An Assessment of Eclipse Bugs' Priority and Severity Prediction Using Machine Learning

Mohammed Q. Shatnawi[1], Batool Alazzam[2]

[1, 2]Computer Information Systems Department, Faculty of Computer and Information Technology
Jordan University of Science and Technology, Jordan

Abstract: The reliability and quality of software programs remains to be an important and challenging aspect of software design. Software developers and system operators spend huge time on assessing and overcoming expected and unexpected errors that might affect the users' experience negatively. One of the major concerns in developing software problems is the bug reports, which contains the severity and priority of these defects. For a long time, this task was performed manually with huge effort and time consumptions by system operators. Therefore, in this paper, we present a novel automatic assessment tool using Machine Learning algorithms, for assessing bugs' reports based on several features such as hardware, product, assignee, OS, component, target milestone, votes, and versions. The aim is to build a tool that automatically classifies software bugs according to the severity and priority of the bugs and makes predictions based on the most representative features and bug report text. To perform this task, we used the Multi-Nominal Naive Bayes, Random Forests Classifier, Bagging, Ada Boosting, SVC, KNN, and Linear SVM Classifiers and Natural Language Processing techniques to analyze the Eclipse dataset. The approach shows promising results for software bugs' detection and prediction.

Keywords: Eclipse, Severity Prediction, Machine Learning, ML, Software Bug, Bug Priority, Bug Detection, Software Security.

1. Introduction:

A software bug is a failure in the program which causes unexpected or unwanted outputs [1]. It is an error that prevents the program to operate its function as it should, either while launching the software or while using its features. System operators and software developers spend huge time testing their proposed software as modules to bypass having any type of bugs or errors and assessing the potentials of having any type of system crashes for any reason. Research has shown that half of the developers' time is spent on fixing bugs, and just 36% is specified for adding new features. The bugs fixing process consists of determining why the program or software is behaving abnormally and trying to fix the part of the component that caused the error [2]. Although, this is their main concern to ensure a smooth user experience, there still exist many cases where human limitations in designing software modules cause certain lacuna in building those modules. The most common types of bugs or defects encountered in the software testing process are Functional Bugs, Logical Bugs, Workflow Bugs, Unit Level Bugs, System-level Integration Bugs, and Out of Bound Bugs [3]. Software bugs should be detected through the early stages of the development life cycle at the testing phase; because the cost to fix the error differs depending on which level it is discovered or fixed. For example, the cost to fix an error after the product is released is four to five times as much as being discovered during the design phase [4].

Classifying software bugs makes the corrective actions process easier and minimizes the defects. This step determines which bug should be fixed at first, which should be considered with higher priority and may affect the whole program operationally, functionally or security wise. For that, there is a need to automate this process to reduce the cost of time and effort. Bugs are prioritized according to two features: bug priority and bug severity, in which the bugs with the highest priority and severity are critical and must be solved at first. Now, the core question is how to define the bug priority? According to the priority, it divides the bugs into four different levels: critical bugs, high-priority bugs, medium-priority bugs, and low-priority bugs. The priority is set by the committee, contributor, or component owner; in this way, the importance of the bugs, and the possible enhancement can be indicated. The bugs' priority levels are symbolized to four levels from P1 to P4. P1 which has the highest priority, and it must be fixed, P4 which has the lowest priority which means is a valid bug, has a choice to fix it but it is not important. As for the bugs' severity, is assigned by the bug reporter which defines how much the bug is important. Bug severity is divided into seven types, which are major, blocker, critical, normal, enhancement, minor, and trivial [5].

- Major: is a major loss of function.
- Blocker: blocks the work of testing and development.
- Critical: is crashes, loss the data, memory leak.
- Normal: is a regular issue, loss of some functions but under specific conditions.
- Minor: is a trivial loss of function or another problem that is easy to be solved.
- Enhancement: This is an enhancement request.
- Trivial: is a cosmetic problem, improving the problem.

Different open-source projects can be an interesting area to analyze, study, and track bugs report such as Eclipse [6]. Eclipse project that composed of five sub-projects which are: Platform, Java development tools, Plug-in Development Environment (PDE), e4, Equinox, and Orion [7].

The provided model in figure 1 depends on the bugs features that are trained model on. Feature selection is an essential step; because not all features in the bugs dataset have the same importance and affect the final results in a big manner. The model will be trained on the selected features to be able to classify the bugs according to the priority and severity level. As an extra advantage, bugs report text will be fed to the model and extracted more characteristics that will be helpful at the prediction phase.

In this research, the Eclipse software bugs dataset will be used to develop a model that can automatically classify and predict the severity and priority of the bugs. The provided model will use bugs features which will be selected carefully to ensure they are representative and achieve the highest

results. At the classification process, machine learning algorithms, Random Forests Classifier, Bagging, Ada Boosting, SVC, and KNN, will be used to make bugs severity and priority classification based on hardware, product, component, OS, resolution, version, and status features.

