ISSN: 2182-2069 / E-ISSN: 2182-2077

Predictive Model for Healthcare Software Defect Severity using Vote Ensemble Learning and Natural Language Processing

Dr.A.M. Adeshina[1*], O.S. Adeleye[2], and Dr. Siti Fatimah Abdul Razak[3]

[1*]Faculty of Information Science and Technology, Multimedia University, Malaysia.
am.adeshina@mmu.edu.my, codedengineer@yahoo.com, https://orcid.org/0000-0002-9919-5367

[2]High Performance Computing Research Laboratory, Nigeria. adeleye.os@hpc.com.ng,
https://orcid.org/0000-0001-7280-0232

[3]Faculty of Information Science and Technology, Multimedia University, Malaysia.
fatimah.razak@mmu.edu.my, https://orcid.org/0000-0002-6108-3183

Received: December 19, 2024; Revised: January 25, 2025; Accepted: February 07, 2025; Published: February 28, 2025

Abstract

Software defects are frequent occurrences, which can lead to various problems. These defects are more devastating in healthcare software. Defects in healthcare software have a higher tendency to claim lives directly than the usual manual healthcare procedures. Defects in healthcare software may cause prolonged treatment of patients, aggravating patient recuperation periods and usually leading to direct monetary wastage and healthcare resources. As software systems continue to grow in size and complexity, the likelihood of defects increases. Even with careful planning, thorough documentation, and rigorous process control during development, defects can arise. Moreover, having many software development tasks carried out by individuals, the differences in approaches and actions can give rise to various defects throughout the development process, potentially resulting in disappointments for users in the near future. Unfortunately, existing methods for software defect prediction often struggle with accuracy issues such as underfitting and overfitting, among other imbalances. Moreover, traditional software defect prediction methods often rely on software metrics, such as Line of Code and Cyclomatic Complexity, which may fail to accurately capture program syntax and semantics. This study contributes to software defect severity prediction by introducing a hybridized approach that leverages the power of vote ensemble learning and natural language processing. With the machine learning performance metrics, including accuracy, precision, and other relevant measures, previously proposed model for software defect severity recorded a weighted average accuracy and precision of 0.84 and 0.85, respectively. However, our proposed model demonstrated superior performance, with accuracy and precision scores of 0.9890 and 0.9891, respectively. Interestingly, the effectiveness of the feature engineering techniques employed in the study, such as synthetic minority oversampling aiding in better understanding and capturing the severity patterns of software defects, is highlighted with the results obtained. Moreover, accuracy and precision of the predictive model was further improved through contributions from the utilization of robust independent variables derived from the word ‘embedding approach’. These findings emphasize the potential of this approach for enhancing software quality assurance processes by accurately predicting the severity levels of defects.

Journal of Internet Services and Information Security (JISIS), volume: 15, number: 1 (February), pp. 437-450.
DOI: *10.58346/JISIS.2025.I1.029*

*Corresponding author: Faculty of Information Science and Technology, Multimedia University, Malaysia.

Keywords: Predictive Modeling, Software Defect Severity, Vote Ensemble Learning, Machine Learning, Natural Language Processing.

1 Introduction

Prediction of software defect severity is significant and continuously attracting attention in software engineering and quality assurance (Firoozi & Imanieh, 2017). With the frequently observed dynamicity in domain of software development, identification alongside with clear prioritization of defects based on their severity is essential for effective resource allocation and risk management (Baggyalakshmi et al., 2024). Impact of software defects varies, ranging from minor inconveniences and to the highly rated critical failures which may usually jeopardize an entire system. Without any doubt, accurately assessing the severity of these software defects is not only crucial for making informed decisions but also significant in the optimization of software development processes (Dung, 2023). The current revolution in the technology through the advent of machine learning and data-driven techniques has revolutionized defect prediction introducing approaches offering the potential to automate the defect-severity assessment processes and also further enhancing its accuracy (Devi et al., 2024). Undoubtedly, ensemble learning has emerged, among other similar techniques, with greater hope and promises of novel methodology for improving the reliability of software-defect prediction models (Anada & Ueshige, 2021). With diverse strengths of algorithms, ensemble learning combines multiple models to produce more robust and accurate predictions (Munirathnam, 2020).

