IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 8, August 2022

DOI: 10.17148/IJARCCE.2022.11825

Bug Tracking System with severity prediction and triage assistance

Dhruv Agrawal [1], Priyansh Shah[2], Mrs. Veena Kulkarni [3]

Student, Department of Computer Engineering, Thakur College of Engineering and Technology, Mumbai, India[1]

Student, Department of Computer Engineering, St. Francis Institute of Technology, Mumbai, India[2]

Professor, Department of Computer Engineering, Thakur College of Engineering and Technology, Mumbai, India[3]

Abstract: Bug reports are an integral part of the software development process. They are used by software developers to improve the quality of the software. Bug triaging deals with the selection of a suitable developer to resolve the bug. With the increase in the number of bugs, this process only becomes troublesome and laborious. If the bug is assigned to a developer who is not able to resolve the bug, it is reassigned to another developer. This bug tossing leads to delays in resolving the bug and a lot of wasted resources. The selected bugs are then prioritized based on their severity and fixed according to the priority. If the severity is incorrectly reported, it results in a waste of time and resources.
In this paper, we present how classical machine learning classification algorithms can be used in bug tracking systems during the process of bug reporting to suggest the severity of the bug and developers(assignee) using NLP (Natural Language Processing) techniques on the summary of the bug report. The predictions from these classification algorithms are then incorporated in the bug report filing and assignment phase of the bug life cycle.
We have collected bug reports from Bugzilla for two open-source projects: Eclipse and LibreOffice and compared the results of various classification algorithms. Even though fully automated assignment is not present, the prediction accuracies are high enough to be used as suggestions to the reporter/assigner in our bug tracking system.

Keywords: Severity Prediction, Bug Triaging, Project management, text-based classification, NLP, TF-IDF.

I. INTRODUCTION

In the software development process, a software bug is an unavoidable and undesirable aspect. The software development teams always work to minimize the bugs in the software and prevent them from occurring again. The process of finding, reporting, tracking, assigning to resolve, and finally resolving the bug is a long and tedious process. Hence, the purpose of the proposed solution is to minimize the errors and develop a single interface for the same. In our bug tracking system, the tester can report the bug whose severity and assignee will be predicted using classification techniques. It will then be suggested to the testing team and the team lead respectively.
A bug report is something that stores all the information needed to document, report and fix issues(bugs) in software. A bug tracking system is a piece of software that maintains track of reported software issues in SDLC. The fundamental advantage of a bug-tracking system is that it provides a centralized view of development requests (both bugs and enhancements; the line between the two is sometimes blurry) and their status. When designing the product road map, or perhaps just "the next release," the prioritized list of pending items (commonly referred to as the backlog) gives essential feedback.
Bug report assignment, also known as “Bug Triage” is one of the important parts of software maintenance. It deals with the selection of a suitable software developer for handling reported bugs such that the assigned developer can fix the reported issue. Incorrect assignment of bug reports to developers can be very expensive in large software projects and is one of the bottlenecks in the bug resolution process.

A. Phases of the Bug Life cycle

The different Phases of the Bug Life cycle are:-
New: - Whenever a new bug is detected, it is in a new state. The tester finds a new bug and reports it to the developers in a bug report.
Assigned: - Once the ‘new’ bug is viewed by the team lead, they assign the bug to someone in the development team for the resolution process and the status is changed to ‘assigned’.
Open: - When the bug is assigned to the developers for the resolution, it is in the open state. It remains in an open state while the development teams are working on the bug. From there, the teams have an option of transferring the bug to the ‘rejected’ state if they feel that the bug is not appropriate.
Fixed: - If the development team is successful in resolving the bug, then the status is changed to ‘fixed’ and the report is closed.

DOI: 10.17148/IJARCCE.2022.11825

Pending retest: - While fixing the bug, the developer team comes up with new code, the same needs to be tested before being deployed. Thus the code is sent for testing and at this point, the status of the bug is ‘pending retest’.
Retest: - The tester receives the fix in the pending retest state and the code needs to be retested before deploying. The tester thus checks the code again and changes the status to retesting.
Reopen: - While testing the new code, if the tester finds that the bug is not resolved, the bug status is set as ‘reopen’ and the entire process is followed again from step New to step Retest.
Verified: - Once the new code is received by the tester in the ‘pending retest’ state, if the code is the proper fix, then the status is changed to the verified state.
Closed:-It is the last stage of the bug life cycle that signals the end of the bug life cycle.

