BUGFORECAST: A NATURE-INSPIRED ENSEMBLE MODEL FOR PREDICTING SOFTWARE BUGS

B. Rithya Reddy
Scholar, Department of MCA
Vaageswari College Of Engineering-Karimnagar

P. Sathish
Supervisor, Department of MCA
Vaageswari College of Engineering-Karimnagar

Dr.V.Bapuji
Professor & Head, Department of MCA
Vaageswari College of Engineering-Karimnagar

ABSTRACT: One essential and practical technique to improve the quality and dependability of software is to anticipate software issues. Improving project management entails identifying release delays early on and implementing cost-effective remedies to improve program quality. One approach is to estimate with information which areas of a complex software system are most likely to contain a substantial number of errors in future versions. However, it is challenging to develop models that can accurately detect errors. The primary purpose of this paper is to explore the efficiency of predictive analysis in software development systems in connection to two types of software issues: criticality and severity. The machine learning algorithm used in this paper was written in Python. This work employs statistical approaches, modeling, machine learning, data mining, and artificial intelligence. Predictive models can be used to optimize the distribution of research resources. During machine learning model training, the seriousness and urgency of a problem are assessed using two approaches: Random Forest (RF) Classifier and Support Vector Machine (SVM). The paper findings demonstrate that, with an accuracy rate of 0.87, The RF Priority Model provides a comprehensive understanding of the model's performance across various priority levels. This inquiry use data mining techniques to detect flaws in the present software configuration. Allowing developers to generate software will improve software quality while lowering development and maintenance expenses.

Keywords: Predictive Analysis; Predictive Models; Software Bugs; Priority; Severity; SVM; Random Forest Classifier.

1.INTRODUCTION

Software defect prediction (SDP) is the proactive identification of underperforming software system components. According to Ali et al. (2022), cloud-based solutions are significant at all stages of the software development life cycle. This process likewise takes a long time (Shatnawi et al., 2022). In most cases, prompt debugging is required to identify and anticipate potential problems. The severity of the discovered errors or mistakes will determine how best to proceed. Furthermore, one of the most critical first stages in the software development process is requirements gathering. Akmel et al. (2017) argue that software testing should begin as early as possible in the development process to ensure that all criteria are met. Software defects are malfunctions, errors, or omissions in software that cause the program to behave unexpectedly, contrary to end users' expectations and software engineers' quality goals (Ali et al., 2019; Olaleye et al., 2021).

All software product releases have software defects caused by poor testing (Zhang et al., 2018). Users should report any issues they discover using issue tracking systems such as Bugzilla, Mantis, Google Code Issue Tracker, GitHub Issue Tracker, JIRA, and others. This prevents the same mistakes from recurring in future versions or new applications. Users often assist engineers in finding and reporting defects, which is a common component of the software maintenance method.

Data concerns are rather common in cloud services. Among the various software defects are the following: logic problems in the cloud (29%), load concerns (4%), space constraints (4%), insufficient error handling (18%), optimization challenges (15%), setup errors (14%), data races (12%), and system hang-ups (4%). Complicating matters further, practically any type of flaw can cause a slew of problems, including complete inoperability, sluggish performance, damaged components, data loss, retrieval of out-of-date information, and corruption. The purpose of this research is to use software prediction techniques to determine the priority and severity of software errors. The goal is to keep flaws out of the system. The historical literature on this issue is summarized in the section that follows.

2. LITERATURE REVIEW

Malgonde and Chari (2019) demonstrated through computer studies that the ensemble-based strategy outperformed earlier ensemble-based benchmarking techniques. The work utilized an optimization model and a dataset to enhance the sprint planning process for two projects. Both the ensemble-based technique and the prediction model were found to be useful in real-world scenarios.

Shetty et al. (2010) demonstrated the effectiveness of error correction through the implementation of a rule-based expert system. Furthermore, they recommended employing an analytical method to accurately assess the success of various repair options. An experiment using a specially constructed prototype demonstrates that, in compared to a reactive fault management system, it is less expensive to run and easier to use.

According to Ahmed and colleagues (2021), the system was developed using natural language processing and supervised machine learning. Eclipse and Mozilla submitted over 2,000 issue reports, which were reviewed. Four classification techniques were used to classify and score the problem reports: Logistic Regression, Decision Tree, Random Forest, and Naive Bayes. The CaPBug framework, which combines an RF classifier with a textual feature, shown how to reliably forecast the category. Furthermore, the system achieved an accuracy rate of 88.78%.

In the same vein, the CaPBug system estimated the relevance of bug reports with a 90% accuracy rate.