2 Related Works

Ensuring the reliability and quality of software is of importance in software development processes. Timely and early software defection is considered as one of the critical aspects in achieving the goal, which can save time and resources, and also enhance the overall software development processes. Similarly, Jacob et al., (2021) study, among other efforts, established the fat that fault-prediction procedures play a pivotal role in this context, aiding developers in identifying potentially problematic segments of code. Of course, software defect prediction is crucial for modern software systems, however, existing methods often struggle with accuracy issues, such as underfitting and overfitting.

Logistic Model Trees (LMT), a novel procedure that uses sentiment analysis approach to create classification ground truths for the binary classification was proposed (Baarah et al., 2021). The predictive analytics framework is considered to be a strong, novel hybrid technique consisting of the combination of both the supervised and the unsupervised learning. Unfortunately, the methodology was considered not sufficient enough to reduce the class imbalance which is typically associated with defect metrics in software engineering and data resampling. Apparently, class imbalance problem is addressed in the real-world domains using oversampling methods which have gained popularity for boosting the representation of rare classes by creating synthetic data (Sharma et al., 2018). However, in our study, towards the efforts of resolving this class imbalance problem, a new approach was introduced by exploring the application of vote ensemble learning.

Vote ensemble learning is a specific form of ensemble learning for the prediction of software defect severity combining different machine learning techniques. The goal is to boost the accuracy of defect prediction with machine learning techniques such as Artificial Neural Networks, Random Forests, and K-Nearest Neighbors (Ranjkesh & Ziabari, 2016). Through harnessing the collective wisdom of all these predictive models, vote ensemble learning seeks to overcome the limitations of individual models and provides a more nuanced and dependable assessment of defect severity.

Traditional software defect prediction methods often rely on software metrics, such as the Line of Code and Cyclomatic Complexity, but they can accurately capture program syntax and semantics. Jacob and his team (Jacob et al., 2021) focused on fault prediction procedures designed to assist in software testing and troubleshooting. These procedures alert developers on those potential flaws in code sections. This study introduces a voting-based ensemble classification method that involves two different approaches for feature selection, a Wrapper-based approach using Python and a Heuristic-based approach using Waikato Environment for Knowledge Analysis (WEKA). Following the feature selection, a classifier was trained using a voting-based ensemble learning algorithm that combined predictions from multiple models. Three base learners, namely, the AdaBoost classifier, Random Forest classifier, and Naive Bayes classifier, were used in the process. The study used NASA's central open datasets to evaluate the performance of the approach. The findings of the study indicate that preprocessing with the Wrapper-based approach outperforms the Heuristic-based approach, and the new voting-based ensemble learning algorithm, comprising Random Forests, Adaboost, and Naive Bayes, proved to be a superior choice compared to the existing algorithms for software defect prediction.

Khan, (2020) presented a comparative analysis results of different machine learning algorithms with hybrid ensemble learning for software defect prediction which proved that the results of the performance of Hybrid Ensemble Learning Technique (HELT) with AdaBoost Support Vector Machine, AdaBoost Random Forest, and Bagging Support Vector Machine gives the best results in Accuracy, Recall, Area Under the Curve (AUC) and F-measure, with up to 100 accuracy. Furthermore, towards software defect prediction for healthcare, Khan et al., (2021) reported a comparison of Support Vector Machine (SVM), Decision Tree (J48), Random Forest (RF), Multilayer Perceptron (MLP), Radial Basis Function (RBF), Hidden Markov Model (HMM), Credal Decision Tree (CDT) and Naïve Bayes (NB). Interestingly, NB and vSVM reported to have fewer Mean Average Error (MAE) and Relative Absolute Error (RAE) respectively. Obviously, the evaluation of software for defect prediction is playing an increasingly critical role in emerging software systems, however, all the existing software defect prediction approaches typically identified to be suffering from low accuracy owing to underfitting or overfitting problems (Realinho et al., 2020).