II. BACKGROUND AND RELATED WORK

The majority of the time only a few bugs are selected according to business needs and available resources. The selected bugs are prioritized (ordered) based on their severity (For example, the bug prevents an important feature of the product from functioning, or the bug affects a big number of consumers) and then addressed in order of priority. If the severity is erroneously assessed, time and resources may be lost in attempting to resolve the bug. We propose a bug tracking system that not only predicts the severity of the bug but also suggests the developer to which it should be assigned based on the bug report.
L. Johnson, M. Borg, and D. Broman [1] in their paper studied the ensemble learner Stacked Generalization(SG) on large-scale proprietary projects in the industry. They also stated the reasons why general-purpose classifiers should be used instead of other more specific previous approaches.
F. Servant and J. A. Jones [2] described a technique that automatically selects the most appropriate developers for fixing the fault represented by a failing test case, and provides a diagnosis of where to look for the fault. This technique method was effective because of these crucial components: issue finder to determine which locations execution corresponds with failure, commit history, and expert allocation to map locations to developers.
T. Zimmermann, R. Premraj, J. Sillito, and S. Breu [3], in their paper, addressed the concerns of bug tracking systems by proposing four broad directions for enhancement : Tool centric, Information, Process and User centric. As proof of concept, they used a decision tree for predicting the location of bugs using eclipse jdk bug reports for the twenty most fixed files.
T. S. Gadge and N. Mangrulkar [4] suggested variously supervised and semi-supervised approaches for labeled and unlabeled data. They used textual data and suggested approach based on a history of an expert known as the vector space model and also is independent of the history of an expert which is a vocabulary-based expertise model. A Triage assisting technique known as MLtriage has been suggested which is a supervised ML algorithm. To calculate the cost of model content boosted collaborative filtering(CBCF) is used.
Pushpalatha M.N, Mrunalini M [5], they have taken NASA’s PITS projects as a source of bug reports for predicting the severity of bugs. They used common preprocessing methods and with that compared some ensemble classification algorithms to measure the accuracy of the severity of the bugs.
Prof. A. F. Otoom, Al-Shdaifat, M. Hammad and E. E. Abdullah [6] have focused on prediction of severity of bugs and for that they have used Bugzilla as their input. They have done pre-processing by tokenization and stemming and have trained the datasets with various classification machine learning algorithms and finally evaluated the results.
A Goyal, N Sardana [7] in their paper have taken the experimental datasets from few open source bug reporting tools to focus on automatic bug assignment to suitable devs. For the examination of results, they created close to 400 machine learning models which would help them predict the assignee for the bug reports.
Pushpalatha M.N, Mrunalini M [8], in their paper like others have collected open source Bug reports from Bugzilla and with that they have compared the accuracy and precision of two algorithms such as Bagging and J48 classification algorithm while predicting the severity of bug reports. The results showed that Bagging showed better results than J48 algorithm.
Shikai Guo *et al* [10] described a technique that is based CNN(called as CNN-AD) along with some other algorithms and dev’s engagement. Their main aim was to focus on improving the performance of bug triaging. The datasets were taken from NetBeans, Eclipse and other open source bug reporting repositories and data pre-processing and word vector was done. The input was then fed to their technique for the prediction of bug resolver/ assignee.
Jude Arokiam and Jeremy S. Bradbury [11], in their paper have focused mainly on how to predict the bug severity at the initial stage of SDLC. Their work depends upon the bug reports of the precedent projects of the company for which the new project’s bug severity prediction needs to be done. They have trained their classifier by passing the document vector form of the precedent project’s bug report. After that, they fed the latest project’s bug report summary (in document vector format) as input to the classifier in order to predict the severity of the bugs.
Vedang Mondreti and Prof. Satish C.J. [12] have proposed a Bug Prediction System where eXtreme Gradient Boosting Framework is used for prediction of severity along with training & testing of model and Synthetic Minority Over-

Sampling Technique algorithm is used for data pre-processing (reducing the imbalance of the dataset). They fetched the datasets from BugZilla and conducted few case studies. Their result showed that the framework used by them returned good accuracy whilst comparing it with other algorithms used for prediction of bug severity.
Dong-Gun Lee and Yeong-Seok Seo [13] in their paper focused on improving the existing LDA (Latent Dirichlet Allocation) based classification algorithm with the help of their proposed method. Earlier, hybrid methods for bug triaging were used but it had compatibility issues, so instead of using LDA with other methods, they have focused on how to improve the LDA itself. Additionally, they have focused on rectifying mis-triage of bug reports along with reclassifying them.
Rashmi Agrawal and Rinkaj Goyal [14] have used word2vec as their data capturing tool from open source bug reports of Jira and Bugzilla. They have also used three averaging methods viz. Tfidf, mean and Univac mean and compared their performance with up to six classification algorithms. The conclusion they drawn was that mean averaging method performed better with most of the classifiers compared to Tfidf and Univac mean.