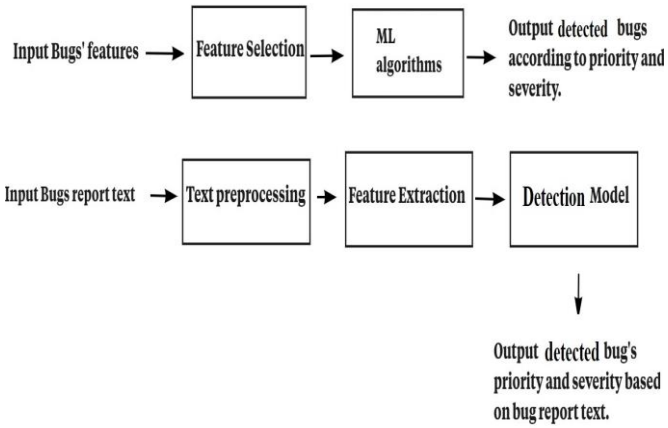

Figure 1: Bugs classification and prediction model architecture.

The results will show an F1 score, accuracy, precision, recall, and execution time for each used algorithm at the testing process. On the other hand, a summary of bug reports will be studied using NLP techniques and try to extract the characteristics of the text and used them in the bugs prediction process. The report text will be preprocessed, converted to numerical form to facilitate dealing with, using TF-IDF.

2. Related Work

During the development life cycle, testers and editors are responsible to detect, triaging, and fixing the bugs that exist in the program or software. Bug triaging is an essential step that must be done at the beginning stages, that will consume the cost whether in the time, money, and effort [8]. Different cases cause software bugs, such as what happens during the development process. One of the drawbacks of using machine learning is that these algorithms cannot extract the important feature patterns for learning the classifier. This work aims to use a novel deep learning model for bug severity classification using CNN and random forest with boosting (BCR). This model can learn from the hidden and high representative features in the dataset. NLP techniques deal with the bug report text, and n-gram extracts the features. For CNN, is used to extract the important feature patterns and BCR to classify the multiple bug severity classes [9].

Machine learning algorithms are used in the classification process to compare the severity accuracy, such as the decision tree and random forest. Many of the bugs' reports have incomplete data that relate to the features, which causes inaccurate results. The case deletion, mean imputation, median imputation, and k nearest neighbor are methods that are used to deal with this issue. Different machine learning algorithms are used to predict future software faults depend on historical data and the comparison between each algorithm performance depends on some measures, such as accuracy, precision, recall, F-measure, and ROC curves. The used dataset consists of the number of faults, and the number of test workers for each day. The main goal of this study is to measure the ML performance in the bugs classification process [10].

Bugs detection is a very essential step that should be done at the early level of the software development lifecycle. Which reflects in the software quality, reliability, and the cost of development positively. The machine learning algorithms, such as Logistic regression, Naive Bayes, and Decision tree, are used for bugs' severity classification process. In addition, statistical analysis helps in the detection of the bug. The imbalanced data problem can be solved using the oversampling methods, such as SMOTE [11]. Bugs' reports have a summary or description field as text, at the bug prediction process there is a need to deal with this text and convert it to numeric format to apply the machine learning algorithms. Bag-of-Words and tf-idf transformer are used to convert the text data into feature vectors. At the training and testing processes different machine learning algorithms and make a comparison between the final accuracy results [12]. The inconsistency in the used datasets to design and develop prediction models for software detects leads to get inaccurate results. Self-Paced Association and Node embedding (SPAN) is used to make connections between the text datasets effectively. Also, Software Bug Report Network is used to identify aimed features [13].

Bug's priority is defined based on the emotion words that are included in the bug reports. Emotion values will be assigned for these emotion words. After that, machine learning models are applied to this data to predict the bugs' priority. THE SentiWordNet emotion words corpus is the most corpus used. Dealing with the text part of the dataset is includes tokenization, part of speech, remove the stop words, and lemmatization [14]. The bug severity, component, and problem title are the based features that will be used to make a model that aims to prioritize the bug. As the first step, the text input features are converted to numeric features using TF-IDF. PCA and NMF are used to reduce the complexity and running time of the algorithms. For the clustering approach, X-Mean and K-Mean algorithms are used, SVM, Naive Bayes classifiers are applied to all features. The FindBugs, JLint, and PMD are three bug-finding tools used in this research applied on open projects, namely Columba Lucene, and Scarab to assign the priority of the bug. The developed model is an improvement of the existing models, which use two new bugs features, classification algorithms, and feature reduction techniques [15].