Class incompatibility in priority classes has been successfully handled by implementing SMOTE. Separate the concepts of resilience and dependability (SMOTE).

Pachouly et al. (2022) emphasized the importance of conducting a comprehensive paper on software defect prediction, encompassing datasets, data validation processes, defect identification, prediction algorithms, and tools. Furthermore, the literature review found that traditional datasets had fewer names than alternative datasets, making understanding the problems under research more challenging. The work introduced a concept for creating a dataset of software predictions with the right features. Furthermore, statistical data validation techniques were employed to accurately categorize software faults.

Alsaedi et al. (2023) state that BR analysis was utilized to provide a novel prediction model for identifying the types of bugs that will be present.

We propose a new learning technique dubbed "ensemble learning" that blends natural language processing (NLP) and machine learning (ML). The simulations revealed that with text augmentation, the suggested model was 96.72% accurate and 90.42% without. These findings reveal that the suggested model is more accurate than most other models already in use.

According to an earlier paper, predicting software failures is an important and practical technique to improve software quality and reliability. Using early estimation approaches in project management identifies problems with releases and cost-effective remedial procedures, resulting in better software (Alsghaier et al., 201). Predictive models can identify critical components of a large software system that are expected to have a significant number of problems in future versions.

However, reliability in defect prediction models is difficult to achieve, and the combined papers demonstrate a variety of approaches. The primary purpose of this work is to analyze and investigate software development models. This would allow us to foresee and comprehend the characteristics, importance, and seriousness of software problems.

3. METHODOLOGY

The severity and priority levels assigned to each software problem make it impossible to completely eliminate all bugs, fixes, patches, and other difficulties. This investigation was conducted using Python. The stages that follow describe how the prediction model is implemented.

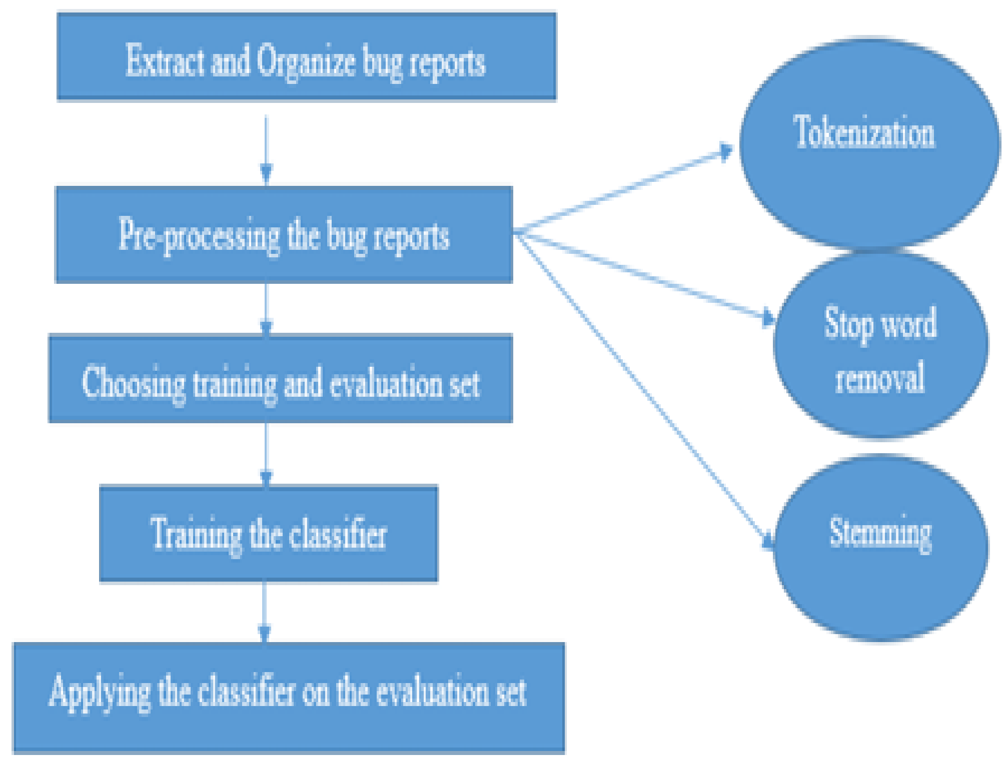

Fig 1: Proposed Methodology

DATASET

This paper employs a dataset to investigate the feasibility of software development tool predictive analysis in determining the severity and impact of software problems. The dataset for this work was created using five open-source tools: Apache, JIRA, JBoss, MongoDB, and Spring. Each of the five open-source programs' websites provided issue reports with a variety of information.