III. PROPOSED METHODOLOGY

We propose a bug tracking system with the following modules shown in Fig.1. The Fig. 2. depicts the phase in the bug life cycle at which we propose the solution for bug triage and severity prediction. We have used NLP classification algorithms on the summary of the bug report to predict bug severity as well as the developer to whom the bug will be assigned (assignee). During the filing of the bug report, the reporter will get the suggestion for the severity based on the summary entered. After the bug report has been filed, the assigner (usually the team lead) will be suggested the assignee. The assignee will then work to fix that defect/bug and send it to the testing team once that bug is fixed. The testing team will retest that bug and change the status of that bug accordingly.

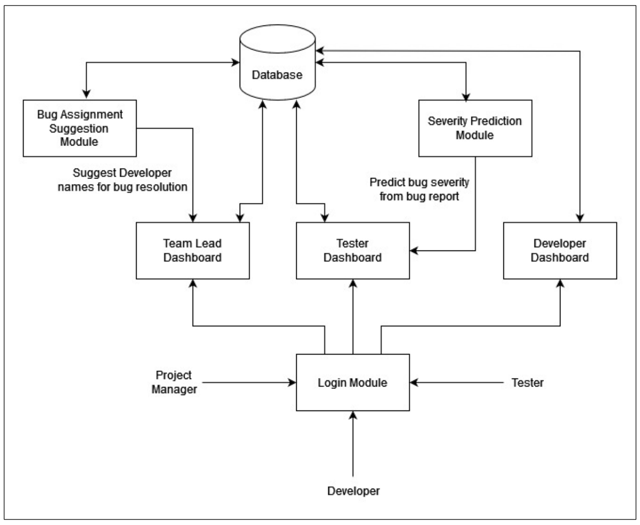

Database
Bug Assignment Suggestion Module
Severity Prediction Module
Suggest Developer names for bug resolution
Predict bug severity from bug report
Team Lead Dashboard
Tester Dashboard
Developer Dashboard
Project Manager
Login Module
Tester
Developer

Fig. 1 Block Diagram of Bug Tracking System

A. Tester/Bug reporter module
The testers will have the UI where the tester can report the bug and receive a suggestion for severity. Then the tester can decide whether he wants to accept the suggestion for bug severity or not.
B. Team Lead module
The team lead (TL) can view the bug reports submitted by the testing team. After this, they will get a suggestion while choosing an assignee for the bug.

C. Login Module
The software provides a different login interface for all the entities involved in the project.
D. Analysis and Prediction Module
The working of this module is hidden from all three entities. Once the tester submits the bug report, the system will perform the text-based classification and process the report to predict the severity. Along with that, the system also provides the suggestion of the developer to whom the resolution process can be assigned.
E. Database module
The bug once resolved to be updated in the database so the history can be used for further training of the model.