There were many tries to find alternatives to the manual code testing and detect the defects, which is time and effort consuming. They got the main advantage from the neural networks, which have multiple layers that allow extracting the high-level features from the original dataset to solve the problems. Deep Belief Networks (DBN), CNN, LSTM, and Transformer architecture are the most popular deep learning techniques used for software bugs detection [16]. Deep neural networks can be appropriate to deal with bug prediction. The TensorFlow and traditional python algorithms are used from the scikit-learn python package. The best setup of the networks is decided according to the best F1-measure result, such as learning rate, number of layers, number of learning epochs, number of neurons, the early stopping, and dynamic learning rates. The final results

are also compared to the final results using machine learning algorithms [17]. The convolution neural networks can be used for software bug triaging to deal with the text features in the dataset. Word2vec is used to make a word vector representation. Then, CNN is combined with batch normalization, pooling, and a full connection approach to learning from the word vector. This work is based on three large open projects, namely Eclipse, Mozilla, and NetBeans. The text preprocessing approach includes word segmentation, Stemming, and Stop words removal. CNN model extracts the high-level features from the text, the different spatial scales are used with convolution kernels. The ReLU is used as an activation function to avoid the gradient descent problem. For the dimensionality problem, the maximum pooling operation is used. This model shows high performance comparing with many other machine learning algorithms [18]. Machine learning is one of the approaches that is used in the bug prediction process. ML is widely used because it gives accurate results and good analysis. Systematic Literature Review (SLR) is a methodology that contains identifying, evaluating, and understanding steps for the available research. During this work, the goal is to know What kind of machine learning techniques have been selected for the prediction model? Which performance measures should be used? and Which metrics are frequently used? The six techniques that have been identified are Bayesian Network (BN), Neural Network (NN), Support Vector Machine (SVM), Clustering, Feature Selection (FS), and Ensemble Learning (EL). Through these used techniques, NN is the most widely used.

For the measurement, there are types, such as graphical measure and numerical measure. The graphical measure consists of the precision-recall curve, cost curve, and ROC curve, whereas numerical measure consists of accuracy, F-Measure, precision, recall [19]. The duplicated software bugs' reports it's a problem that faces many of the software companies, which these companies have a large number of users and customers that can submit a huge number of bugs reports for the same fault. Many techniques are used to deal with this problem, such as the classification technique.

The feature extraction technique that reduces the feature size and yet retains the information that is most critical for the classification. The N-grams are used to deal with the text part of the dataset, which compute the distance between the incoming stack traces and the historical set of bug reports stack traces. For the linear combination between the reports and non-textual features such as component and severity. This approach is applied to the Eclipse dataset which has thousands of bugs reports [20]. The main goal or purpose of detecting the root cause of the software bugs is to reduce the manual involvement in the software development process. Using the Eclipse bugs dataset, ten of the root cause categories are decided based on the most common bugs causes. Then, using ML algorithms, such as Naive Bayes, Maximum entropy, SVM, and Decision Trees. F1-score, precision, and recall are used as measurements [21]. Feature selection strategy is used at bugs' severity types detection process. The work shows that a ranking-based strategy and ensemble feature selection show better performance than using a commonly used maximization-based strategy based on F1-score measure using Eclipse, and Mozilla bugs' severity dataset [22]. Bugs' priority is very important to tell when the bug should be resolved, in case the process of filtering bug reports and assigning priority manually is very heavy, time and effort consuming. With the increasing number of bugs and bugs' reports, there need to design a model that can detect the bugs' priority. The provided model uses component name, summary, assignee, and reporter as features in the model that may affect bugs' priority. The model is a 5-layer deep learning RNN-LSTM neural network. The text-preprocessing step is applied to the bugs' summary. Bugs' report passes through five levels, Open, In-Progress, Resolved, Closed, and Reopened. Bug's priority is classified or labeled to low and high.

The results from this provided model are compared to others from models using ML algorithms, such as SVM, KNN. The study showed that LSTM reported the best performance results based on all performance measures [23]. Software bugs' datasets have many features that can be used at bugs' priority determining process. Problem title, summary, and component name features are used in this work. The textual data is converted to numerical formal to facilitate dealing with using the TF-IDF technique. PCA and NMF are used for feature reduction, besides clustering and classification algorithms. The authors in [24] proposed a novel approach for malware detection in Android devices since they are vulnerable for continuous attacks although they are the most popular. The authors used ensemble machine learning algorithms for malware and anomaly detection since using a single anomaly classifier will not be as effective as using ensemble classifiers. The authors proposed grid search N-gram system call sequence features to improve the accuracy of anomaly and malware detection in mobile-based devices.

Many works focus on using the bugs' reports to classify the bugs' severity as secure and non-secure bugs. Using text data and making preprocessing on the text by N-gram for feature extraction. Then use a machine learning algorithm, such as a stacking-based Naive Bayes classifier to classify the extracted features data. The model performance is measured using F1- score and compared to SVM and Decision Tree algorithms performance on the Eclipse dataset [15].

The used approach counts the probability for all the sentences for both classes 0 and 1according to the probability for each word in the sentence. Then, make a comparison between which class has the highest probability [25]. Three features are used to detect the bugs' severity, such as component name, product, and OS; because of the severity of the existed bug associated with the component and product. Besides these features, the stack trace is used in this model as a source of bugs' reports. KNN algorithm is used in this work, the used algorithm computes the nearest class of the bugs' severity for the inputted bug report [26]. The idea of bugs data study is the same, the core difference in what are the used ML algorithms, the feature selection techniques, the selected features from the bugs' reports, and the environment of the dataset. The final results are compared according to precision, recall, accuracy, F1-score. Some approaches depend on unstructured data, such as summaries, and descriptions of the bugs. For that, unstructured data need to be preprocessed using for example word embedding, tokenization [27]. Bugs reports are submitted significantly by the users, Stack Overflow is a website for asking and

answering different questions and problems. For that this website can be used to collect a dataset about potential bugs that may users suffer from as bugs' summaries and descriptions. Then, apply techniques for preprocessing the text. Using the maximum likelihood method to train the logic regression model [28]. Text mining techniques have a core role in analyzing and developing models of bugs' reports databases. Such as word embedding that captures the semantics of the text. The work shows the effectiveness of using word2vec for bugs' severity prediction process, the results show using a bigger window size improves the classifier's performance. Xgboost and Random Forest ML algorithms are used because of their ability for classifying the bugs' severity with a small number of records or infrequent appearance of words specific to each class.

