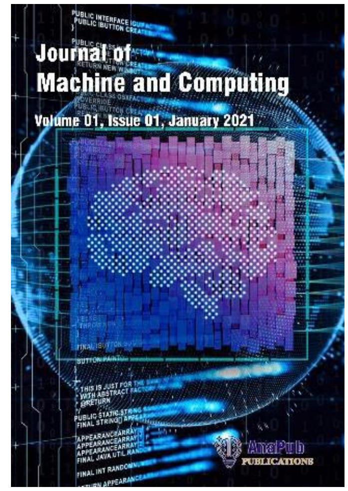
Journal Pre-proof

A Robust Evaluation of Bug Pre-Processing and Classification Logic using NLP Computation with Machine Learning Technique



Mamatha Racharla, Lalitha Surya Kumari P and Sharada Adepu

DOI: 10.53759/7669/jmc202505172

Reference: JMC202505172

Journal: Journal of Machine and Computing.

Received 25 April 2025

Revised from 02 July 2025

Accepted 28 July 2025

Please cite this article as: Mamatha Racharla, Lalitha Surya Kumari P and Sharada Adepu, " A Robust Evaluation of Bug Pre-Processing and Classification Logic using NLP Computation with Machine Learning Technique", Journal of Machine and Computing. (2025). Doi: https:// doi.org/10.53759/7669/jmc202505172.

This PDF file contains an article that has undergone certain improvements after acceptance. These enhancements include the addition of a cover page, metadata, and formatting changes aimed at enhancing readability. However, it is important to note that this version is not considered the final authoritative version of the article.

Prior to its official publication, this version will undergo further stages of refinement, such as copyediting, typesetting, and comprehensive review. These processes are implemented to ensure the article's final form is of the highest quality. The purpose of sharing this version is to offer early visibility of the article's content to readers.

Please be aware that throughout the production process, it is possible that errors or discrepancies may be identified, which could impact the content. Additionally, all legal disclaimers applicable to the journal remain in effect.

© 2025 Published by AnaPub Publications.

Authors Pre-Proof

A Robust Evaluation of Bug Pre-Processing and Classification Logic using NLP Computation with Machine Learning Technique

[1]Mamatha Racharla*, [2]P. Lalitha Surya Kumari, [3]Sharada Adepu

[1]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Hyderabad-500075, Telangana.
mamatha.racharla@klh.edu.in

[2]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Hyderabad-500075, Telangana.
vlalithanagesh@gmail.com

[3]Department of Computer Science and Engineering, G. Narayanamma Institute of Technology & Science, Hyderabad.
sharada.nirmal@gmail.com

***Corresponding Author:** Mamatha Racharla

Abstract-Enhancing the software maintenance greatly depends on the precise and prompt handing out of bug reports according to their bug-category and importance. To resolve the aforementioned problems, an automated method of classifying and ranking bug reports is required. Numerous scholars have recently looked into the automated classification and prioritization of bug reports. But not much has been accomplished in this area. During software development, the most crucial stages are testing and maintenance. In these phases of development activity, bug reports are essential. When software modules are being tested, the software quality assurance team creates a bug report. But the main issue that comes up while analysing bug data that is written in normal text. As a result, processing and extracting information from it is extremely challenging. The aforementioned requirements are the driving force for this research. The Proposed research suggested creating a hybrid model that takes advantage of machine learning models' contextual awareness as well as more conventional feature extraction methods (such as TF-IDF). A downstream classifier (such as an SVM, logistic regression) can receive these two feature sets (one from TF-IDF and the other from BERT) after they have been concatenated. This enables the model to take advantage of the extensive contextual relationships that BERT captures as well as the statistical importance of phrases (TF-IDF) These two approaches were used separately in the earlier research, which resulted in less performance. The research made use of a confidential dataset that was acquired from a private company upon request for performing testing, the data included from eight hundred employees. To aid in model training, bug keywords were first taken out of the bug description field. The results shows that proposed model achieves 89% accuracy.

Keywords: Term Frequency, Bug Reports, Inverse Document Frequency, NLP, Machine Learning.