DOI: 10.17148/IJARCCE.2022.11825

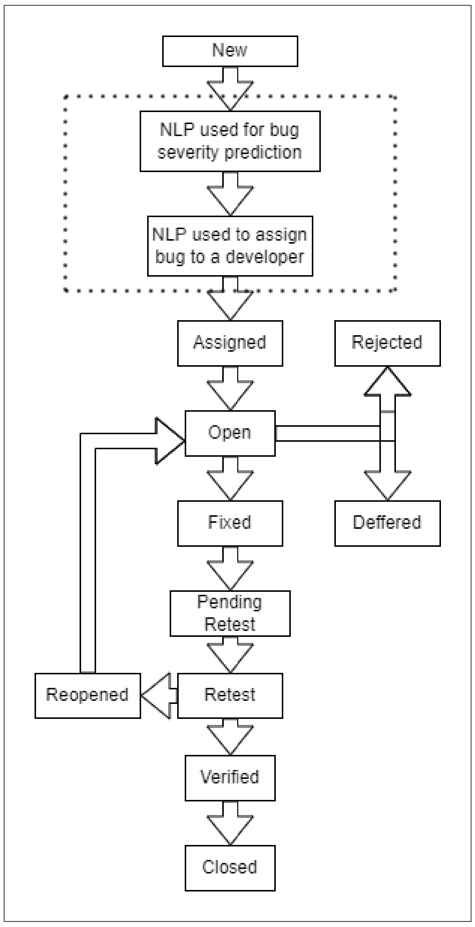


Fig. 2 Our approach in the bug life cycle

F. ML Modules

Initially as depicted in fig. 3, the tester files the bug report with a detailed description of it in the summary field using technical keywords like the module in which it was found and the functionality which is failing. In the next step, Data mining is done to turn raw data into relevant information.

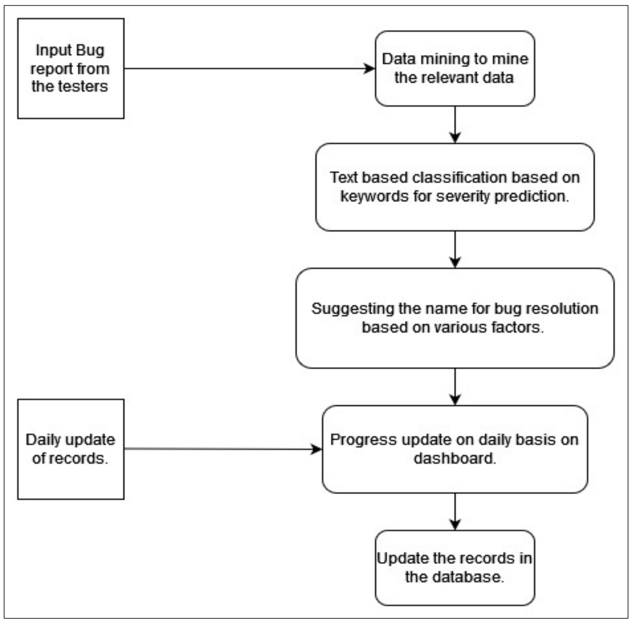

Input Bug report from the testers
Data mining to mine the relevant data
Text based classification based on keywords for severity prediction.
Suggesting the name for bug resolution based on various factors.
Daily update of records.
Progress update on daily basis on dashboard.
Update the records in the database.

Fig. 3 Flowchart of ML Modules

After that, Text-based classification is used on the summary provided by the tester. This helps in extracting generic tags/features from unstructured text. With this, the severity of the bugs will be predicted. Moreover, according to severity, the assignee will be predicted to resolve the bugs. These are then suggested to the tester which he/she can consider for finally submitting the report. Daily updates of records will be done by developers, testers, and managers. Progress updates will be visible on their respective dashboards. These updates will be committed to the database regularly.

IV. IMPLEMENTATION

We are using the summary of the bug report for both bug triage assistance and severity prediction. For both of them, we have used Natural Language Processing(NLP) techniques in the summary of the bug reports. To implement NLP in python, Natural Language Toolkit(NLTK) was used. We have used conventional classification algorithms and compared their performance. The reason we used them is to show that even without specialized techniques, we can get substantial results that can be generalized to other projects and thus help in reducing bug tossing.

A. Dataset

The project is based on supervised learning. To train and test, we need access to a software project's bug reports. Since the bug reports are highly confidential for the company, we selected the bug reports of open source projects like Libre Office and Eclipse. We mined the bug reports of these projects from Bugzilla. The software is intended for the software development teams of the firms.
As the dataset used was of open-source software, there were a few notable differences which we had to counter summarized as below: -

1. The conventional software development teams consist of a limited number of members within the team whereas the scope of open-source software is open to all.
2. The distribution of tasks in a development team is generally balanced whereas that in open-source is highly unbalanced.
3. The open-source dataset is highly skewed and thus requires a large amount of preprocessing.

B. Data Preprocessing and Augmentation

To resolve the above-mentioned issues, the team did the following actions as a part of preprocessing to make the dataset suitable for the algorithm: -

1. As the dataset was highly imbalanced, there was a need to discard the outliers. The dataset consisted of a huge amount of entries of developers resolving bugs only a few amount of times. (1 to 5)
2. There was a huge discrepancy in the distribution of data as a result, data augmentation was performed on the dataset.