3. Proposed System

In case most software programs have become large and complicated, software bugs need to take into consideration. Detecting and fixing bugs problem at initial levels that reflected positively on the quality, security, and performance of the program, and will save time and effort. Machine learning algorithms and Natural language processing techniques (NLP) have a great effort in software bugs classification and prediction. Depending on the historical bugs data ML and NLP can be used to build an automated model for classification and prediction software bugs instead of the manual way the be used by testers and editors.

With the emergence of new types of software bugs, there is a significant need to find new ways to detect and deal with them. Software bugs reports can be used to study and recognize new features and characteristics of new bugs. NLP techniques give a manner to analyze, study and extract characteristics from the text, which enable the testers and editors to recognize the software bugs deeply and facilitate the detecting and fixing processes.

In this research, the Eclipse software bug dataset will be used to develop a model that can automatically classify and predict the severity and priority of the bugs. The proposed model can be helpful and facilitate the testers' and editors' work during the testing process and the development life cycle generally.

The overall architecture of the proposed system can be found in figure (2).

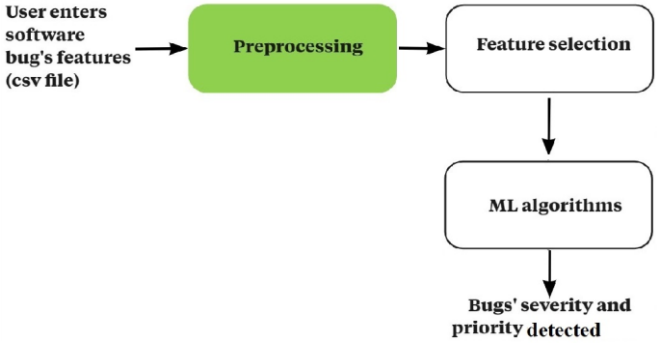

Figure 2: The overall architecture of our proposed Bugs' severity and priority classification and prediction system

3.1 Eclipse overview:

Eclipse is an open-source project since November 2000, is an integrating development tools platform, is an open and extensible architecture based on plug-ins [29]. Although Eclipse was developed for Java applications, plug-ins also allow the developers to create other applications with other languages, such as C, C++, COBOL, PHP, Python, and Perl. Using the plug-ins, Eclipse can work with network applications, database management systems, and modeling tools. IBM has established the Eclipse and gave to the open-source community. The consortium's primary goal was marketing and business affairs to provide code administrated and controlled by the Eclipse community.

The Java Development Tool (JDT) is a plug-in permit to use Eclipse as Java IDE. As for the PyDev plug-in permits Eclipse to be used as Python IDE. CDT is a plug-in that allows Eclipse to be used for developing applications using C/C++ [30].

The Eclipse Modeling Framework (EMF) is a code generator for creating software applications and tools based on a structured data model. EMF has a big advantage in providing tools and runtime support; to generate a set of Java classes for the model, as well as adapter classes for viewing and command-based editing, and a basic editor [31].

3.2 Dataset Overview

The dataset size is 10000 rows, that consists of 30 features describing the bugs. These features are: Bug ID, Product, Component, Assignee, Status, Resolution, Summary, Changed, Assignee Real Name, Classification, Flags, Hardware, Keywords, Number of Comments, Opened, OS, Priority, QA Contact, QA Contact Real Name, Reporter, Reporter Real Name, Severity, Tags, Summery, Target Milestone, URL, Version, Votes, Whiteboard, and Alias.

These features have different effects on determining the importance of each resulting bug, and thereafter its severity and priority. This dataset is collected from the Eclipse environment and includes various bugs that harm this IDE. It has two main classes, severity, and priority. The severity is divided into seven classes, major, blocker, critical, normal, enhancement, minor, and trivial, and three priority classes, high, middle, and low. These two main classes are used together to decide which bug should be solved first, which is the most harmful, and which is normal and can skip it.

The data set contains a summary as a short text description about the bugs. In this case, the text needs to deal with in another way, many features can be extracted from the text which is helpful in the classification process and identifying each type. The discrimination between the minor bugs from the blocker bugs for example can be learned. Figure 3 shows part of dataset features.

Figure 3: An overview on the Eclipse dataset

These features will be studied and analyzed to choose the most effective and representative features that will improve the final results.