I. INTRODUCTION

A bug report contains a variety of information, including requests for features, requests for functionality enhancements, code faults, logical flaws, and compatibility problems. Priority, summary, affected component description, and open/close status are among the headings that make up a report [2]. But the main issue that comes up when analyzing bug reports is that data which is written in normal language. As a result, processing and extracting information from it is extremely challenging. The development team has to put in a lot of work to comprehend and fix the issues that have been reported. There are numerous research that deal with the problems pertaining to bug reports [4]–[9]. Bug classification [2], [20], [21], bug severity prediction [15],[16] bug assignment, bug localization, bug priority bug categorization and bug report summarizing are a few of these. The most crucial information needed from each bug report is still the classification and prioritizing of bugs. The information extraction technique was automated in the majority of the investigations using supervised machine learning algorithms . By using these techniques, a manually labelled dataset of bug reports is trained to create a classification model that is subsequently utilised to automatically prioritize and classify new issues using pre-defined labels. Large labelled datasets are necessary for supervised learning approaches, but they are not readily accessible. The majority of the datasets that are currently available lack category and priority information. Moreover, the majority of research that is now accessible focuses on a single issue alone, such as automating bug prioritization [11] or bug classification . As a result, the field of concurrently classifying and prioritizing bug reports has seen relatively little activity . Consequently, a strong framework that can simultaneously automate defect prioritization and bug classification is required.

II . LITERATURE SURVEY

This study proposes by Kajal Tameswar at., al. (2023) suggested a hybrid models of K-Means (KM) and numerous Nature-inspired algorithms (NIAs) to identify the initial values and cluster center of KM. The combination of KM and NIAs outperforms the typical stand-alone K-means in terms of prediction accuracy. On average, the researchers achieved an improved accuracy of 7% with KM and coral reefs algorithms when compared to typical k means. NASA datasets were used for the tests, and assessment criteria such as accuracy, F1 score,and computation time were used to compare performance.

The performance of various classifier algorithms, such as Naive Bayes, Support Vector Machines (SVM), K-Nearest Neighbors, Artificial Neural Networks (ANN), and Decision Trees (DT), was compared in the comparative analysis conducted by Bansal, Malti et al. (2022). Decision Tree and Random Forest were chosen as classifiers for the trials, and they were used on datasets from bug tracking systems that is open-source like Mantis, Debian, Launchpad, , and Bugzilla. The suggested approach separated severity into critical, normal, and minor categories and included four priority classes: urgent, high, normal, and low. With an accuracy of 0.75, or average, Random Forest outperformed DT when the classifiers' performance evaluated using time consumption, Median-Absolute Error (MAE) and Mean-Squared Error (MSE)

ABA-TF-IDF, a novel time-based bug localization technique that utilizes the Time TF-IDF weighing mechanism, is presented in the publication by R. Shakripur et al. (2021). In industry a Version Control System (VCS) software repository, which keeps track of project specifics and source code changes, provided the information. 4 machine learning algorithms— Vector Space Model (VSM), Naive Bayes, Support Vector Machine (SVM), and Smooth Unigram Model (SUM)—were used to train the model and evaluate its performance in the tests. Using VCS data and temporal bug information with TF-IDF to improve bug localization is the main breakthrough. The study's findings reveal notable gains in Mean Reciprocal Rank (MRR), indicating the efficacy of the suggested ABA-TF-IDF method.
In particular, the MRR rose by as much as 11.8%, demonstrating the superior performance of the suggested technique over conventional bug localization techniques.

The researchers developed a brand-new method for classifying bugs called Bug Named-Entity Recognition (BNER). 3 key elements were included in this novel approach: the (POS)- Parts of Speech of entities in the bug reports, description phrases, and a strong distribution. Based on these elements, a classification system was created to group bugs into a preset sixteen categories. A Semi-Supervised BNER system and the creation of a baseline corpus with complete information form the basis of this method, which was put forth by C. Zhou et al. (2018). The feature extraction process was made easier by a method built into the bug repository. Data from Eclipse and Mozilla, two

Authors Pre-Proof

online software bug repositories, were used to train and assess the suggested method. The results illustrated the efficacy of their methods, emphasizing the importance of establishing a baseline corpus during the preliminary phases. The semi-supervised BNER system markedly enhanced accuracy, with a considerable improvement of 70% to 80%. The study emphasized the potential efficacy of BNER in identifying entities across several projects, indicating its relevance beyond singular repositories. This highlights the adaptability and functional efficacy of the insect Named-Entity Recognition method in improving insect classification precision and efficiency.

