast quantities of data well.

For the parameters of the random forest, we can discover that the good number of trees for the random forest is 150 trees; the accuracy for a random forest depends on the strength of the individual tree classifiers and a measure of the dependency between them. The trees are categorized by the rating system that earns the most votes. Increasing the number of trees in the random forest could only increase the computational processing time while has no fair change in per- second output earnings.

Utilizing hybrid and optimization approaches, these machine learning algorithms provided an efficient bug assignment framework that meets the requirements.

10. FUTURE ASPECTS

For future studies, using certain selection filtering algorithms can be considered to choose the key words to define bug reports. Chi-square selection is best suited for the computational needs of the experiment. We should develop a new optimization algorithm. We should try other means for prediction instead of this one. The time efficiency factor defines the efficiency of the framework. Our methodology explores the link between selecting the ideal developer and the bug report's severity levels. Current developers should be able to improve the prediction without having to train the parameters on each new update. This approach increases the reliability of how long it takes to upgrade the system. But it should be done electronically for each time to preserve the system's awareness.

Many possible improvements and successful system possibilities can be considered, considering that is always necessary that the system should act automatically for analyzing, recognizing, and fine feature selection, for instance deep learning libraries could be used with some modifications to get some good outcomes for the same application or other applications [27]. Also, code testing and time should be considered while executing different tasks for adaption and resolving incoming new different tasks automatically.

ACKNOWLEDGEMENT

This research was scientifically supported by Professor Abdullahi Abdu Ibrahim. We thank our colleagues from Altinbas University who provided insight and expertise that greatly assisted the research.

REFERENCES

1. B.Azhagusundari, Antony Selvadoss Thanamani,**Feature Selection based on Information Gain**, *IJITEE*, vol.2, PP. 2278-3075, 2013.
2. Ethem Alpaydin, **Introduction to Machine Learning**. Published by *MIT press*, London, England, 2010.
3. John Anvik, Lyndon Hiewand, and Gail Murphy, **Who Should Fix this Bug**, *ICSE*, Shanghai, China, May 20-28, 2006.
4. Leo Breiman, "**Random Forests**", *Springer Machine Learning*, vol. 45, Issue 1, PP. 5-32, 2001.
5. **MATLAB Documentation**, available at http://www.mathworks.com/help/matlab/
6. Miao Liua,c, Mingjun Wangb, Jun Wanga, and Duo Lic , **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**, *Sensors and Actuators*, Vol. 177, PP. 970–980, 2013.
7. Leo Breiman, **Out-Of-Bag Estimation**, Published by *Statistics Department* University of California,1996.
8. Thais Mayumi Oshiro, Pedro Santoro Perez, and Jos´e Augusto Baranauskas, **How Many Trees in a Random Forest**, s*Department of Computer Science and Mathematics*, University of Sao Paulo, Lecture Notes in Computer Science 7376, 2012.
9. Vrushali Y Kulkarni and Pradeep K Sinha, **Random Forest Classifiers: A Survey and Future Research Directions**, *International Journal of Advanced Computing*, Vol. 36, Issue.1, 2013.
10. **WEKA Manual**, Version 3.6.2, University of Waikato, New Zealand, 2010.
11. **Xavier Amatriain Articles**, Pompeu Fabra University. Associate Professor in Computer Science, VP of Engineering at Quora.
12. Yan Ma, Lan Guo and Bojan Cukic; **A Statistical Framework for the Prediction of Fault-Proneness**, *West Virginia University*, Morgantown, WV 26506.
13. Yongheng Zhao and Yanxia Zhang, **Comparison of decision tree methods for finding active objects**, *National Astronomical Observatories*, 2007.
14. Hikai Guo, Shifei Chen, Siwen Wang, Decheng Zhang, Yaqing Liu, Chen Guo, Hui Li, and Tingting Li, **A Multi-Factor Approach for Selection of Developers to Fix Bugs in a Program**, *Applied sciences*, vol.9, pp.3327, 2019.

15. Jifeng Xuan, He Jiang, Zhilei Ren, and Weiqin Zou, **Developer Prioritization in Bug Repositories**, *IEEE*, 2012.
16. Ibrahim Aljarah, Shadi Banitaan, Sameer Abufardeh, Wei Jin and Saeed Salem, **Selecting Discriminating Terms for Bug Assignment: A Formal Analysis**, *PROMISE '11*, September 20–21, Banff, Canada, 2011.
17. Anjali Goyal and Neetu Sardana, **Empirical Analysis of Ensemble Machine Learning Techniques for Bug Triaging**, IC3, *IEEE*, 2019.
18. Guo, S., Chen, R., Li, H., Zhang, T., Liu, Y., **Identify Severity Bug Report with Distribution Imbalance by CR-SMOTE and ELM**, *Int. J. Softw. Eng. Knowl. Eng*., vol.29, PP.139–175, 2019.
19. **Valuating an Assistant for Creating Bug Report Assignment Recommenders**. Available at: http: //ceur-ws.org/Vol-1705/04-paper.pdf, accessed on 20 June 2019).
20. Eng, W., Xu, J., Zhao, H., **An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem**, *IEEE Access*, vol. 7, PP. 20281–20292, 2019.
21. Liu, Y., Wang, X., Zhai, Z., Chen, R., Zhang, B. Jiang, Y. **Timely daily activity recognition from headmost sensor events**, ISA Trans. 2019.
22. Xuan, J., Jiang, H., Ren, Z., Zou, W. **Developer prioritization in bug repositories**. proceedings of the 34th *International Conference on Software Engineering* (ICSE), Zurich, Switzerland, pp. 25–35, 2012.
23. Shokripour, R., Anvik, J., Kasirun, Z.M., Zammani, S. A., **Time-based approach to automatic bug report assignment**, *J. Syst. Softw.*, vol.102, PP. 109–122, 2015.
24. Xia, X., Lo, D., Ding, Y., Al-Kofahi, J., Nguyen, T., **Improving automated bug triaging with specialized topic model**, *IEEE Trans. Software Eng.*, Vol. 43, PP. 272–297, 2016.
25. Wu, W., Zhang, W., Yang, Y., Wang, Q., **Time series analysis for bug number prediction**, *Proceedings of the 2nd International Conference on Software Engineering and Data Mining*, Chengdu, China, pp. 589–596, 2010.
26. Christopher A. Choquette-Choo, David Sheldon, Jonny Proppe, John Alphonso Gibbs, and Harsha Gupta, **A multi-label, dual-output deep neural network for automated bug triaging**, *Intel PSG*, San Jose, California, United States, *IEEE*, 2019.
27. A. Y. Saleh, L. Ilango, **Detection of COVID-19 in Computed Tomography (CT) Scan Images using Deep Learning,** *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no.5, 2020.