JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

# PREDICTING SOFTWARE DEFECT COMPLEXITY AND ACCURACY USING BUG ESTIMATION AND CLUSTERING

**[1]Sinduja.B, [2]P.M. Kamatchi**
[1]PG Scholar, [2]Assistant Professor
[1]Department Of Computer Science and Engineering, Krishnasamy College of Engineering and Technology, Nellikuppam Main Road, S.Kumarapuram, Cuddalore, Tamil Nadu 607109
[2]Department Of Computer Science and Engineering, Krishnasamy College of Engineering and Technology, Nellikuppam Main Road, S.Kumarapuram, Cuddalore, Tamil Nadu 607109

***Abstract:*** Data mining can be used to explore interesting patterns in software bug repositories. Managing large quantity of new bugs submitted by testers/reporters on daily basis in large organizations is a complex task. This increases the defect triage process, where the bugs are analyzed and prioritized within specified period to assign appropriate developers to fix the defects based on the severity level of the bugs. In this paper, a prediction model is proposed to predict complexity and severity level of bugs. Complexity and severity level of a bug helps the development and testing team in an organization to plan future software releases. For all the bugs stored in bug repository, the time required to fix the bug i.e., fix duration/time is calculated, and complexity clusters (simple, medium, and complex) are created for the computed fix duration. The bugs are then mapped to the complexity cluster based on the estimated fix duration. In this study, we have extracted bug report dataset from Bugzilla bug tracking system. We compare four algorithms, namely, Naïve Bayes, Decision Tree, Random Forest, and Convolutional Neural Network for Classification and parameters measured in terms of accuracy and F-score is evaluated.

***IndexTerms* – Bug Complexity, Clustering, Prediction, Accuracy.**

## I. INTRODUCTION

The proposed model performs three major tasks in the Clustering phase. The first task is to calculate the time required to fix the bugs (fix duration) stored in bug repository. The second task is to create clusters called complexity cluster based on the bug’s fix duration. And the third task is to map the bugs to complexity cluster, which defines bug’s complexity. For example, bugs with less fix duration falls under simple cluster whereas bugs with more fix duration falls under complex cluster.

To build a prediction model, the bug report is transformed into a feature of vectors/attributes (bug ID, description etc.) using machine learning algorithms. Therefore, the bug dataset is pre-processed using normalized pre-processing techniques to convert into vector of features. Using appropriate feature selection methods, the most discriminative features are selected from the available feature list. In this study, machine learning algorithms are used to classify and predict the severity levels of bug reports. Once the model is trained, the performance of the predicted model must be evaluated using the available dataset and accuracy levels for different machine learning algorithms are obtained.

Figure 1 illustrates the overall architecture of the proposed system. Data is extracted from any of the bug tracking tools and stored in bug repository. Fix duration is calculated for the extracted bug dataset and clusters are created. The bugs are mapped to the complexity cluster. The bug data is pre-processed using normalized pre-processing technique for removing any outliers and noise in the data. Finally, the bug report is represented as a feature matrix consisting of n selected features. The pre-processed data is then trained and using four machine learning algorithms the system is evaluated for better accuracy. Prediction is done based on the accuracy of each algorithm. The accuracy curve is plotted, and desirable results are achieved.