Therefore, with this study, there is an attempt to address this problem by proposing an ensemble learning approach to achieve accurate defect prediction, where various machine learning algorithms, that is, Artificial Neural Network, Random Forest, and K-Nearest Neighbor methods, are integrated together. The proposed software defect prediction workflow was introduced, and experiments were conducted to verify the effectiveness of the proposed method. Extensive experimental results verified that our proposed method can improve defect prediction accuracy when compared with existing methods previously presented (Yang et al., 2022).

3 Material and Methods

The Proposed Framework

The Ensemble Machine Learning phase combines various algorithms to enhance prediction accuracy. The Data Acquisition phase handles collection of software defect reports, whereas the report pre-processing phase prepares the data for further analysis. The framework, consisting of six phases, the Data Acquisition, the Report Pre-processing, the Sentiment Analysis, the Word Embedding, the Minority Oversampling, and the Ensemble Machine Learning, is proposed for this study. Through the framework, by leveraging natural language processing and analyzing software defect reports from cross-

project benchmarks, the machine learning model for predicting software defect severity levels is aimed to be enhanced. The framework is illustrated in Figure 1.

Dataset

This study incorporates a combination of software defect reports from various repositories, including NASA's central open datasets and datasets from the PROMISE Repository of Software Engineering Databases, Eclipse, Mozilla, and Apache. This is among the strategies to establish a robust and comprehensive big dataset that adequately mitigates the key validity risks identified in initial investigations.

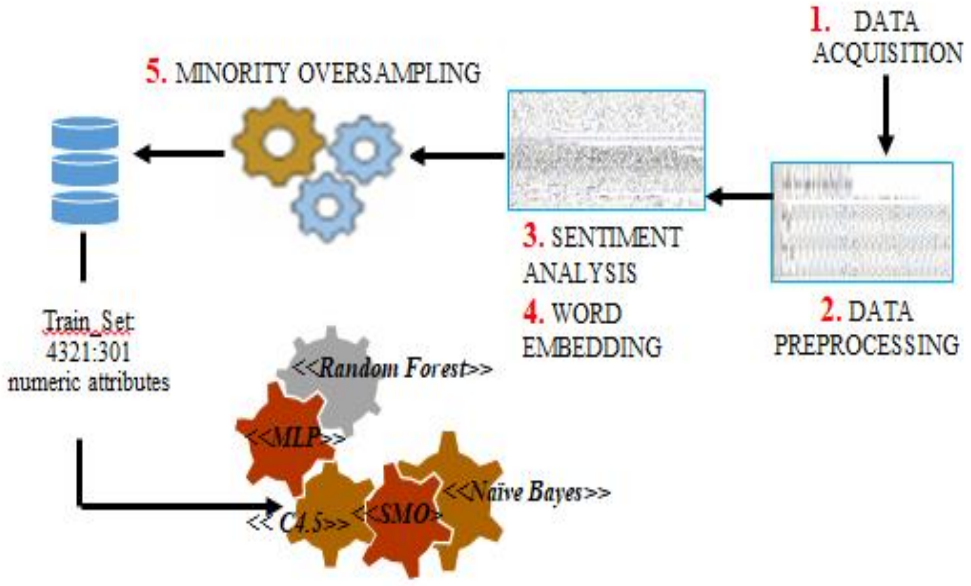

Figure 1: The Proposed Framework (Adeleye & Adeshina, 2023)

Data Pre-Processing

This study aims to enhance the accuracy of the predictive model by ensuring that the extracted features were meaningful and representative of the underlying information in reports. Therefore, the defect reports was allowed to undergo a series of Natural Language Processing (NLP) procedural, the techniques that encompass tokenization, elimination of stop words, conversion to lowercase, and lemmatization. With these techniques employed, significant keywords were extracted, irrelevant words were eliminated and the language used in the reports was standardized.