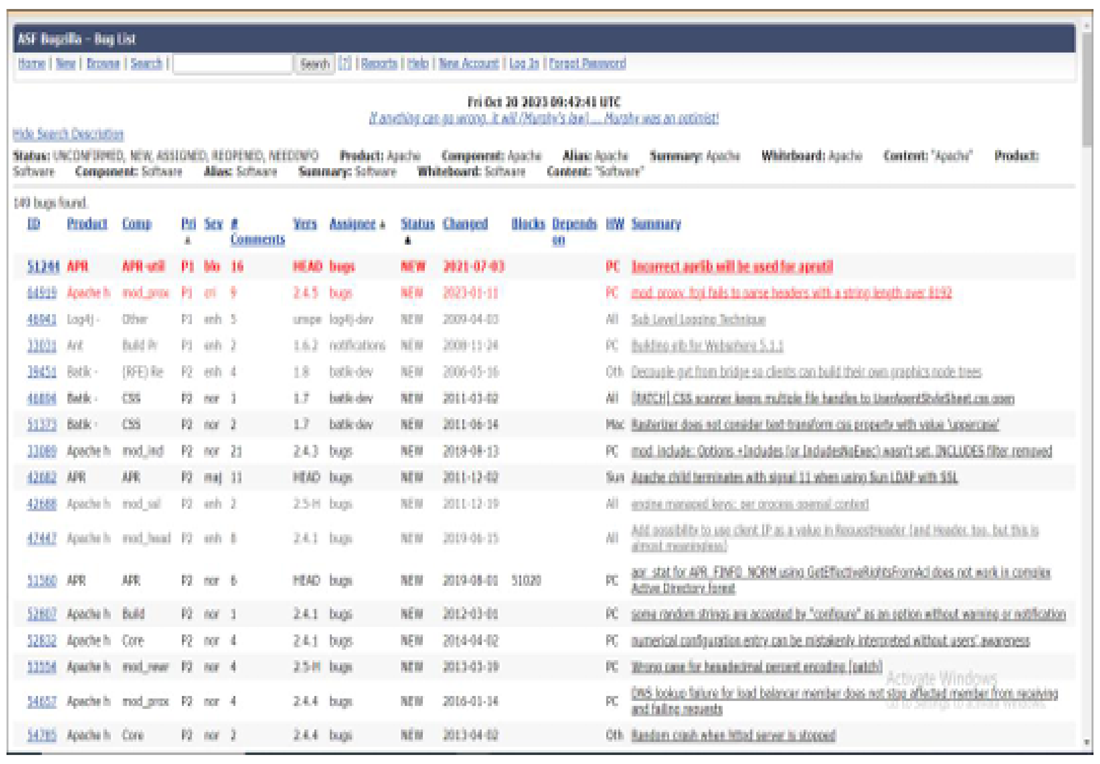

Fig 2: Datasets

LIBRARIES:

Seaborn and Matplotlib allow you to generate visual representations of data. The Pandas program is used to process and analyze data. These programs allow for effective plotting and graphing. Additionally, scikit-learn modules such as classification_report, label encoder, random forest classifier, and SVC are required for data preparation, model training, and success evaluation.

DATA PRE-PROCESSING AND FEATURE ENGINEERING:

This code is meant to make machine learning approaches more accessible for issue data analysis. The prefix "Unknown" before the "Assignee" element to indicate insufficient information. The information in the "Changed" fields is utilized to create datetime objects. The first elements of feature engineering are the fields "Changed_Year" and "Changed_Month," which are named after the "Changed" date and utilized to determine the year and month. The LabelEncoder allows machine learning algorithms to store category data such as product, component, assignee, status, version, depends on, hardware, summary, blocks, severity, and priority. Blocks, severity, and priority are all noted as well. At this stage, data is meticulously compiled to facilitate model testing and training.

SPLITTING THE DATA AND MODEL TRAINING:

Machine learning relies largely on code since it divides the dataset into training and testing sets and instructs Random Forest models on how to assess the importance and severity of issues. Using the Scikit-learn train_test_split method, the data is partitioned into features (X) and priority and severity target variables (y_severity and y_priority), respectively. Because of stratified divisions, severity and importance labels can be kept consistent between training and testing sets. Using the training set, we trained two Random Forest classifiers, each consisting of 100 decision trees. Classifiers learn about relationships and patterns within a dataset in order to predict the outcomes of future occurrences in the testing set. We now understand the basic components. These photographs and measurements are designed to aid future assessments of model success.