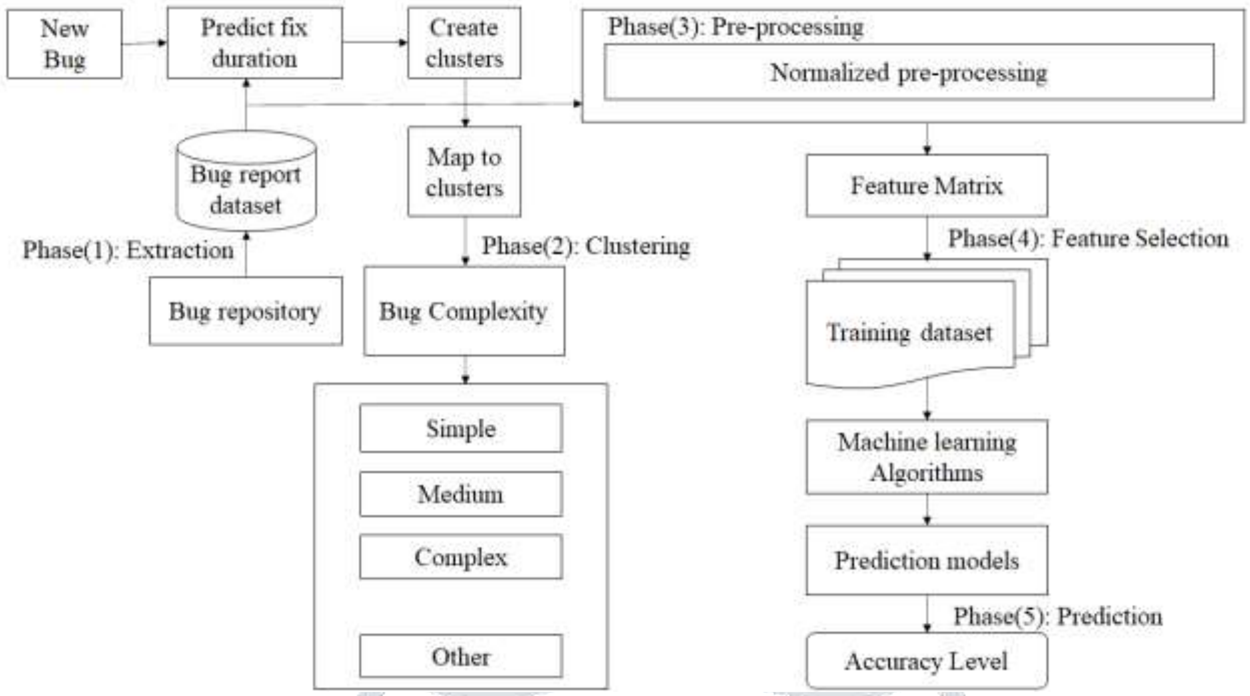


Fig 1: System Architecture

## II. METHODOLOGY

### DATA EXTRACTION

Software development and testing related data in large organizations are managed and tracked with the usage of some internal/external tool. Software bug tracking tools are used for managing and tracking the status of a bug within a project. These bug tracking tools manage the bug repositories internally where all the bugs raised by the testing team and its related data are stored. In our study, we have extracted the dataset from Bugzilla bug tracking tool. By specifying the bug id, all relevant bug information can be fetched. The extracted datasets include features like bug ID, bug description, severity level of the bug, status, created by, created date and time etc.

### CLUSTERING

Clustering is a method that groups similar objects with similar characteristics. Objects in same cluster are similar to one another and dissimilar to objects in another cluster. Clustering is an unsupervised machine learning algorithm used for classification. There are several clustering algorithms and techniques available for measuring the distance between cluster data points. In our study, we have used DBscan (Density-based Spatial Clustering of Applications with Noise) and SOM (Self-Organizing Map) clustering algorithms respectively.

### DATA PRE-PROCESSING

In the pre-processing step, the bug dataset needs to be pre-processed to convert into a vector of relevant features where the features correspond to attributes/words in the dataset. The aim of this pre-processing step is to remove any irrelevant, noisy, error and duplicate data in the bug dataset to ensure high quality clusters and accuracy.

### FEATURE SELECTION

Feature selection is the most important step to select the most relevant features by reducing the input variable from a list of features to get rid of noise in data using appropriate feature selection methods. Feature selection increases the prediction accuracy of the model by eliminating irrelevant and redundant features. Finally, feature matrix is generated for the bug dataset where each row represents n selected features.