Sentiment Analysis

Sentiment analysis involves classifying the emotions conveyed in text into positive, neutral, and negative states by assessing their polarity. The VADER approach was used to assign sentiment values to words, with severity levels determined based on their frequency within the defect reports. The compound score, calculated using the VADER-based polarity score calculator, represents the overall sentiment of each report by combining the scores of the three states (positive, neutral, and negative).

Synthetic Minority Oversampling of Defect Reports

The dataset of numeric reports demonstrates imbalanced sampling, which poses challenges to the predictive accuracy of machine-learning models. In such cases, the Synthetic Minority Oversampling Technique (SMOTE) is valuable for generating synthetic instances and effectively balancing representations of severity levels. SMOTE uses the process of computing characteristic attributes among neighboring instances to preserve distinctive features specific to the minority class. Thus, it enhances the performance of the predictive model. One noteworthy insight is that SMOTE efficiently computes the characteristic attributes along the path connecting neighboring instances, thereby inheriting the distinctive nature and features of the original minority class. This optimization enables the identification of any member belonging to the smaller class's 5-k nearest neighbors of the smaller class.

Addressing imbalanced datasets through techniques such as SMOTE not only enhances the model's ability to predict severity levels accurately but also promotes a more comprehensive understanding of the dataset's underlying patterns and characteristics. Through creating synthetic instances that align with the minority class, SMOTE aids in creating a more representative and balanced dataset, enabling machine learning models to make more reliable predictions and contribute to the overall success of the predictive model.

Software Defect Report Preprocessing

Software defect reports were acquired from heterogeneous sources, from historical digital libraries and textual data, and preprocessing of the textual corpus was implemented as described previously. Preprocessing was performed using the Orange data mining toolkit through a preprocess *text* widget.

Sentiment Analysis of Preprocessed Defect Reports

Sentiment analysis was used to analyze the preprocessed defect reports. Sentiment analysis computes four polarities: positive, negative, neutral, and compound sentiment scores for each defect report instance. The compound score, which combines the three polarity scores, was used to determine the label (severity level) for each defect instance. Defect instances with a compound score greater than 0.05 were labeled as low severity, whereas instances with a compound score less than or equal to -0.05 were labeled as high severity. Therefore, each defect instance was assigned a label of low or high severity based on the sentiment compound score computed during this phase of the study. These severity labels serve as the ground truth for the training set.

Word Embedding of the Preprocessed Defect Report

The preprocessed defect report returned from the previous phase was subjected to the word2vec deep-learning model. The model serves as the feature extractor of the numeric feature vector and was to extract 300-no numeric vectors from each of the defect instances supplied during the implementation phase. The Orange widget of the word embedding was activated in the preprocessed defect report, as observed in the framework of Figure 1.

Synthetic Minority Oversampling of Training Set

The encapsulation of the 300-no extracted feature vectors with the 1-no ground truth (low severity or high severity level) for each defect report formed the training set for this study. Therefore, each defect report was represented by a 301-no numeric vectors when the results of sentiment analysis and word

embedding were combined. The 300-no word embedding represented the predictor variables (independent variables), whereas the 1-no compound polarity score (low or high severity) represented the dependent variable. However, existing studies have proven that the representations of low and high severity will not be even; hence, there is bound to be a larger representation of one over the other, which constituted a bias problem. Therefore, minority oversampling of the less-represented class is necessary. The SMOTE code snippet for minority oversampling was implemented in Python.

Machine Learning

The ensemble machine learning phase was implemented using a code snippet. The base learners were trained as single learners using a resampled training set. Upon successful training, the resulting machine learning model was saved and used to design a software defect severity prediction solution that could be used in the industry. The interface of the solution was designed using a code snippet through importation of the necessary Python libraries.