TRAINING MACHINE LEARNING MODELS:

The Random Forest Classifier and Support Vector Machines (SVM) are two popular methods in machine learning for determining the importance and severity of a problem. Specific training is performed on each algorithm in terms of severity and priority predictions. One hundred decision trees collectively safeguard the Random Forest Classifier against highly specialized tasks. Training the severity_model with the feature matrix X_train and the seriousness labels y_severity_train enables it to predict the gravity of a given circumstance. The feature matrix X_train and the priority labels y_priority_train for the priority model are both trained with the same set of data.

4. RESULTS AND DISCUSSIONS

To measure the severity of an issue, we employ both the SVM (svm_severity_model) and the Random Forest (severity_model) models. Forming category reports aims to evaluate the models' performance. The classification report thoroughly summarizes numerous parameters for each class, including support, accuracy, recall, and the F1-score. Two articles quantify the problem's severity and offer a solution hierarchy, with one addressing SVM models and the other Random Forest models. Using these measurements, one can find areas where models can enhance their accuracy in evaluating the significance and severity of problems. This method simplifies determining the models' functioning. The following table displays the severity model report. The classification report for each class includes a detailed summary of several aspects. Support, recall, F1 score, and precision are some of the measurements used. Different approaches of categorizing model problems can be enhanced and validated by employing metrics linked to problem severity and priority. This strategy aids comprehension of how the models work.

Table 1: Precision, along with Recall, along with F1- Score for various

models

RANDOM FOREST SEVERITY MODEL REPORT:				
CLASS	**PRECISION**	**RECALL**	**F1-SCORE**	**SUPPORT**
0	0.00	0.00	0.00	0
1	0.00	0.00	0.00	2
2	0.14	0.50	0.22	2
3	0.00	0.00	0.00	2
4	0.00	0.00	0.00	1
5	0.85	0.74	0.79	3
Accuracy			0.60	30
Macro avg	0.17	0.21	0.17	30
Weighted avg	0.66	0.60	0.62	30
RANDOM FOREST PRIORITY MODEL REPORT:				
CLASS	**PRECISION**	**RECALL**	**F1-SCORE**	**SUPPORT**
0	0.00	0.00	0.00	1
1	0.86	1.00	0.92	24
2	1.00	0.50	0.67	4
4	0.00	0.00	0.00	1
Accuracy			0.87	30
Macro avg	0.46	0.38	0.40	30
Weighted avg	0.82	0.87	0.83	30
SVM SEVERITY MODEL REPORT:				
CLASS	**PRECISION**	**RECALL**	**F1-SCORE**	**SUPPORT**
1	0.00	0.00	0.00	2
2	0.14	0.50	0.22	2
3	0.00	0.00	0.00	2
4	0.00	0.00	0.00	1
5	0.86	0.78	0.82	23
Accuracy			0.63	30
Macro avg	0.20	0.26	0.21	30
Weighted avg	0.67	0.63	0.64	30
SVM PRIORITY MODEL REPORT:				
CLASS	**PRECISION**	**RECALL**	**F1-SCORE**	**SUPPORT**
0	0.00	0.00	0.00	
1	0.92	0.96	0.94	
2	0.60	0.75	0.67	
4	0.00	0.00	0.00	
Accuracy			0.87	30
Macro avg	0.38	0.43	0.40	30
Weighted avg	0.82	0.87	0.84	30

The Random Forest Severity Model's accuracy, recall, and F1-score metrics vary per severity category. The model's accuracy for severity class 5 is impressive, as indicated by an F1-score of 0.79. The overall accuracy is 0.60, although precision and memory use are significantly lower for Class 0 and other severity levels. The Random Forest Priority Model is an excellent choice since it has more stable distributions of precision, recall, and F1-score, as well as a high accuracy rate of 0.87. The model estimates the impact of a class 1 event rather easily, while class 0 and class 2 events provide a more difficult problem. The reports from the Priority and Severity SVM models show analogous trends. The models' forecasting skills are confined to categories 5 and 1 in terms of severity and relevance. Despite this, the models perform best in these courses. The image can be used to represent the distribution of severity.