### PREDICTION

In our study, machine learning algorithms are employed to predict the bug's severity level. These algorithms deal with unstructured data in the bug reports. In this study, we compare four algorithms, namely, Naïve Bayes, Decision Tree, Random Forest, and Convolutional Neural Network (CNN) for Classification and parameters measured in terms of accuracy and F-score is evaluated. The accuracy measures how correctly the trained model predicts bug's severity level. On the other hand, the F-score is the harmonic mean of precision and recall and is used to measure the accuracy of the predicted model.

$$\text{F-measure} = 2 * \frac{precision * recall}{(precision + recall)}$$

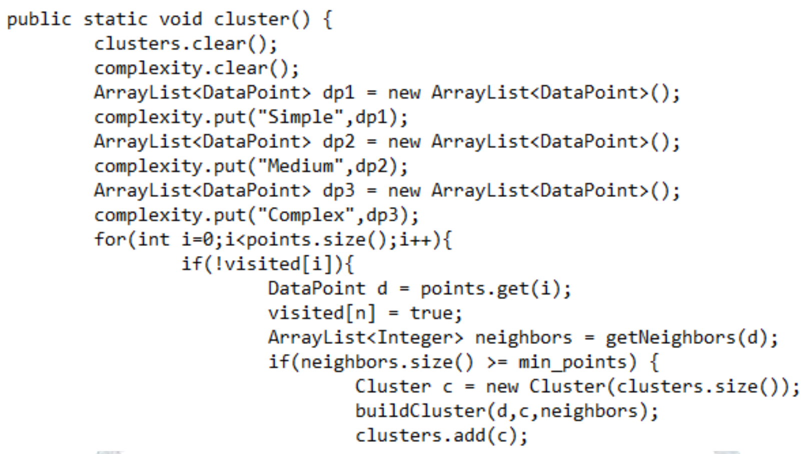
## III. IMPLEMENTATION

### CALCULATING FIX DURATION OF BUG

Bug's fix duration can be calculated by considering the parameters/features: bug state, creation date and modified date. For "Fixed" or "Resolved" bug states, the fix duration is the difference between bug modified date and created date. This difference (fix duration) can be generated in any time format e.g., hours, days, weeks, months etc. In this paper, we represent the bug's fix duration in no. of days.

## CLUSTERING TO PREDICT BUG'S COMPLEXITY

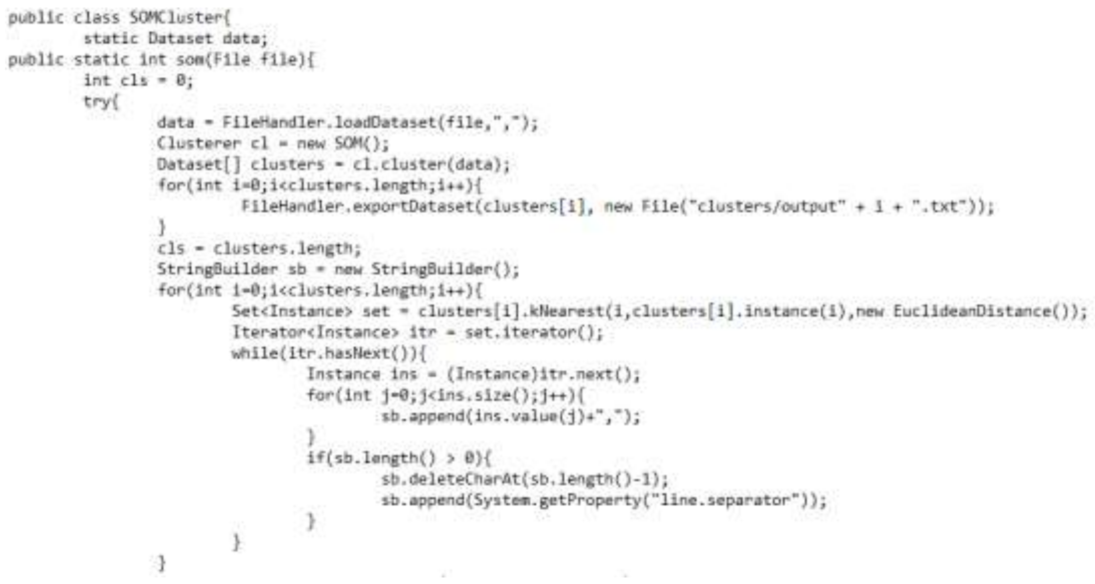
Based on the fix duration calculated the bugs are clustered as Simple, Medium, and Complex which defines the complexity level of the bugs. Here DBscan (Density-Based Spatial Clustering of Applications with Noise) is used for creating clusters. DBSCAN groups densely grouped data points into a single cluster and detects noise in the dataset perfectly. The java code for predicting bug's complexity is given below.