4 Result and Discussion

In the first phase of the implementation, software defect reports were acquired from various repositories, including NASA's central open datasets, Eclipse, datasets from the PROMISE Repository of Software Engineering Databases, and Apache. A total of 4,321 defect reports were obtained and represented in the CSV format. The acquired data were then subjected to preprocessing, sentiment analysis, and word embedding using an orange framework, as depicted in Figure 2.

During the preprocessing phase, the entire set of 4,321 defect reports were cleaned, and the resulting preprocessed data are presented in the data table in Figure 3. The polarity scores (positive, negative, neutral, and compound) for each of the 4,321 reports were calculated and are enclosed in the red ribbon of the data table shown in Figure 4. The compound score was used to assign a severity label (low or high) to each of the 4,321 reports.

Of the total, 2,578 defect reports were labeled as low-severity, whereas the remaining 1,743 were labeled as high-severity. This data sampling resulted in a highly imbalanced distribution, with a majority of low-severity instances and a minority of high-severity instances. To address this imbalance, the minority oversampling technique, specifically the synthetic minority oversampling technique (SMOTE), was employed.

Figure 5 presents a data table containing 300 nonnumeric feature vectors extracted from each of the 4,321 defect instances. These feature vectors served as independent variables and were used to predict the dependent variable, which was determined by the compound sentiment score. It is evident that there were 2,578 instances of low severity in the defect instances, indicating the same distribution in the word-embedding data table. In contrast, the high-severity class was in the minority, with only 1,743 representations compared to 2,578 in the low-severity class. To address the class imbalance, high-severity instances were oversampled using SMOTE, which involves generating synthetic versions of their representations to increase the number of instances. The results of minority oversampling, both before and after SMOTE computation, are presented in Figure 6, which displays the sampling states.

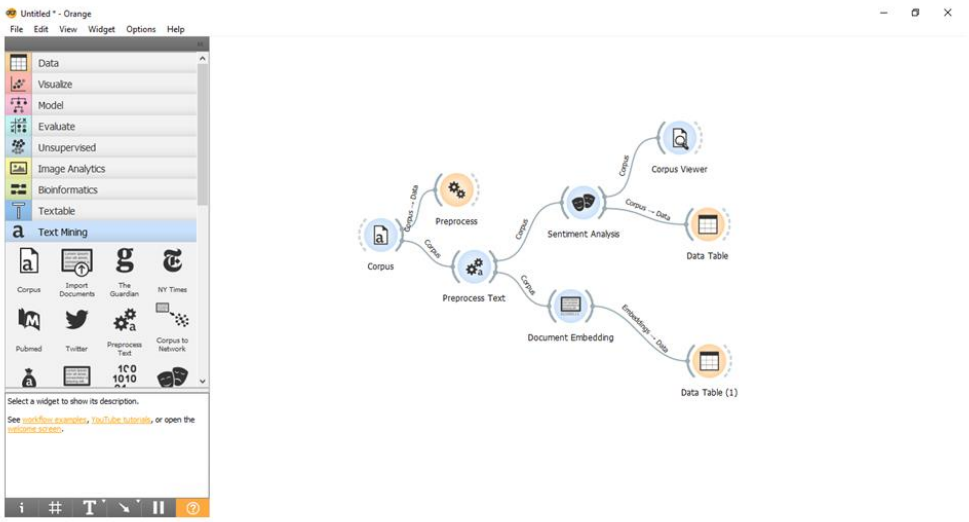

Figure 2: Framework Used on the Orange Data Mining Toolkit

An ensemble of base learners, including Random Forest, Decision Tree, MLP, Sequential Minimal Optimization (SMO), and Naïve Bayes, was trained on the resampled training set to create a software defect severity predictor solution. The solution interface is illustrated in Figure 7. The Defect Report Pane allows the user to upload a software defect report file selected from existing files on the host computer, as shown in Figure 8. In Figure 9, the defect report from the selected file is displayed in the 'Defect Report Pane'.