In their study, Goseva et al. (2018) classified non-security and security bugs in a dataset from the national aeronautics and space administration (NASA) using both supervised and unsupervised methods. For both strategies, two feature vector techniques were used: Bag of Word Frequency, Term Frequency (TF), and Term Frequency - Inverse Document Frequency (TF-IDF). The supervised approach used several classifiers, such as Naïve Bayes Multinomial (NBM), k-Nearest Neighbours (K-NN), Support Vector Machine (SVM), Naïve Bayes (NB), and Bayesian Network (BN), whereas the unsupervised approach used an anomaly detection technique. According to the results, the supervised method fared better at classifying bugs than the unsupervised method. This implies that using different classifiers and labeled data in a supervised environment produced better outcomes than the unsupervised approach, demonstrating the value of guided learning in the classification of bugs into security and non-security within the NASA dataset.

III. METHODOLOGY PROPOSED

The proposed method addresses tracking data for a software project. The information in the project's issue description is provided in standard text format and outlines the issue or problem that must be fixed by the developer. Such statistics make it extremely difficult to extract pertinent information. It is extremely challenging to adequately extract all of the necessary data from the description field. However, one might begin with easy steps and at the very least extract what is possible from some of the well-known forms. NLP is a subfield of AI that enables systems to understand processes and derive some meaning from human natural language. First, the machine will be fed with enough data so that it can learn from experience, then the machine will create the word vector, and finally, by performing simple algebraic expressions on the obtained word vector, a machine can provide answers as a human does. Lemmatization and stemming techniques are used in the bug classification process to standardize the linguistic data found in bug reports. Then, using this standardized data, a vocabulary designed especially for software defects is created. Next, the importance of each keyword is evaluated by calculating its tf and idf. After generating a vocabulary and ranking keywords, ML techniques are used to categorize newly received software bug reports. By using the enriched data, these algorithms are able to anticipate and classify issues into many groups, including network, GUI, software, and performance bugs. This classification process lays the groundwork for later research stages, allowing for more focused analysis and approaches to bug fixation for different kinds of bugs. log-based Bug Prioritization. The framework is shown in Fig.3.1.

Authors Pre-Proof

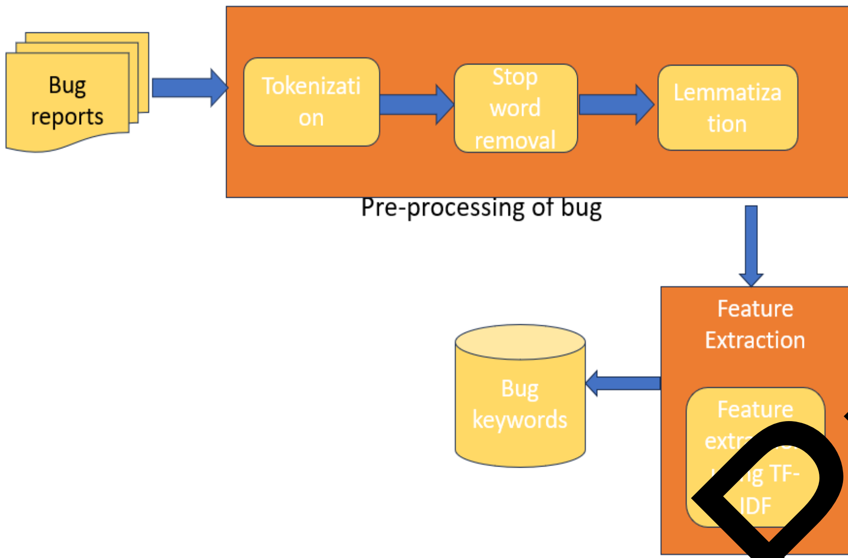


Fig 3.1: Proposed Framework for pre-processing bug reports

This option helps remove stop words unique to a corpus by defining the threshold for discarding phrases whose document frequency is greater than the specified number. Maximum features: It establishes the maximum number of columns that can be included in the final matrix n-gram range: This includes single words, bi-grams, and tri-grams. It specifies the range of n-grams to take into account.An encoded vector with a length that matches the vocabulary is the output.