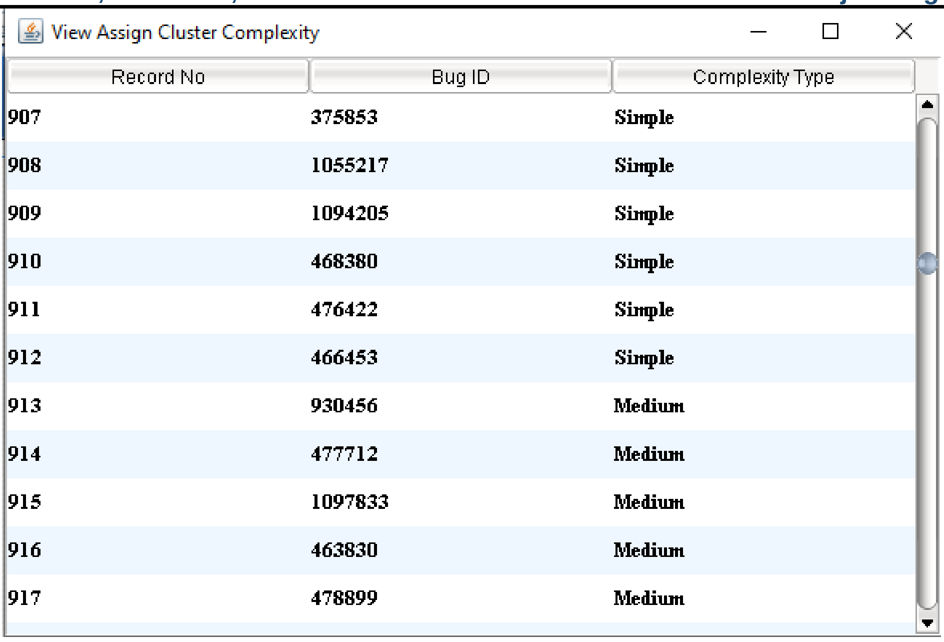
```
public static void cluster() {
        clusters.clear();
        complexity.clear();
        ArrayList<DataPoint> dp1 = new ArrayList<DataPoint>();
        complexity.put("Simple",dp1);
        ArrayList<DataPoint> dp2 = new ArrayList<DataPoint>();
        complexity.put("Medium",dp2);
        ArrayList<DataPoint> dp3 = new ArrayList<DataPoint>();
        complexity.put("Complex",dp3);
        for(int i=0;i<points.size();i++){
                if(!visited[i]){
                        DataPoint d = points.get(i);
                        visited[n] = true;
                        ArrayList<Integer> neighbors = getNeighbors(d);
                        if(neighbors.size() >= min_points) {
                                Cluster c = new Cluster(clusters.size());
                                buildCluster(d,c,neighbors);
                                clusters.add(c);
```

## MAPPING FIX DURATION OF BUG TO COMPLEXITY CLUSTER

In this module, three different clusters are created based on the complexity level (simple, medium, complex) of the bugs. The clusters are named as Cluster 0, Cluster 1 and Cluster 2 respectively. Simple bugs are grouped under Cluster 0, Medium bugs under Cluster 1 and Complex bugs under Cluster 2. SOM (Self-Organizing Map) is used for this purpose. SOM is an unsupervised clustering algorithm that groups similar data together and displays similarities among data. The java code and snapshot for mapping the fix duration of bug to complexity cluster is given below.