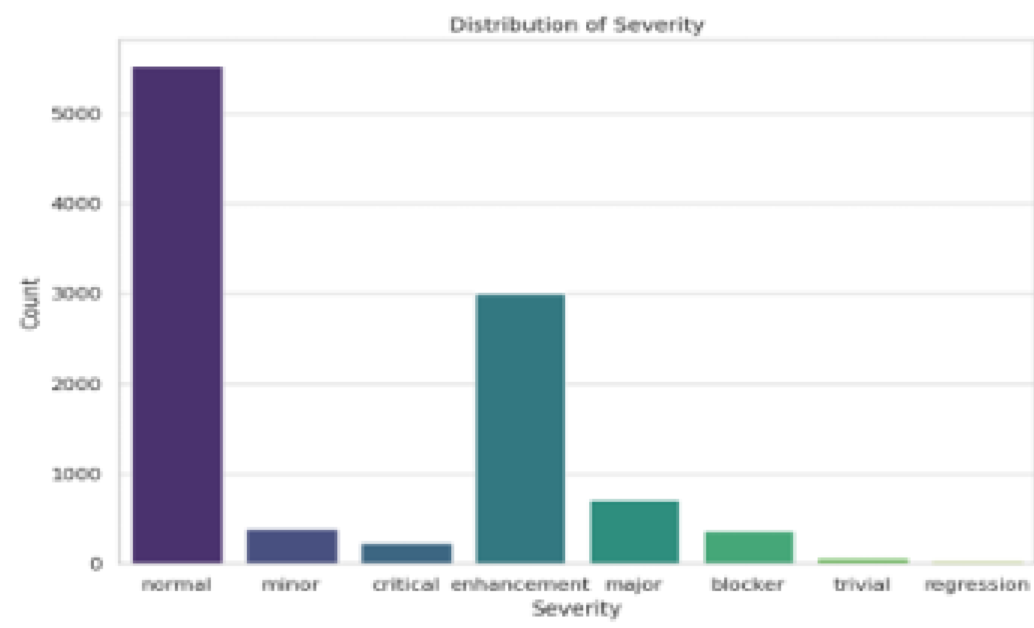

Fig 3: Distribution of Severity

The following graphic demonstrates how problem objectives are resolved.

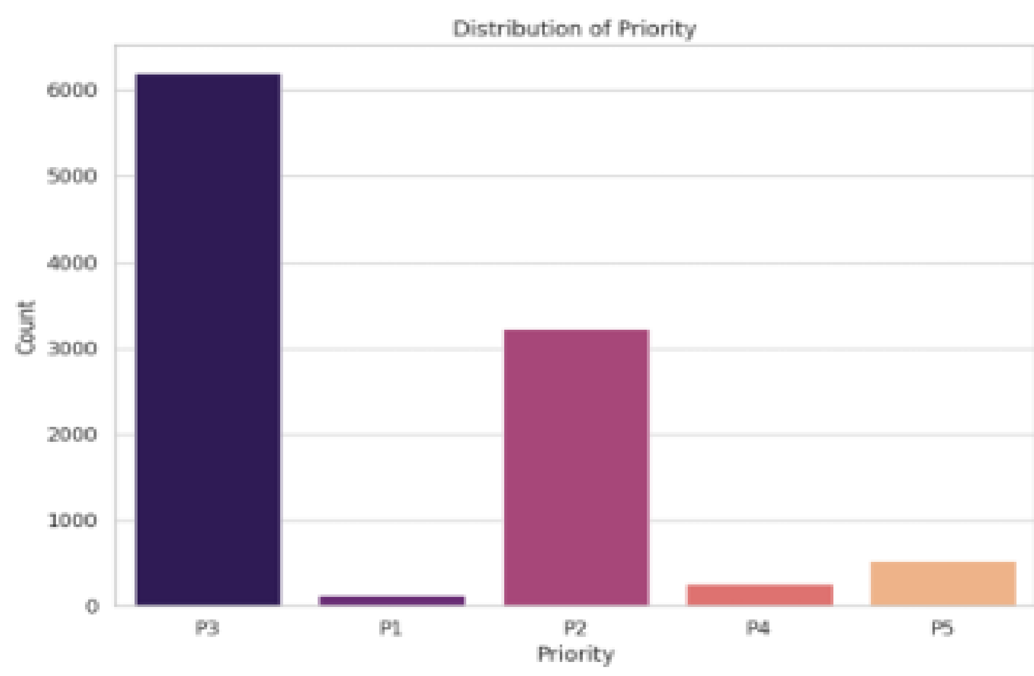

Fig 4: Distribution of priority

The current graph depicts the severity of the fault based on our findings.