3.3 Data Pre-processing and Cleaning

The most important aspect that was taken into consideration in the preprocessing phase was the feature selection. Because all features do not have the same importance level in the task, and each one may affect the final results differently. So, the features must be chosen in a way that efficiently serves the task. In addition, the dataset in not balanced, which that affect the accuracy degree that the results may be not accurate as should.

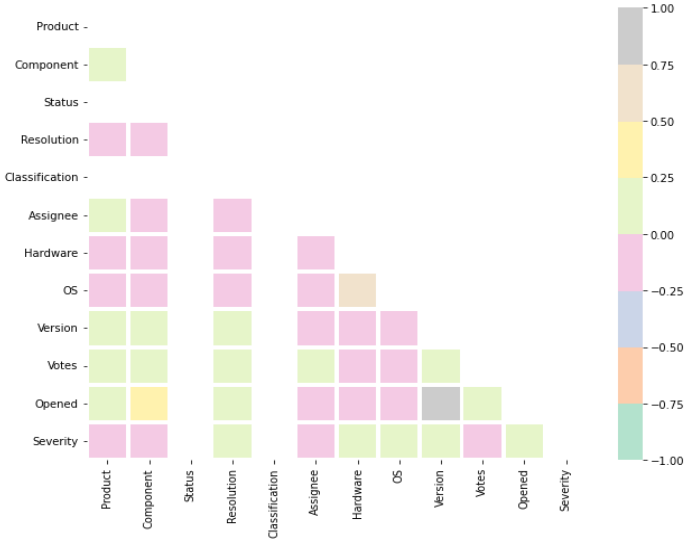

Figure 4: Correlation between the bugs' features, with severity as the target feature

3.3.1 Text Pre-processing

The text may contain numbers, special characters, foreign letters, or unwanted spaces. All of these reduce the efficiency of the model's work. In this step unwanted characters, punctuations, numbers, or spaces were removed from the description and the summary of the text in the dataset. The regular expressions from the Python library along with lemmatization were applied on the dataset. used to deal with the text cleaning process. The main goal of using lemmatization is to avoid producing features that are semantically similar but syntactically different.

3.3.2 Converting Text to Numbers

There are different approaches are used to convert the text data into numerical forms, such as the Bag of Words (BoW) model that will be used in this work. The Bag of Word model has one drawback as it cannot consider the frequency of the word that may vary from one document to another, or from text to another. To overcome this problem, Term Frequency-Inverse Document Frequency (TF-IDF) was also used. TF-IDF multiplies the word frequency by the inverse document frequency.

3.3.3 Label Encoding

In machine learning, most of the used datasets contain multiple labels and these labels may be words or numbers depending on the class that represents. Label encoding is converting labels into numeric form to be machine-readable. It is an essential pre-processing step when dealing with structured datasets in supervised learning.

The correlation between the chosen bugs' features is shown in Figure 4, the figure shows the relationship for each feature with the other which allows knowing how may affect each other. For example, the strongest relation as shown in the figure is between OS and Hardware. This figure helps to explain what are features that more affect the Severity as the target feature, Version, Hardware, OS, and Resolution have the highest correlation score.

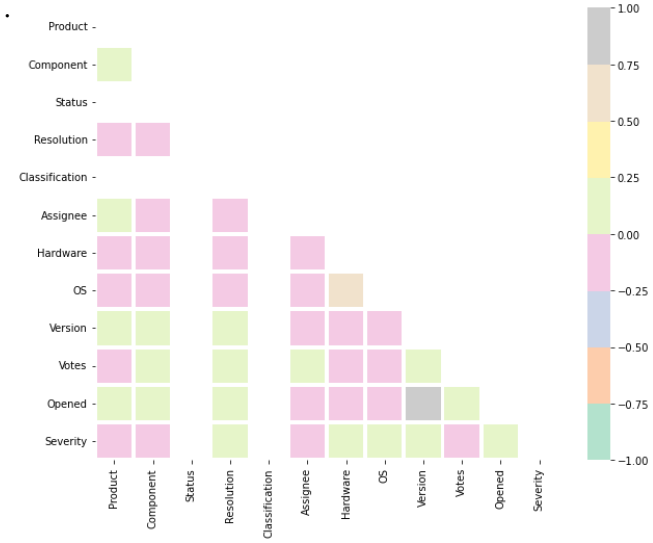

Figure 5: Correlation between the bugs' features, with priority as the target feature.

As for the Priority as the target feature, Figure 5 shows the correlation between the chosen features and how much affect each other. As clearly shown the Priority feature has the highest correlation score with the Version feature.

3.4 System Methodology

The most important and affected features will be used in the classification process such as Priority, OS, Hardware, Component, Product, Version, Votes, Resolution, Classification, Status, Severity. As for, the severity, and priority features will be used as a label for bugs classification and prediction alternately. The severity will be re-labeled as an integer number to facilitate dealing with it.

Because this work focuses and works on a multi-labeled dataset the challenge is to build a model which can distinguish between different bugs' severity and priority types. A series of Machine learning algorithms were experimented to assess their performance such as KNN , Random Forest, and SVM.