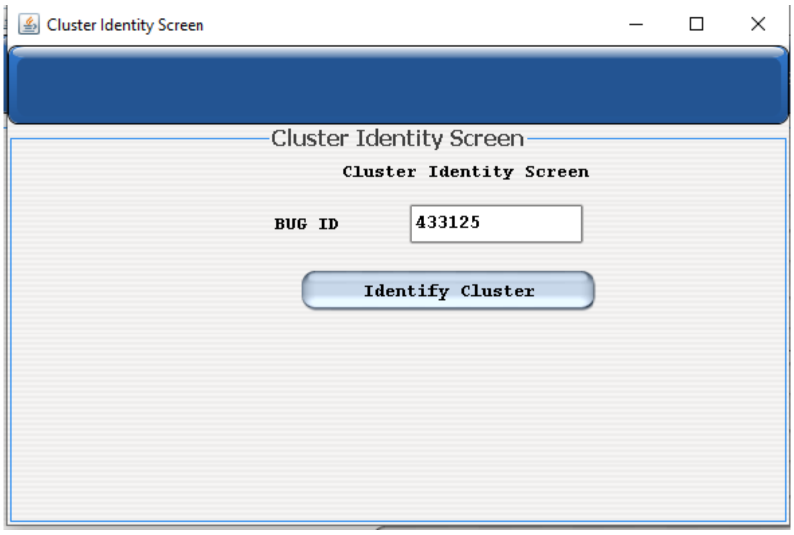
```
public class SOMCluster{
        static Dataset data;
public static int som(File file){
        int cls = 0;
        try{
                data = FileHandler.loadDataset(file,",");
                Clusterer cl = new SOM();
                Dataset[] clusters = cl.cluster(data);
                for(int i=0;i<clusters.length;i++){
                        FileHandler.exportDataset(clusters[i], new File("clusters/output" + i + ".txt"));
                }
                cls = clusters.length;
                StringBuilder sb = new StringBuilder();
                for(int i=0;i<clusters.length;i++){
                        Set<Instance> set = clusters[i].kNearest(i,clusters[i].instance(i),new EuclideanDistance());
                        Iterator<Instance> itr = set.iterator();
                        while(itr.hasNext()){
                                Instance ins = (Instance)itr.next();
                                for(int j=0;j<ins.size();j++){
                                        sb.append(ins.value(j)+",");
                                }
                                if(sb.length() > 0){
                                        sb.deleteCharAt(sb.length()-1);
                                        sb.append(System.getProperty("line.separator"));
                                }
                        }
                }
```


View Assign Cluster Complexity

| Record No | Bug ID | Complexity Type |
|---|---|---|
| 907 | 375853 | Simple |
| 908 | 1055217 | Simple |
| 909 | 1094205 | Simple |
| 910 | 468380 | Simple |
| 911 | 476422 | Simple |
| 912 | 466453 | Simple |
| 913 | 930456 | Medium |
| 914 | 477712 | Medium |
| 915 | 1097833 | Medium |
| 916 | 463830 | Medium |
| 917 | 478899 | Medium |

## CLUSTER IDENTITY

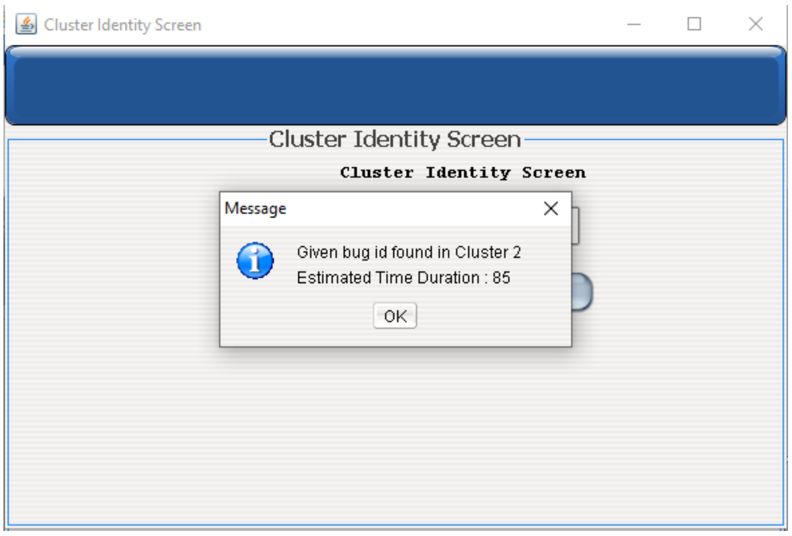
Given a Bug Id, this cluster identity identifies which bug belong to which cluster and its estimated time. Complexity Cluster Chart allows to visualize bug clusters. It graphically represents the total number of bugs in Simple, Medium and Complex categories. Execution time chart allows to find the execution time of DBscan and SOM algorithms respectively. The snapshot for cluster identity is given below.