We dropped the assignees who had fixed very small numbers of bugs and also those who had fixed disproportionately large number of bugs. Because of the dataset's characteristics, this was necessary. Since the dataset is based on an open-source project, the dataset is very skewed because of occasional contributors. Then to further balance and increase the size of the now reduced dataset, we performed data augmentation.
Data augmentation is the technique of increasing the size and diversity of the dataset used for training a model in machine learning. Since the size of the dataset reduced considerably, we used data augmentation to increase and balance it.

We have used Easy Data Augmentation (EDA) [9] for performing data augmentation for our text classification model. It consists of four operations:

1. Random deletion: Removes a word from a sentence with probability p
2. Random swap: Randomly choose two words in the sentence and swap them n times.
3. Synonym replacement: Choose n-words randomly which are not stop-words and replace them with their synonyms
4. Random Insertion: Choose a random word and insert its synonym in a random position n times.

The resulting dataset thus generated improved the accuracy and F1 score to a great extent.
In the Eclipse dataset, for severity, after cleaning the data and performing EDA, the total number of records is 10,000 and the total number of classes (severity types) is 7, viz.: major, minor, critical, enhancement, normal, trivial, and blocker. And for assignee prediction, 3295 records out of 10,000 were selected after EDA having a total of 15 classes(assignees).
Next, we removed stop words, and punctuation and performed lemmatization. We used stop words from the corpus package and WordNetLemmatizer from the stem package of nltk library.
Lemmatization: It is one of the most common text pre-processing techniques used in Natural Language Processing (NLP). It is the act of combining the many inflected forms of a word so that they may be studied as a single item. It is similar to Stemming but the key difference is Lemmatization brings context to the word.

C. Feature Extraction

For feature extraction, we used TF-IDF (Term Frequency-Inverse Document Frequency).

TF-IDF: Term Frequency Inverse Document Frequency of records is the calculation of how relevant a word in a series or dataset is to a text. The tf-idf weight consists of two terms: Normalized Term Frequency (tf), and Inverse Document Frequency (idf).

Normalized Term Frequency (tf): It is the count of word t in document d divided by the number of words in document d.
Inverse Document Frequency (idf): The idf of the word is the number of documents in the dataset separated by the frequency of the text.
tf-idf(t, d) = tf(t, d) * idf(t)

Since TF-IDF weights words based on relevance, it can be used to determine that the words with the highest relevance are the most important. Hence, we used tf-idf technique on the summary of the bug report provided by the testing team. It helps us to convert text into a matrix (or vector) of features.

D. Finding the Severity and Assignee

For severity and assignee suggestion, we have compared different classification algorithms like SVM, Logistic Regression, Multinomial Naive Bayes, and Decision Tree Classifier.

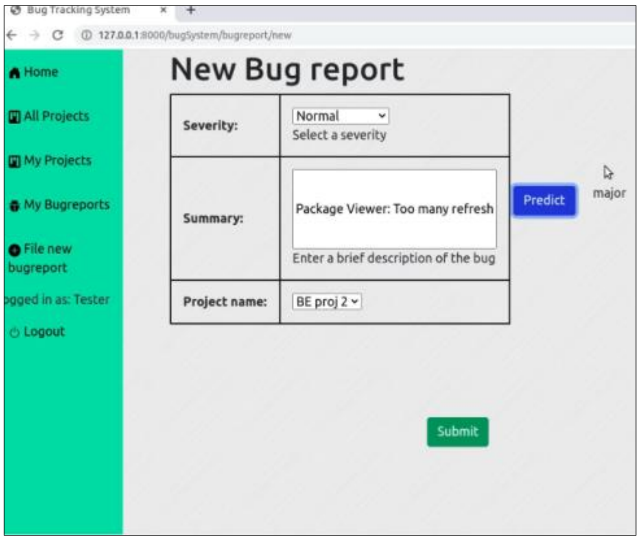
Support Vector Machine(SVM): This supervised machine learning algorithm can do non-linear classification effectively by employing the kernel technique, which involves implicitly mapping its inputs into high-dimensional feature spaces.
Logistic Regression: This type of statistical analysis is often used for predictive analytics and modeling. It helps us to predict the likelihood of an event happening or a choice being made.

Multinomial Naive Bayes: This popular algorithm in Natural Language Processing (NLP) assesses the likelihood of each tag for a particular sample and returns the tag with the highest probability. In this, a feature's existence or absence has no bearing on the inclusion or exclusion of another feature.