Now that the summary field has been tokenized and the most relevant keywords relevant to projects have been extracted, it is given as input to a classification algorithm to classify the bug severity. Now that the model can classify the bugs entered if we can determine the relationship between employee skill set and bug-solving capacity from historical data, the trigger can efficiently assign the right person to the right task. The model was trained using the Decision Tree algorithm, which provided better performance metrics than the other algorithms. From a confusion matrix method, the performance of a classification algorithm is summarized. Because there are more than two classes in the dataset and each class has an unequal number of observations, classification accuracy alone may be deceiving. After generating a vocabulary and ranking keywords, ML models are used to categorize newly received software reports on bugs. By using the enriched data, these algorithms are able to anticipate and classify issues into many groups, including network, GUI, software, and performance bugs. This classification process lays the groundwork for later research stages, allowing for more focused analysis and approaches to bug fixation for different kinds of bugs. log-based Bug Prioritization.

IV. RESULTS AND DISCUSION

In the first stage of the investigation, the extraction of relevant bug keywords from the bug description field using data preprocessing techniques was achieved. natural language processing (NLP) and Text mining approaches were used to identify and separate key terms and phrases that were indicative of bug traits and attributes. To get the dataset ready for additional model training and analysis, important attributes were taken from the bug descriptions. As the study moved into its second phase, the investigation's focus shifted to applying machine learning techniques to assess the problems' severity. With the use of machine learning techniques, the study aimed to find recurrent patterns in the historical problem resolution data that could direct the estimation of the defect severity based on the bug presented by the

Authors Pre-Proof

program user. In the field of bug prediction, ensemble methods like XGBoost and CatBoost provide a number of advantages over conventional machine learning algorithms. Initially, these algorithms are highly effective in managing intricate relationships and non-linearities in the data, offering a more precise depiction of the complex patterns seen in software defect incidents.

Moreover, robust and resilient predictive model are created with ensemble methods where it combines multiple base models to create a. F1 Score, Recall, Accuracy, Precision collectively measures a classification model's performance by measuring overall correctness, the ability to avoid false positives, completeness in capturing true positives, and a balanced view of Precision and Recall, making them essential for evaluating predictions, especially in cases with imbalanced data. In Fig.4.2 the outcomes of pre-processing the summary field and feeding it to the model as training data is shown. For classification, the Random Forest method was utilized.