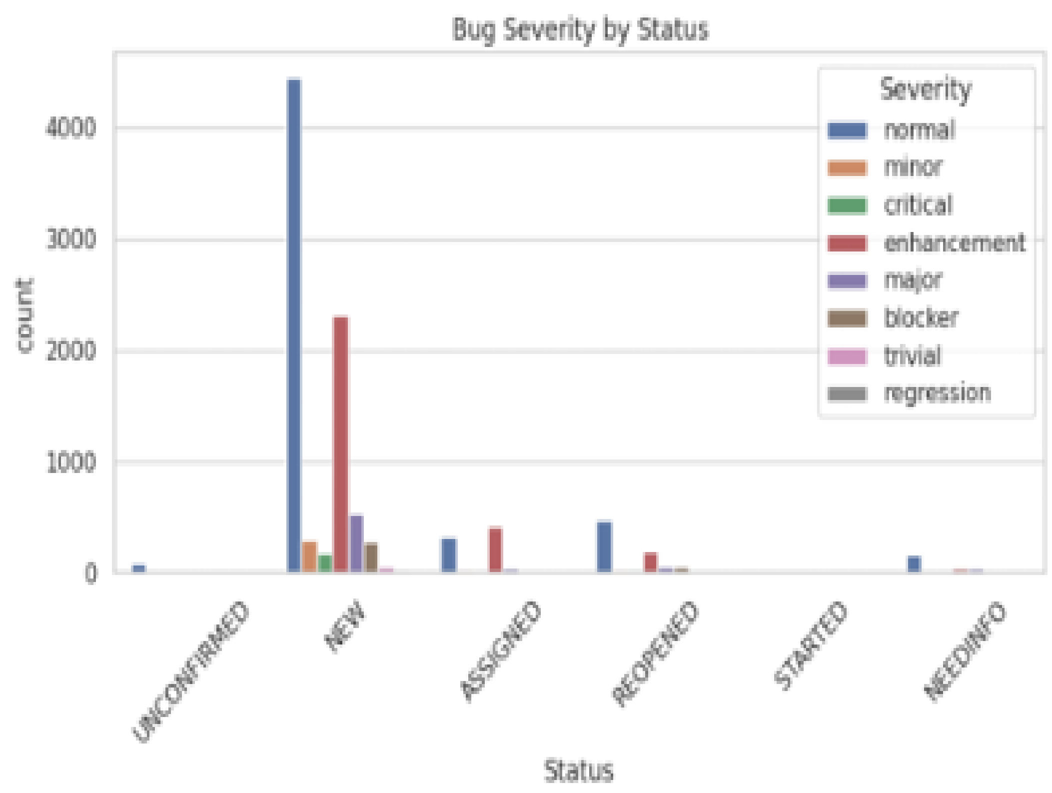

Fig 5: Bug Severity by Status

The graph shows the severity and importance rankings and distributions.