Dataset was not balanced and therefore the results may not be accurate. To overcome this issue we used two techniques: feature selection and oversampling (the Synthetic Minority Oversampling Technique (SMOTE)) along with ensemble learning method (e.g. voting classifier, bagging decision tree, Ada Boost). The evaluation metrics used are the level of accuracy, F1 score, precision, and recall. Also, the tf-idf will be used to study the text features, count the terms that are repeated in the corpus; that may help to distinguish between the types of bugs' severity.

4. Experimental Results and Discussion

In this work, several experiments were produced to show the power of the provided system. Different Machine Learning algorithms were applied in these experiments to detect Eclipse's software bugs.

Machine Learning algorithms were applied in two stages. In the first stage, the highest correlated features were used, such as Priority, OS, Hardware, Component, Product, Version, Votes, Resolution, Classification, Status, Severity, the Severity, and Priority were used as target feature alternately. As for the second stage, bugs' reports were used as a feature in the bugs detection process labeled by bugs' severity.

Table 1 summarizes the results obtained from several ML algorithms used. It also shows the final results of the system using different ML algorithms to make the detection process of the bugs' severity depending on the bugs' summary text. The used algorithms extracted features from the reports and used them in the detection process and distinguishing between severity different types

Table 2, on the other hand, shows the final results of the system using different ML algorithms to make the detection process of the bugs' priority depending on the bugs' summary text. The used algorithms extracted features from the reports and used them in the detection process and distinguishing between priority different types.

Table 2: The results of the system using the bugs' reports as system feature, and bugs' priority as target feature.

Algorithm Name	Accuracy	Training Time	Prediction Time	F1 score	Precision	Recall
multinomial NB	59 %	0.24	0.05 sec	73	59	96
Random Forest	60.4 %	0.30 sec	0.43 sec	73	63	87
SVM	61 %	10.5 sec	0.06 sec	73	65	84
Majority voting	65 %	43.0 sec	0.79 sec	79	67	95
Bagging	60.4 %	1729.4 sec	13.1 sec	77	73	83
Ada Boosting	59 %	111.83 sec	6.41 sec	73	61	90
Stochastic Gradient Descent	61 %	687.14 sec	0.14 sec	75	61	96

Tables 3 summarize the results obtained using several features' alternations and variations and their effect in the assessment process.

Table 3: The results of the system using the representative bugs' features, and bugs' severity as target feature

Algorithm Name	Accuracy	F1 score	ROC
Dummy classifier	65.6%	3.0	0.5
Random Forest	100%	1.0	1.0
KNN	92.4	7.4	8.2
Bagging | PCA	100%	1.0	1.0
Bagging | Multi	9.9	9.9	9.9
Ada Boosting	91%	5.3	7.7
SVC	99.8%	9.9	9.9

Table 4: The results of the system using the representative bugs' features, and bugs' severity as target feature/ Over-sampling.

Algorithm Name	Accuracy	F1 score	ROC
Ada Boosting	90.8%	7.9	9.2
Bagging	100%	1.0	1.0

Table 5: The results of the system using the representative bugs' features, and bugs' severity as target feature/ Feature Selection.

Algorithm Name	Accuracy	F1 score	ROC
Ada Boosting	91.1%	5.3	7.7
Bagging	90.2%	7.9	8.3

Table 6: The results of the system using the representative bugs' features, and bugs' priority as target feature

Algorithm Name	Accuracy	F1 score	ROC
Dummy classifier	64.6 %	2.0	0.5
Random Forest	100 %	1.0	1.0
KNN	94.5 %	8.2	8.8
Bagging | Multi	100%	1.0	1.0
Bagging | PCA	9.9	9.9	9.9
Ada Boosting	90.9%	5.3	7.7
SVC	99.8%	9.9	9.9

Table 7: The results of the system using the representative bugs' features, and bugs' priority as target feature/ Over-sampling.

Algorithm Name	Accuracy	F1 score	ROC
Ada Boosting	90.8%	7.8	9.2
Bagging	100%	1.0	1.0

Table 8: The results of the system using the representative bugs' features, and bugs' priority as target feature/ Feature Selection.

Algorithm Name	Accuracy	F1 score	ROC
Ada Boosting	90.9%	5.5	7.7
Bagging	100 %	9.9	9.9

4.1 Result Analysis

The previous tables show the effect of feature selection on the assessment process proposed in this paper. It is shown that when using bugs' summary as the main feature and data label with priority and severity alternately, the results range between 73% - 79% in reference to F1-score, since we are using unstructured data.

In terms of using structured data as model features, such as OS, Hardware, Component name, Product, Version, Votes, Resolution, Classification, Status, and data label with priority and severity alternately. These features were chosen based on the correlation degree between them and the priority and severity; to ensure the result is accurate, representative. The final results of using ML algorithms, such as KNN, Bagging, Ada-Boosting, SVC, Random Forest, show a very good performance based on F1-score, ROC curve, and accuracy measures.

In order to improve the result and the model performance, two techniques were used, Feature selection and Over-sampling techniques. SMOTE focuses on samples near the border of the optimal decision function and will generate samples in the opposite direction of the nearest neighbors' class and connect inliers and outliers.