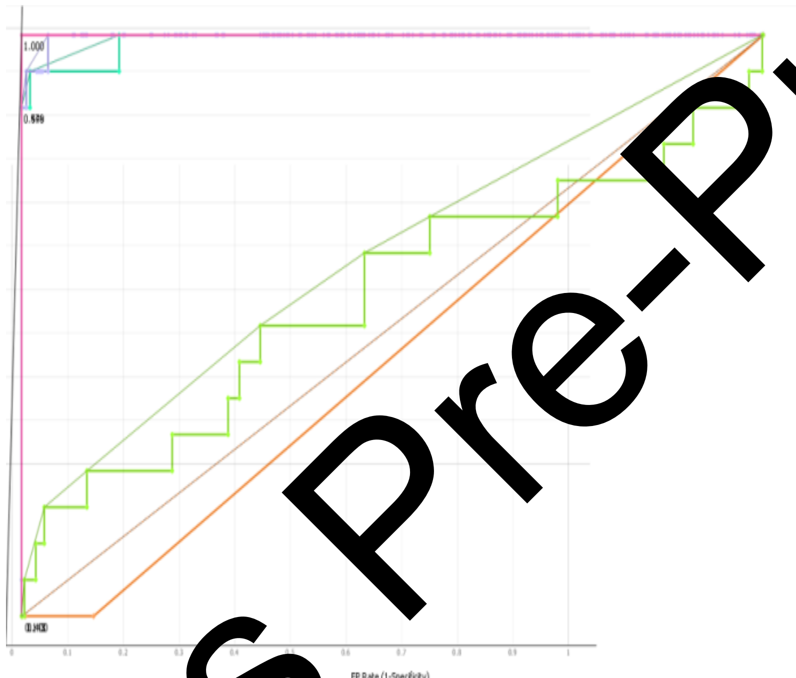


Fig 4.1: Enhanced ROC with Ensemble Algorithms

Authors Pre-Proof

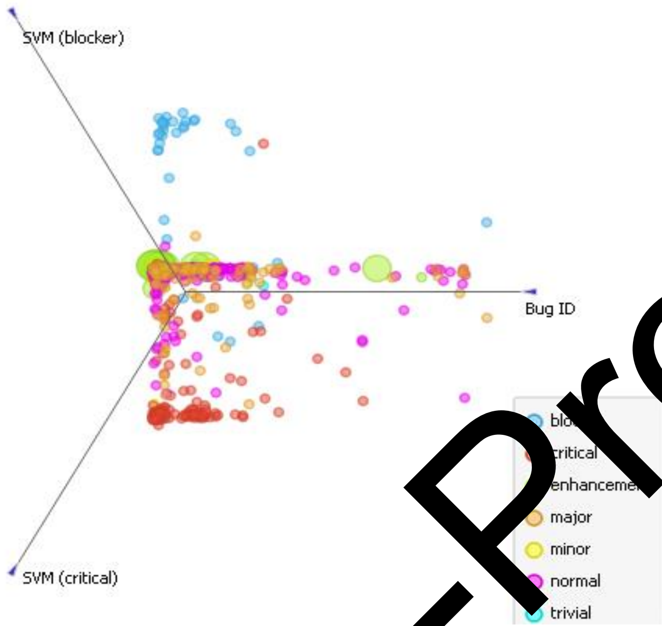


Fig 4.2: Linear projection with Ensemble algorithms

The results of the ML model's classification of the different bugs into major, minor, critical, blocker, and trivial bugs are displayed in Fig. 4.3. From the above results, using Random Forest the test accuracy obtained is about 76% and the trainingaccuracyis89%respectively.

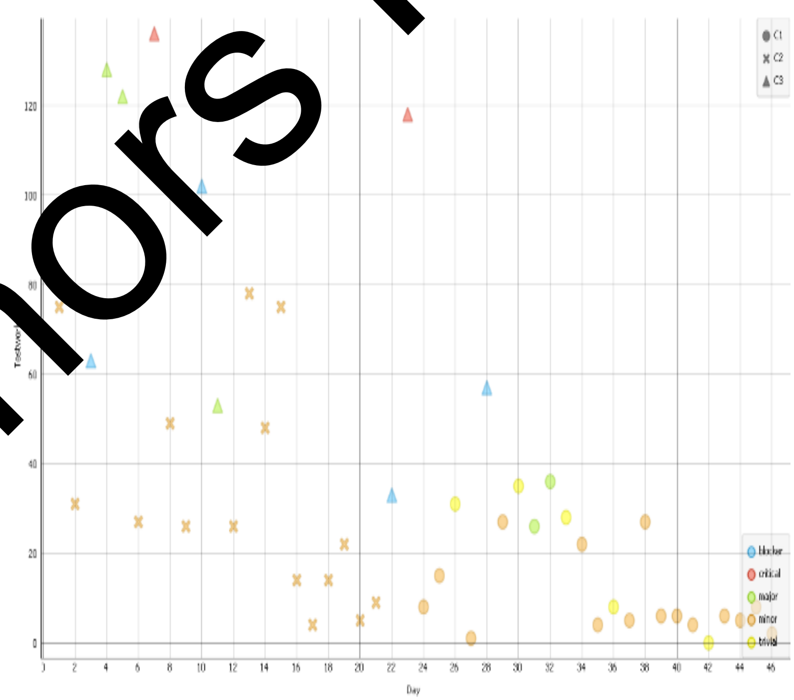


Fig 4.3: Model classifying the bugs into different categories

Authors Pre-Proof

Fig 4.4 shows the decision tree model for different bug types. To enhances the accuracy of the model, the researchers can also experiment with some more sophisticate deep-learning techniques.