Decision Tree Classifier: A decision tree is a flowchart-like tree structure in which the internal node represents a feature, the branch represents a decision rule, and each leaf node represents the outcome. They offer a highly effective framework within which you can set out possibilities and analyze the implications of those options.

The trained model then returns the severity and assignee for the entered summary which can then be used by the tester to better judge the severity of the bug while submitting the report and assigning it to a developer.

The newly submitted bug reports can then be used again for training the machine learning model for improvements.

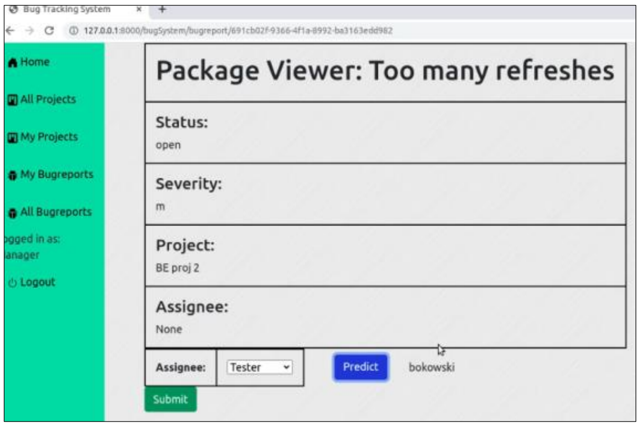

Bug Tracking System
127.0.0.1:8000/bugSystem/bugreport/new

New Bug report

Severity:	Normal Select a severity
Summary:	Package Viewer: Too many refresh Enter a brief description of the bug
Project name:	BE proj 2

major

Fig. 4 Bug Severity Prediction


Bug Tracking System

127.0.0.1:8000/bugSystem/bugreport/691cb02f-9366-4f1a-8992-ba3163edd982

Home
All Projects
My Projects
My Bugreports
All Bugreports
ogged in as:
anager
Logout

Package Viewer: Too many refreshes

Status:
open

Severity:
m

Project:
BE proj 2

Assignee:
None

Assignee: Tester
Predict
bokowski
Submit

Fig. 5 Bug Assignee Prediction

V. RESULT AND DISCUSSION

The results are shown in tables I, II, III, and IV depicting the accuracy, precision score, recall score, and F1 score. We have compared the results of 4 classification algorithms: SVM, Multinomial Naive Bayes, Logistic Regression, and Decision Tree Classifier. These were tested on bug reports from Bugzilla of open source projects Eclipse and LibreOffice. Of these 4, we found SVM to be the better performing algorithm for our datasets.

$$F1\text{-}score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{Precision}=\frac{\textit{Number of bug reports correctly predicted as C}}{\textit{Number of bug report predicted as C}}$$

$$\text{Recall}=\frac{\textit{Number of bug reports correctly predicted as C}}{\textit{Number of bug reports of class C}}$$

$$\text{Accuracy}=\frac{\textit{predicted correctly}}{\textit{total number of prediction}}$$

TABLE I LIBREOFFICE ASSIGNEE PREDICTION

Dataset	Algorithm	Accuracy	Precision Score	Recall Score	F1 Score
LibreOffice	SVM	0.84	0.84	0.84	0.84
	Multinom ial Naive Bayes	0.53	0.75	0.53	0.47
	Logistic Regression	0.77	0.80	0.77	0.77
	Decision Tree Classifier	0.83	0.83	0.83	0.83

TABLE II LIBREOFFICE SEVERITY PREDICTION

Dataset	Algorithm	Accuracy	Precision Score	Recall Score	F1 Score
LibreOffice	SVM	0.63	0.55	0.63	0.58
	Multinom ial Naive Bayes	0.66	0.44	0.66	0.53
	Logistic Regression	0.67	0.59	0.67	0.55
	Decision Tree Classifier	0.50	0.51	0.50	0.52

DOI: 10.17148/IJARCCE.2022.11825

TABLE III ECLIPSE SEVEIRTY PREDICTION

Dataset	Algorithm	Accuracy	Precision Score	Recall Score	F1 Score
Eclipse	SVM	0.61	0.53	0.61	0.56
	Multinomial Naive Bayes	0.65	0.49	0.65	0.51
	Logistic Regression	0.65	0.53	0.65	0.54
	Decision Tree Classifier	0.53	0.51	0.53	0.52