For the feature-selection technique, according to the correlation result in Figures 4 and 5, nine features were used as they have highest relationship score with both bugs' severity and priority. In this case two ML algorithms were used, Bagging and Ada-Boosting, the final results were very good in reference to the F1-score, accuracy, and ROC curve measures.

5. Conclusion

This paper proposes a detection bugs' severity and priority system using the Eclipse bugs dataset. Since dealing with bugs process is not easy and take a huge effort, financial and time cost.

The proposed model passed on two phases for the detection of bugs. The first stage was bugs' summary which was preprocessed before being used in the model. Then, the machine learning algorithms, Naive Bayes, Random Forest, SVM, Ada Boosting, Bagging, KNN, and Stochastic Gradient Descent, were applied and trained the model to distinguish and detect the bugs' severity and priority types. To improve the performance two techniques were applied, Over-sampling, and feature selection, the performance improvement was noticed.

In the second phase, 12 representative features were selected from the 30 features from the dataset, Product, Component, Status, Resolution, Classification, Assignee, Hardware, OS, Version, Votes, and Opened. Then applied the same used ML algorithms and notice the model performance during the bugs' severity and priority detection process.

The model performance was better when the 12 features were used instead of using the bug's summary. The comparison process was done depending on Accuracy, F1 score, and ROC curve final results, before and after applying the over-sampling and feature-selection techniques.

References

[1] 360logica, "Difference between Defect, Error, Bug, Failure and Fault!," https://www.360logica.com/blog/difference-between-defect-error-bug-failure-and-fault/ Accessed in October 2021.

[2] Gema Rodríguez-Pérez, Gregorio Robles, Alexander Serebrenik, Andy Zaidman, Daniel M. Germán & Jesus M. Gonzalez-Barahona, "How bugs are born: a model to identify how bugs are introduced in software components," Springer Nature, p. 47, 2020.

[3] J. Unadkat, "6 Common Types of Software Bugs Every Tester Should Know," https://www.browserstack.com/guide/types-of-software-bugs, BrowserStack, June 2021, accessed in October 2021

[4] Sanket, "The exponential cost of fixing bugs," Deepsource, January 2019, Sanket, "The exponential cost of fixing bugs," DEEPSOURCE, 2019, accessed in October 2021.

[5] T. Sidorova, "Software Testing Basics: Types of Bugs and Why They Matter," ScienceSoft, https://www.scnsoft.com/software-testing/types-of-bugs, accessed in November 2021.

[6] Hassan, Zohaib, Iqbal, Naeem, Zaman, Abnash, "Towards Effective Analysis and Tracking of Mozilla and Eclipse Defects using Machine Learning Models based on Bugs Data," Soft Computing and Machine Intelligence, vol. 1, no. 1, 2021.

[7] Vikas Chandra, Leo Ufimtsev, David Williams and John Arthorne and others., "Eclipse/Bug Tracking," Eclipse Foundation, 2021, https://wiki.eclipse.org/Eclipse/Bug_Tracking, accessed in November 2021.

[8] Guo, S., Zhang, X., Yang, X. et al. Developer Activity Motivated Bug Triaging: Via Convolutional Neural Network. Neural Process Lett 51, 2589–2606 (2020). https://doi.org/10.1007/s11063-020-10213-y

[9] Ashima Kukkar 1, Rajni Mohana 1, Anand Nayyar 2, Jeamin Kim 3, Byeong-Gwon Kang 4,* and Naveen Chilamkurti 5, "A Novel Deep-Learning-Based Bug Severity Classification Technique Using Convolutional Neural Networks and Random Forest with Boosting," Sensors, p. 22, 2019.

[10] Awni Hammouri, Mustafa Hammad, Mohammad Alnabhan, Fatima Alsarayrah, "Software Bug Prediction using Machine Learning Approach," (IJACSA) International Journal of Advanced Computer Science and Applications,, vol. 9, p. 6, 2018.

[11] S. Delphine Immaculate; M. Farida Begam; M. Floramary, "Software Bug Prediction Using Supervised Machine Learning Algorithms," in 2019 International Conference on Data Science and Communication (IconDSC), Bangalore, India, 2019.

[12] Thamali Madhushani Adhikari; Yan Wu, "Classifying Software Vulnerabilities by Using the Bugs Framework," in 2020 8th International Symposium on Digital Forensics and Security (ISDFS), Beirut, Lebanon, 2020.

[13] Hufsa Mohsin a, Chongyang Shi a,∗, Shufeng Hao b, He Jiang c, "SPAN: A self-paced association augmentation and node embedding-based model for software bug classification and assignment," ELSEVIER, vol. 236, p. 107711, 2022 .

[14] QASIM UMER , HUI LIU , AND YASIR SULTAN, "Emotion Based Automated Priority Prediction for Bug Reports," in IEEE Access, Beijing 100081, China, 2018.

[15] SHAHID IQBAL1, RASHID NASEEM 2, SALMAN JAN 3, SAMI ALSHMRANY4, MUHAMMAD YASAR5, AND ARSHAD ALI4, "Determining Bug Prioritization Using Feature Reduction and Clustering With Classification," in Digital Object Identifier, 2020.