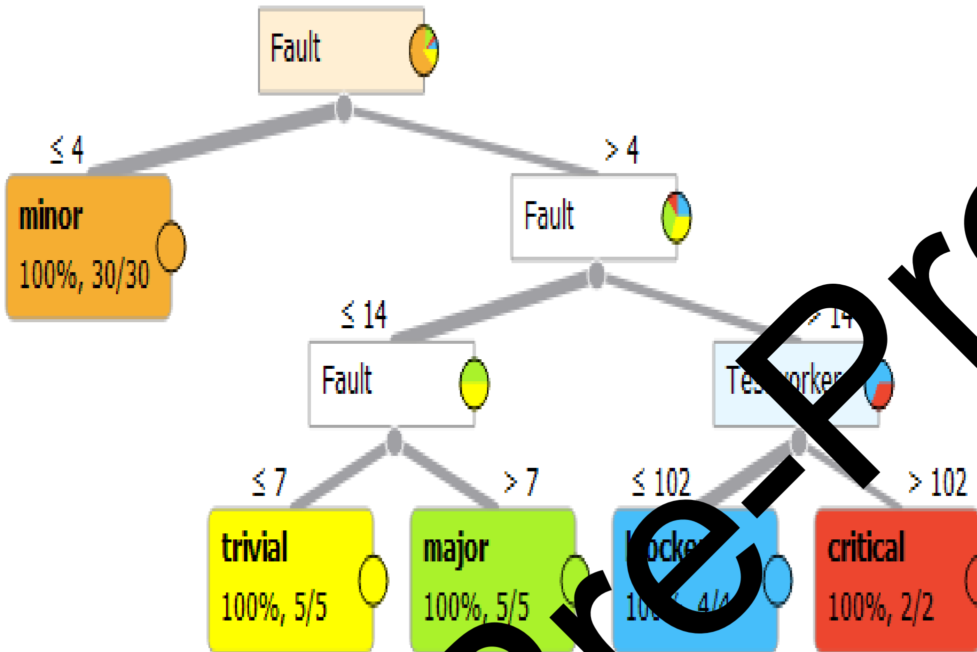


Fig 4.4: Decision Tree showing the bug type

V. CONCLUSION AND FUTURE WORK

Bug reporting is critical to software maintenance and development. They let s/w developers, QA teams, and customers to detect and report relevant issues. Due to the size of these reports, manual extraction is time-consuming and inefficient. Categorization and prioritization need the use of an automated technique. This research aims to automatically categories and priorities bug reports. Natural language processing has greatly improved software development by collecting bug keys from description fields. Deep learning techniques can be used to improve accuracy in this research.

REFERENCES

[1] Messaoudi, D. and Nessah, D. 2024. Enhancing Neural Arabic Machine Translation using Character-Level CNN-BILSTM and Hybrid Attention. Engineering, Technology & Applied Science Research. 14, 5 (Oct. 2024),17029–17034. DOI:https://doi.org/10.48084/etasr.8383.

[2] G. Catolino, F. Palomba, A. Zaidman, and F. Ferrucci, “Not all bugs are the same: Understanding, characterizing, and classifying bug types,” J. Syst. Softw., vol. 152, pp. 165–181, Jun. 2019.

[3] Yang, Wenyu, et al. "RepoLike: amulti-feature-based personalized recommendation approach for open-source repositories." *Frontiers of Information Technology & Electronic Engineering* 20.2 (2019): 222-237.

[4] Goseva-Popstojanova, Katerina, and Jacob Tyo. "Identification of security related bug reports via text mining using supervised and unsupervised classification." 2018 IEEE International conference on software quality, reliability and security (QRS). IEEE, 2018.[8] R. Shakripour, J. Anvik, Z. M. Kasirun, and S. Zamani, “A time-based approach to automatic bug report assignment,” J. Syst. Softw., vol. 102, pp. 109–122, Apr. 2015, doi: 10.1016/j.jss.2014.12.049.

[5] R. Lotufo, Z. Malik, and K. Czarnecki, “Modelling the ‘hurried’ bug report reading process to summarize bug reports,” Empirical Softw. Eng., vol. 20, no. 2, pp. 516–548, 2015.