The solution reads the embedded version of the defect report at the back-end. When the user clicks the 'predict' button, the severity level of each defect instance in the selected file is shown on the pane located beneath the 'Defect Report Pane', as depicted in Figure 10.

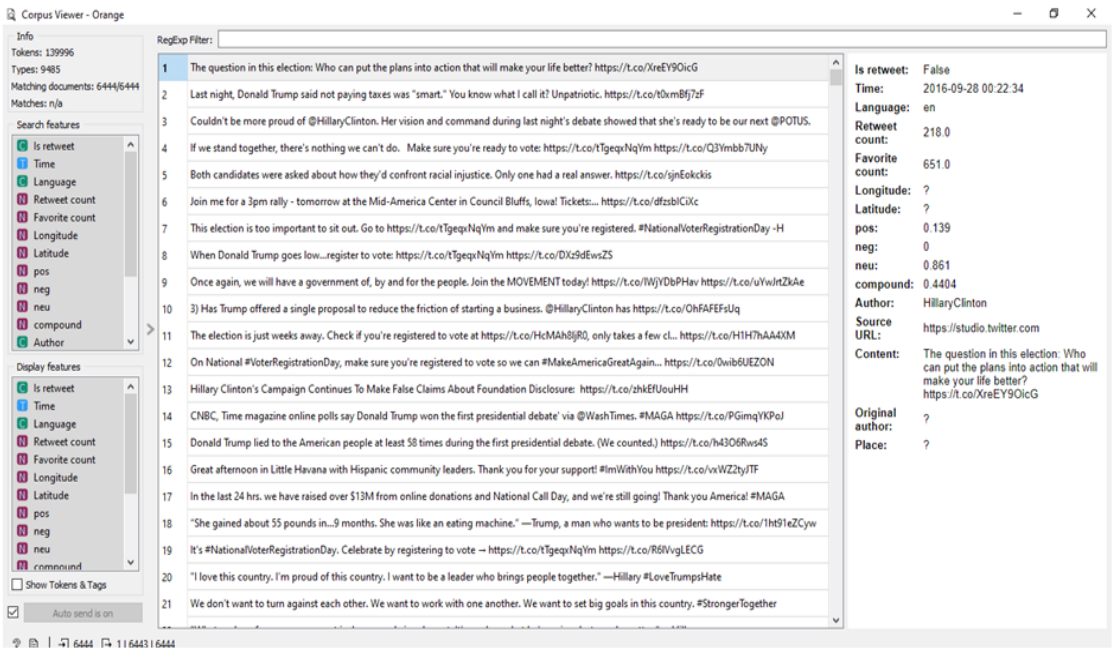

Figure 3: Framework Used on the Orange Data Mining Toolkit