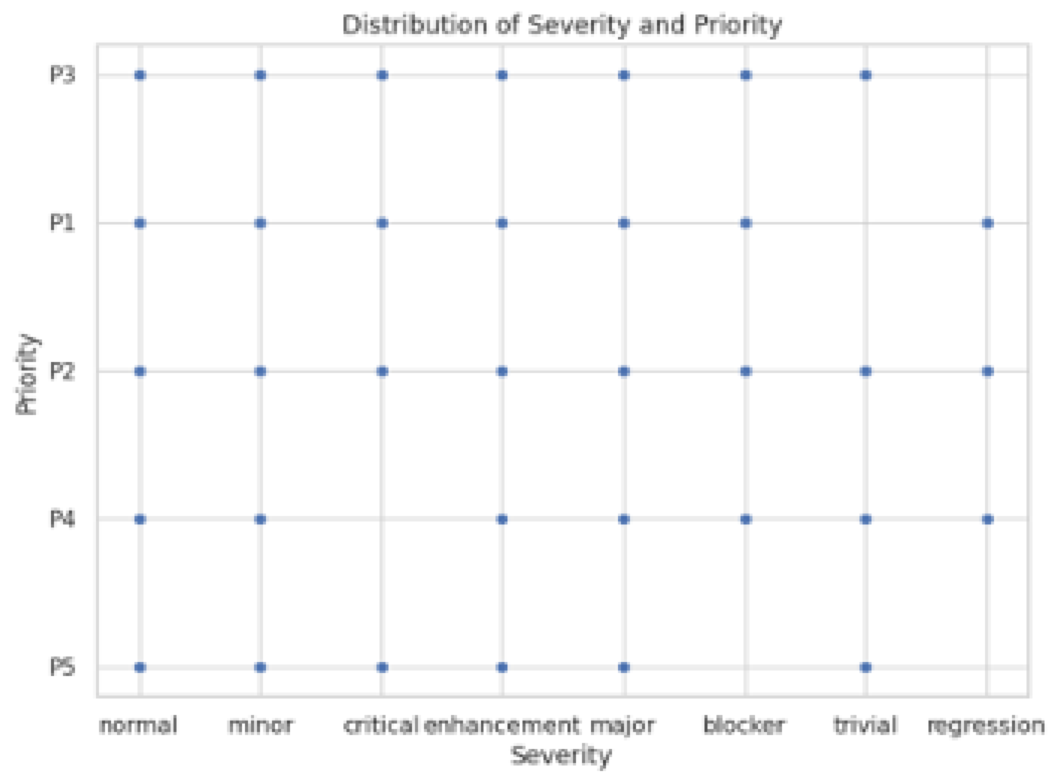

Fig 6: Distribution of Severity and Priority

The histogram displaying severity levels depicts the distribution of severity levels within the group. The magnitude of a problem is represented by its quantity, which ranges from zero to seven. The x-axis of this histogram reflects intensity levels, and the y-axis represents the number of issues at each intensity level. The histogram clearly shows that concerns with moderate to severe repercussions are most typically noted at severity levels 2 and 5. A comparatively modest proportion of instances fell into the severity 0-7 category. The prioritizing distribution is represented by a histogram of priority levels with values ranging from 0 to 4. Each issue is assigned an urgency number to indicate how important its resolution is. In contrast to the low frequency of levels 0, 2, and 3, the high frequency of priority level 1 on the graph suggests that a large number of situations deserve immediate attention. A histogram of intensity levels is shown below.

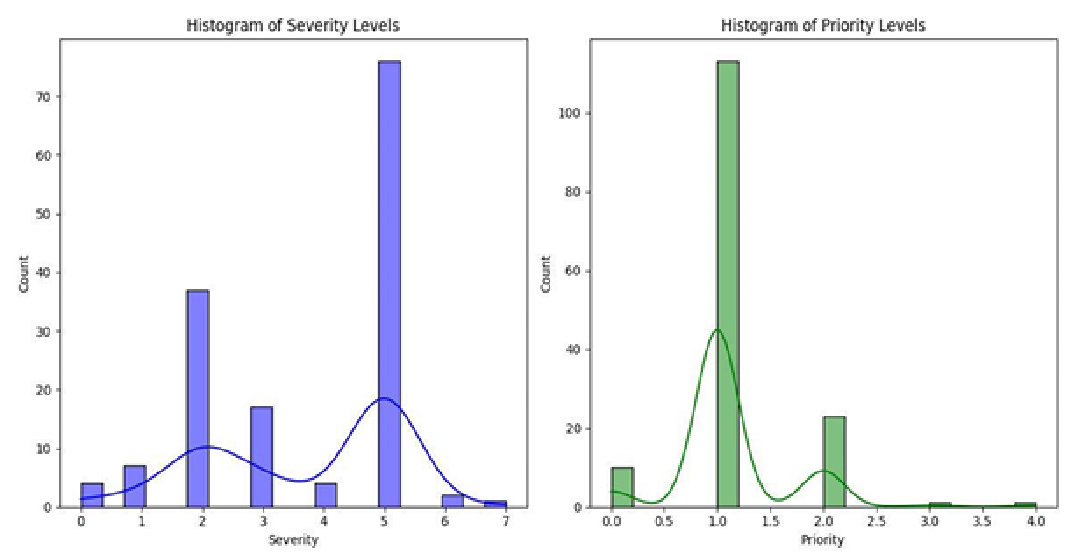

Fig 7: Histogram of Severity and Priority levels.

5. CONCLUSION

This research aims to generate educated hypotheses about the nature, severity, and impact of software development tool difficulties. In the Random Forest Severity Model, the severity class determines the frequency, accuracy, and F1 score. An F1 score of 0.79 for severity class 5 indicates that the model is accurate. Class 0 and other severity groups have much lower memory and precision, despite a 0.60 accuracy rate. With more constant precision, recall, and F1-score, as well as a robust accuracy of 0.87, the Random Forest Priority Model outperforms the others. The histogram depicts the cohort's severity-degree frequency distribution. The importance rating of a problem indicates its level of severity on a scale of zero to seven. In the graph, the y-axis represents the severity levels, and the x-axis indicates the number of issues at each level. The histogram depicts the cohort's severity-degree frequency distribution. Concerns are graded on a scale of zero to seven, from least to most significant. The x-axis of the histogram shows the number of worries sorted by severity. The severity level of each issue is shown on the y-axis. The Random Forest Priority Model graphic visualizes the model's planned behavior at different priority levels. According to the paper, a random forest model could be an effective way to integrate predictive analysis into SDKs. The goal of this approach is to identify the types, severity, and importance of software issues. This work is an important resource for future research because it offers academics with a straightforward way for enhancing the quality of studies that seek to quantify the severity and impact of software problems.