Authors Pre-Proof

[6] C.-Z. Yang, C.-M. Ao, and Y.-H. Chung, ‘‘Towards an improvement of bug report summarization using two-layer semantic information,’’ IEICE Trans. Inf. Syst., vol. 101, no. 7, pp. 1743–1750, 2018.

[7] Chang, C.K., M.J. Christensen, and T. Zhang, Genetic algorithms for project management. Annals of Software Engineering, 2001. 11(1): p. 107-139.

[8] Izadi, M., Ganji, S., Heydarnoori, A., Gousios, G.: Topic recommendation for software repositories using multi-label classification algorithms (2020)

[9] X. Li, H. Jiang, D. Liu, Z. Ren, and G. Li, ‘‘Unsupervised deep bug report summarization,’’ in Proc. IEEE/ACM 26th Int. Conf. Program Comprehension CPC), May/Jun. 2018, pp. 144–14411. [13] H. Jiang, X. Li, Z. Ren, J. Xuan, and Z. Jin, ‘‘Toward better summarizing bug reports with crowdsourcing elicited attri ,’’ IEEE Trans. Rel., vol. 68, no. 1, pp. 2–22, Mar. 2019.

[10] Z. Ge, Z. Song, S. X. Ding, and B. Huang, ‘‘Data mining and analytics in the process industry: The role of machine learning,’’ IEEE Access, vol. 5 590–20616, 2017, doi: 10.1109/ACCESS.2017.2756872.

[11] M. Kumari and V. B. Singh, ‘‘An improved classifier based on entropy and deep learning for bug priority prediction,’’ in Proc. 18th Int. Conf. Int Syst. Desi Appl. (ISDA), 2018, pp. 571–580.

[12] W. Zou, D. Lo, Z. Chen, X. Xia, Y. Feng, and B. Xu, ‘‘How practitioners perceive automated bug report management techniques,’’ I E Trans. S w. Eng., vol. 46, no. 8, pp. 836–862, Aug. 2020.

[13] Bansal, Malti, Apoorva Goyal, and Apoorva Choudhary. "A comparative analysis of K-nearest neighbor, genetic, support v machine, n tree, and long short term memory algorithms in machine learning." Decision Analytics Journal 3 (2022): 100071.

[14] Yong Liu 1 , Xuexin Qi 1 , Jiali Zhang 1 , Hui Li 1,* , Xin Ge 1, Automatic Bug Triaging via Deep Reinfo ment Lear g and Jun 2 Appl. Sci. 2022, 12, 3565. https://doi.org/10.3390/app12073565.

[15] Kukkar, Ashima, et al. "Bug severity classification in software using ant colony optimization-based fea weighting technique." Expert Systems with Applications 230 (2023): 120573.

[16] H. Jantan, R. Hamdan, and Z. Othman, “Intelligent techniques for decision support system in human resour anagement,” Decision Support Systems, vol. 2, no. 10, pp. 261–276, 2020.

[17] Balogun, A.O., Basri, S., Abdulkadir, S.J., Adeyemo, V.E., Imam, A.A., Bajeh, A.O.: Sof are d ct diction: analysis of class imbalance and performance stability. J. Eng. Sci. Technol. 14, 3294–3308 (2019)

[18] Tan, Youshuai, et al. "Bug severity prediction using question-and-answer irs fro ck overfl ournal of Systems and Software 165 (2020): 110567.

[19] Laradji, I.H., Alshayeb, M., Ghouti, L.: Software defect prediction g ense le lea g on selected features. Inf. Softw. Technol. 58, 388–402 (2015)

[20] Ahmed, B., Ali, G., Hussain, A., Baseer, A. and Ahmed, J. 2021 alysis of T Feature E ctors using Deep Learning on Fake News. Engineering, Technology & Applied Science Research. 11, 2 (Apr. 2021), 7001–7005. DOI: s://d rg/10.48084/etasr.4069.

[21] Elangovan, D. and Subedha, V. 2023. Adaptive Particle Grey lf Optimizer with Deep Learning-based Sentiment Analysis on Online Product Reviews. Engineering, Technology & Applied Science Research. 13, 3 (Ju 023), 10989–10993.DOI:https://doi.org/10.48084/etasr.5787

Authors Pre-Proof