TABLE IV ECLIPSE ASSIGNEE PREDICTION

Dataset	Algorithm	Accuracy	Precision Score	Recall Score	F1 Score
Eclipse	SVM	0.54	0.53	0.54	0.53
	Multinomial Naive Bayes	0.45	0.44	0.45	0.39
	Logistic Regression	0.52	0.57	0.52	0.49
	Decision Tree Classifier	0.46	0.47	0.46	0.46

VI. CONCLUSION

The severity of software issues and assignee must be assessed for team leads to prioritize and assign bug reports. It is labor-intensive and time-consuming if done manually. As a result, automating the process becomes critical. Current bug tracking systems do not have any feature where severity and assignee can be predicted. With our solution, bug tossing can be reduced which eventually saves a lot of time by suggesting the severity of the bug and assigning a developer for the same in the SDLC.
Additionally, we have concluded that even with classical classification algorithms we can generate substantial results for the prediction of severity and assignee which can be used in a real bug tracking system as evident from our proposed bug tracking system.

REFERENCES

[1] Leif Jonsson, Markus Borg, David Broman, et al. “Automated bug assignment: ensemble-based machine learning in large scale industrial contexts”. Empir software eng 21, pp. 1533–1578 (2016).

[2] F. Servant and J. A. Jones, "Whosefault: Automatic Developer-to-fault assignment through fault localization," 2012 34th International Conference on Software Engineering (ICSE), Zurich, 2012, pp. 36-46

[3] T. Zimmermann, R. Premraj, J. Sillito, and S. Breu, "Improving bug tracking systems," 2009 31st International Conference on software engineering - companion volume, Vancouver, BC, 2009, pp. 247-250.

[4] S. Gadge and N. Mangrulkar, "Approaches for automated bug triaging: a review," 2017 International Conference on innovative mechanisms for industry applications (ICIMIA), Bangalore, 2017, pp. 158-161.

[5] Pushpalatha M.N., Mrunalini M., “Predicting the severity of closed source bug reports using ensemble methods.” In: satapathy s., bhateja v., das s. (EDS) smart intelligent computing and applications. Smart innovation, systems, and technologies, vol 105. Springer, Singapore, 2017, pp. 589-597

[6] A. F. Otoom, D. Al-Shdaifat, M. Hammad and E. E. Abdallah, "Severity prediction of software bugs," 2016 7th International Conference on Information and Communication Systems (ICICS), Irbid, 2016, pp. 92-95

[7] A. Goyal and N. Sardana, "Empirical Analysis of Ensemble Machine learning techniques for Bug Triaging," 2019 Twelfth International Conference on contemporary computing (IC3), Noida, India, 2019, pp. 1-6.

[8] M N. Pushpalatha and M. Mrunalini, "Predicting the severity of bug reports using Classification algorithms, "2016 International Conference on circuits, controls, communications and computing (I4C), Bangalore, 2016, pp. 1-4.

[9] Github.com, ‘EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks', 2019. [Online]. Available: https://github.com/jasonwei20/eda_nlp. [Accessed: Sept 2020].

DOI: 10.17148/IJARCCE.2022.11825

[10] Shikai Guo, Xinyi Zhang, Xi Yang, Rong Chen, Chen Guo, Hui Li, Tingting Li, " Developer Activity Motivated Bug Triaging: Via Convolutional Neural Network" Springer Science+Business Media, Llc, Part Of Springer Nature 2020, Pp: 2589–2606

[11] Jude Arokiam, Jeremy S. Bradbury, "Automatically Predicting bug severity early in the development process", 2020 IEEE/ACM 42nd international conference on software engineering: new ideas and emerging results (ICSE-NIER), pp:17-20

[12] Vedang Mondreti, Prof. Satish C.J., "Bug Severity Prediction System using Xgboost Framework", 2020 IEEE International Conference on machine learning and applied network technologies (ICMLANT).

[13] Dong-Gun Lee, Yeong-Seok Seo, "Improving Bug Report Triage Performance using Artificial Intelligence-based document generation model", Lee and Seo hum. Cent. Comput. Inf. Sci. (2020).

[14] Rashmi Agrawal, Rinkaj Goyal, "Developing Bug Severity Prediction Models Using Word2vec", International Journal Of Cognitive Computing In Engineering 2 (2021).