Cluster Identity Screen

Cluster Identity Screen

Cluster Identity Screen

BUG ID 433125


Cluster Identity Screen

Cluster Identity Screen

Cluster Identity Screen

Message

Given bug id found in Cluster 2
Estimated Time Duration : 85

## MACHINE LEARNING ALGORITHMS

In the below section, we briefly present the machine learning algorithms used for prediction in this work.

### 1) NAIVE BAYES
Naive Bayes is a probabilistic machine learning algorithm based on Bayes theorem used for classification with the assumption that every pair of features are independent. It predicts the probability of a class from a given set of features.

### 2) DECISION TREE
Decision Tree representing a tree structure is a supervised machine learning algorithm mainly used for classification. In decision tree model, each internal node in the tree represents a single attribute/feature where the data is split while leaf nodes represent final outcome or decision.

### 3) RANDOM FOREST
Random Forests is a method that operates on several decision trees at training time. The Random Forest classifier merges multiple decision trees and produces outcome that has accurate and stable prediction results.

### 4) CONVOLUTIONAL NEURAL NETWORK
A convolutional neural network (CNN) process data through multiple layers. A CNN consists of an input layer, one or many hidden layers and an output layer. The operations that are repeated over the layers of CNN are Convolution, Activation and Pooling. The purpose of using CNN in our study is to minimize manual feature extraction and to produce highly accurate results.

## IV. RESULTS AND DISCUSSIONS
In this paper, four machine learning algorithms were applied on the bug dataset for predicting the severity level of bugs. The algorithms are Naïve Bayes, Decision trees, Random Forest, and Convolutional neural network. Feature selection is applied after normalized pre-processing and n features are selected. Based on the accuracy, the accuracy of different classifiers ranges between 71.25% to 92 The below table 1 and figure 2 shows the performance and accuracy results of the machine learning algorithms relevant to the Bugzilla bug dataset with pre-processed data and selected features.

Table 1: Performance results of different machine learning algorithms on Bugzilla dataset

| S.NO | ALGORITHMS | ACCURACY |
|---|---|---|
| 1 | Naïve Bayes | 71.25 |
| 2 | Decision Trees | 84.50 |
| 3 | Random Forest | 88.75 |
| 4 | Convolutional Neural Network | 92 |