[16] Akimova, Elena N., Alexander Y. Bersenev, Artem A. Deikov, Konstantin S. Kobylkin, Anton V. Konygin, Ilya P. Mezentsev, and Vladimir E. Misilov. 2021. "A Survey on Software Defect Prediction Using Deep Learning" *Mathematics* 9, no. 11: 1180. https://doi.org/10.3390/math9111180

[17] Rudolf Ferenc, Dénes Bán, Tamás Grósz, Tibor Gyimóthy, Deep learning in static, metric-based bug prediction, Array, Volume 6, 2020, 100021, https://doi.org/10.1016/j.array.2020.100021.

[18] S. Iqbal, R. Naseem, S. Jan, S. Alshmrany, M. Yasar and A. Ali, "Determining Bug Prioritization Using Feature Reduction and Clustering With Classification," in *IEEE Access*, vol. 8, pp. 215661-215678, 2020, doi:

10.1109/ACCESS.2020.3035063.

[19] 1Syahana Nur'Ain Saharudin, 1Koh Tieng Wei and 2Kew Si Na, "Machine Learning Techniques for Software Bug Prediction: A Systematic Review," Journal of Computer Science, vol. 16, p. 12, 2020.

[20] Korosh Koochekian Sabor, Abdelwahab Hamou-Lhadj, Alf Larsson, "DURFEX: A Feature Extraction Technique for Efficient Detection of," in IEEE International Conference on Software Quality, Reliability and Security (QRS), Prague, Czech Republic, 2017.

[21] H. Lal and G. Pahwa, "Root cause analysis of software bugs using machine learning techniques," *2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence*, 2017, pp. 105-111, doi: 10.1109/CONFLUENCE.2017.7943132.

[22] Wenjie Liu, Shanshan Wang, Xin Chen and He Jiang, "Predicting the Severity of Bug Reports Based on Feature Selection," International Journal of Software Engineering and Knowledge Engineering, vol. 28, p. 22, 2018.

[23] Hani Bani-Salameh, Mohammed Sallam, Bashar Al shboul, "A Deep-Learning-Based Bug Priority Prediction Using RNN-LSTM Neural Networks," e-Informatica Software Engineering Journal, vol. 15, no. 1, p. 29–45, 2021.

[24] Nor Azman Mat Ariff, Mohd Zaki Mas'ud, Nazrulazhar Bahaman, Erman Hamid, and Noor Azleen Anuar, Ensemble Method for Mobile Malware Detection using N-Gram Sequences of System Calls. *International Journal of Communication Networks and Information Security (IJCNIS), 13*(2). doi:https://doi.org/10.54039/ijcnis.v13i2.4937 .

[25] Zaher Shuraym M. Alharthi, Ravi Rastogi, An efficient classification of secure and non-secure bug report material using machine learning method for cyber security, Materials Today: Proceedings, Volume 37, Part 2, 2021, Pages 2507-2512.

[26] Sabor, Korosh & Hamdaqa, Mohammad & Hamou-Lhadj, Abdelwahab. (2019). Automatic Prediction of the Severity of Bugs Using Stack Traces and Categorical Features. Information and Software Technology. 123. 106205. 10.1016/j.infsof.2019.106205.

[27] Luiz Alberto Ferreira Gomes a , ∗ , Ricardo da Silva Torres b , Mario Lúcio Côrtes b, "Bug report severity level prediction in open source software: A survey and research opportunities," Information and Software Technology, vol. 115, pp. 58-78, 2019.

[28] Youshuai Tan a , Sijie Xu a , Zhaowei Wang b , Tao Zhang c , ∗, Zhou Xu d , Xiapu Luo e, "Bug severity prediction using question-and-answer pairs from Stack Overflow," Journal of Systems and Software, volume 3, 2020.

[29] Catherine. Griffin, Introduction to the Eclipse Modeling Framework, IBM, 2003. https://www.omg.org/news/meetings/workshops/MDA_2003-2_Manual/Tutorial_4_Griffin.pdf, accessed in November 2021.

[30] "Eclipse - Overview," Tutorials Point. https://www.tutorialspoint.com/eclipse/eclipse_overview.htm, accessed in September 2021.

[31] Dave Steinberg, Frank Budinsky, Marcelo Paternostro, Ed Merks, EMF: Eclipse Modeling Framework, 2nd Edition, Addison-Wesley Professional., 2008.

Table 1: The results of the system using the bugs' reports as system feature, and bugs' severity as target feature.

Algorithm Name	Accuracy	Training Time	Prediction Time	F1 score	Precision	Recall
Multinomial NB	56.1 %	0.08	0.25 sec	79	65	99
Random Forest	64.5 %	0.63 sec	53.3 sec	78	67	95
SVM	64.3 %	0.07 sec	12.4 sec	78	66	95
Majority voting	65 %	1.02 sec	66.0 sec	79	67	95
Bagging	64 %	0.93 sec	2929.22 sec	77	67	92
Ada Boosting	64 %	80.3 sec	4.94 sec	78	66	97
Stochastic Gradient Descent	65 %	1067.1 sec	0.12 sec	79	66	97