6. REFERENCES

1. Ali, S., Baseer, S., Abbasi, I. A., Alouffi, B., Alosaimi, W., & Huang, J. (2022). Analyzing the interactions among factors affecting cloud adoption for software testing: a two-stage ISM-ANN approach. Soft Computing, 26(16), 8047-8075
2. Shatnawi, M. Q., & Alazzam, B. (2022). An Assessment of Eclipse Bugs' Priority and Severity Prediction Using Machine Learning. International Journal of Communication Networks and Information Security, 14(1), 62-69.
3. Sathish Polu and Dr. V. Bapuji. “Analysis of DDOS Attack Detection in Cloud Computing Using Machine Learning Algorithm”, Tuijin Jishu/Journal of Propulsion Technology, Vol. 44, No.5, Pages:2410-2418, ISSN:1001-4055, December2023.
4. Akmel, F., Birihanu, E., & Siraj, B. (2017). A literature review paper of software defect prediction using machine learning techniques. Int. J. Emerg. Res. Manag. Technol, 6(6), 300-306.
5. Ali, S., Ullah, N., Abrar, M. F., Majeed, M. F., Umar, M. A., & Huang, J. (2019). Barriers to software outsourcing partnership formation: an exploratory analysis. IEEE Access, 7, 164556-164594.
6. Ali, S., Nasir, A., Samad, A., Basser, S., & Irshad, A. (2022). An

automated approach for the prediction of the severity level of bug reports using GPT 2. Security and Communication Networks, 2022.

7. Boddupalli Anvesh Kumar, Dr. V. Bapuji, " EFFICIENT AND PRIVACY -PRESERVING MULTI-FACTOR DEVICE AUTHENTICATION PROTOCOL FOR IOT" International Journal of Innovative Research in Technology,(IJIRT), Volume 9, Issue7, ISSN:2349-6002. December 2022, (UGC CARE LIST -1).
8. Olaleye, T. O., Arogundade, O. T., Abayomi-Alli, A., & Adesemowo, A. K. (2021). An ensemble predictive analytics of COVID-19 infodemic tweets using bag of words. In Data science for COVID 19 (pp. 365-380). Academic Press.
9. Sathish Polu and Dr. V. Bapuji, "Distributed Denial of Service (DDOS) Attack Detection in Cloud Environments Using Machine Learning Algorithms", International Journal of Innovative Research in Technology, (IJIRT), Volume 9, Issue7, ISSN:2349-6002.December 2022, (UGC CARE LIST – I).
10. Boddupalli Anvesh Kumar, Dr. V. Bapuji," Secure And Lightweight Authentication Protocols for Devices in internet of Things", Tuijin Jishu/Journal of Propulsion Technology, Vol. 44, NO.,Pages:2419-2427,ISSN:1001-4055,December 2023.
11. 5Zhang, Y., Chen, Y., Cheung, S. C., Xiong, Y., & Zhang, L. (2018, July). An empirical paper on TensorFlow program bugs. In Proceedings of the 27th ACM SIGSOFT international symposium on software testing and analysis (pp. 129-140).
12. Malgonde, O., & Chari, K. (2019). An ensemble based model for predicting agile software development effort. Empirical Software Engineering, 24, 1017-1055.
13. Shetty, J., Babu, B. S., & Shobha, G. (2020). Proactive cloud service assurance framework for fault remediation in cloud environment. International Journal of Electrical & Computer Engineering (2088-8708), 10(1).
14. Boddupalli Anvesh Kumar, Dr. V. Bapuji, "Efficient Privacy Preserving Communication Protocol For IoT Applications", The Brazilian Journal of Development ISSN 2525-8761, publishes by Brazilian Journals and Publishing LTDA. (CNPJ 32.432.868/0001-57) Vol.No.10, Pages:402-419 January 2024.

15. Sathish Polu and Dr. V. Bapuji," "Mitigating Ddos Attacks in Cloud Computing Using Machine Learning Algorithms", The Brazilian Journal of Development ISSN 2525-8761, published by Brazilian Journals and Publishing LTDA. (CNPJ 32.432.868/0001-57) Vol.No.10, Pages:340-354January2024. https://ojs.brazilianjournals.com.br/ojs/index.php/BRJD/article/view/66109
16. Ahmed, H. A., Bawany, N. Z., & Shamsi, J. A. (2021). Capbug-a framework for automatic bug categorization and prioritization using nlp and machine learning algorithms. IEEE Access, 9, 50496-50512.