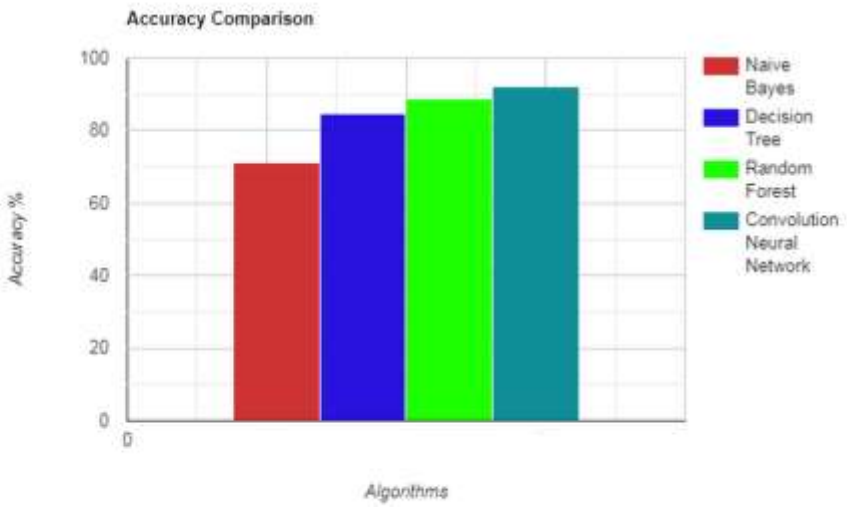


Fig 2: Accuracy of different machine learning algorithms

## V. CONCLUSION
In this paper, the complexity of the bug is predicted using data mining approach. Based on the estimated fix duration, complexity clusters are created by using clustering techniques DBscan and SOM respectively. We also applied machine learning algorithms to predict the severity level of software bugs. The performance is evaluated by comparing four machine learning algorithms, namely Naïve Bayes, Decision Trees, Random Forest and Convolutional neural network and accuracy graph is plotted. Finally, the proposed model predicts the severity of bugs submitted to Bugzilla bug tracking tool with high accuracy.

## VI. REFERENCES

[1] T. Zhang, H. Jiang, X. Luo, and A. T. Chan, "A literature review of research in bug resolution: Tasks, challenges and future directions," The Computer Journal, vol. 59, pp. 741-773, 2016.
[2] C. Liu, J. Yang, L. Tan, and M. Hafiz, "R2Fix: Automatically generating bug fixes from bug reports," in 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation, 2013, pp. 282-291.

[3] J. Uddin, R. Ghazali, M. M. Deris, R. Naseem, and H. Shah, "A survey on bug prioritization," Artificial Intelligence Review, vol. 47, pp. 145-180, 2017.
[4] A. Lamkanfi, S. Demeyer, Q. D. Soetens, and T. Verdonck, "Comparing mining algorithms for predicting the severity of a reported bug," in 2011 15th European Conference on Software Maintenance and Reengineering, 2011, pp. 249-258.
[5] A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals, "Predicting the severity of a reported bug," in 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010), 2010, pp. 1-10.
[6] 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, 2012, pp. 215-224.
[7] T. Zhang, J. Chen, G. Yang, B. Lee, and X. Luo, "Towards more accurate severity prediction and fixer recommendation of software bugs," Journal of Systems and Software, vol. 117, pp. 166-184, 2016.
[8] R. Jindal, R. Malhotra, and A. Jain, "Prediction of defect severity by mining software project reports," International Journal of System Assurance Engineering and Management, vol. 8, pp. 334-351, 2017.
[9] A. F. Otoom, D. Al-Shdaifat, M. Hammad, and E. E. Abdallah, "Severity prediction of software bugs," in 2016 7th International Conference on Information and Communication Systems (ICICS), 2016, pp. 92-95.
[10] K. Jin, A. Dashbalbar, G. Yang, B. Lee, and J.-W. Lee, "Improving predictions about bug severity by utilizing bugs classified as normal," Contemp Eng Sci, vol. 9, pp. 933-942, 2016.
[11] C.-Z. Yang, K.-Y. Chen, W.-C. Kao, and C.-C. Yang, "Improving severity prediction on software bug reports using quality indicators," in 2014 IEEE 5th International Conference on Software Engineering and Service Science, 2014, pp. 216-219.
[12] V. Singh, S. Misra, and M. Sharma, "Bug severity assessment in cross project context and identifying training candidates," Journal of Information & Knowledge Management, vol. 16, p. 1750005, 2017.
[13] W. Liu, S. Wang, X. Chen, and H. Jiang, "Predicting the severity of bug reports based on feature selection," International Journal of Software Engineering and Knowledge Engineering, vol. 28, pp. 537-558, 2018.
[14] N. K. S. Roy and B. Rossi, "Towards an improvement of bug severity classification," in 2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications, 2014, pp. 269-276.
[15] G. Yang, K. Min, J.-W. Lee, and B. Lee, "Applying Topic Modeling and Similarity for Predicting Bug Severity in Cross Projects," KSII Transactions on Internet & Information Systems, vol. 13, 2019.