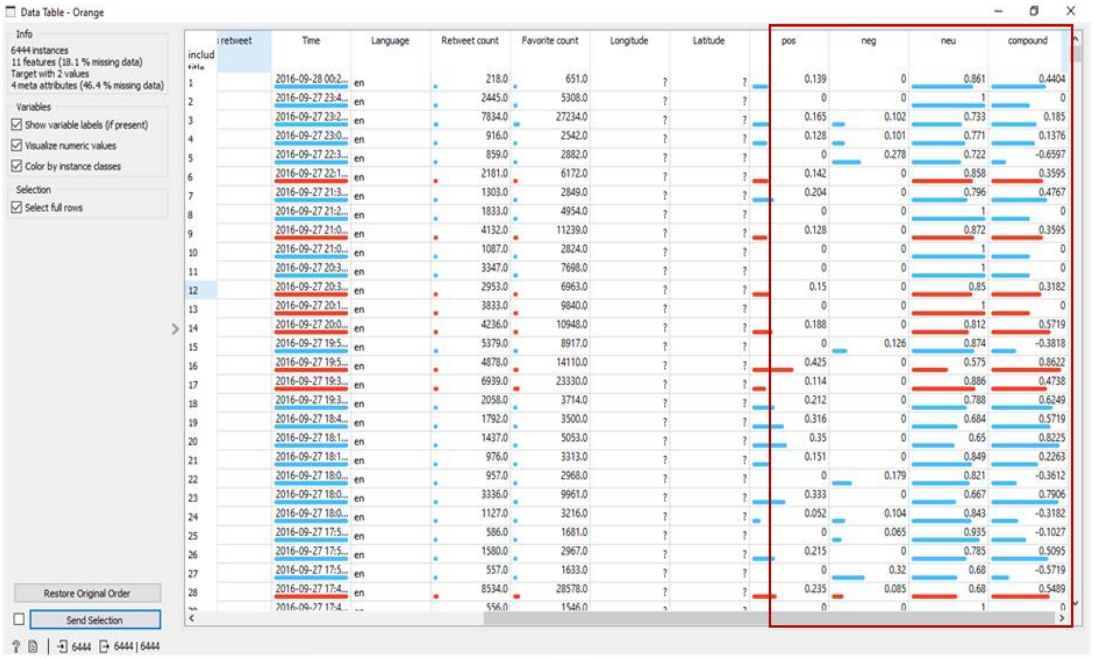

Figure 4: Data Table Showing Sentiment Polarity Scores

Figure 5: Data Table Showing the Numeric Feature Vectors

Figure 6: Data Distribution Before and After SMOTE

Figure 7: The Interface of the Solution

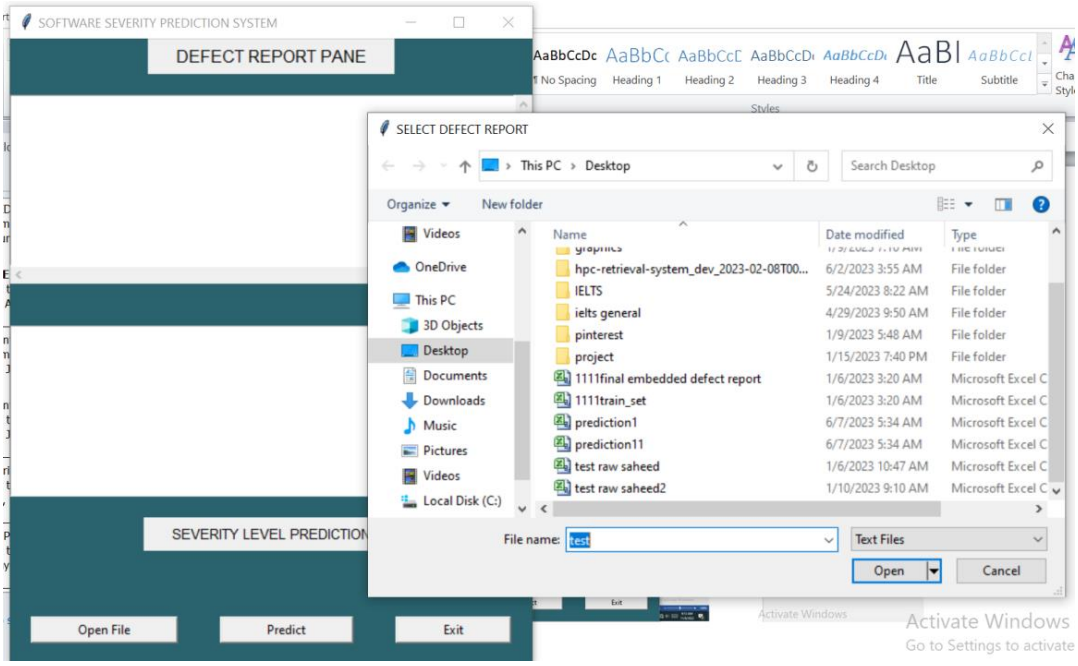

Figure 8: The Upload Widget of the Solution

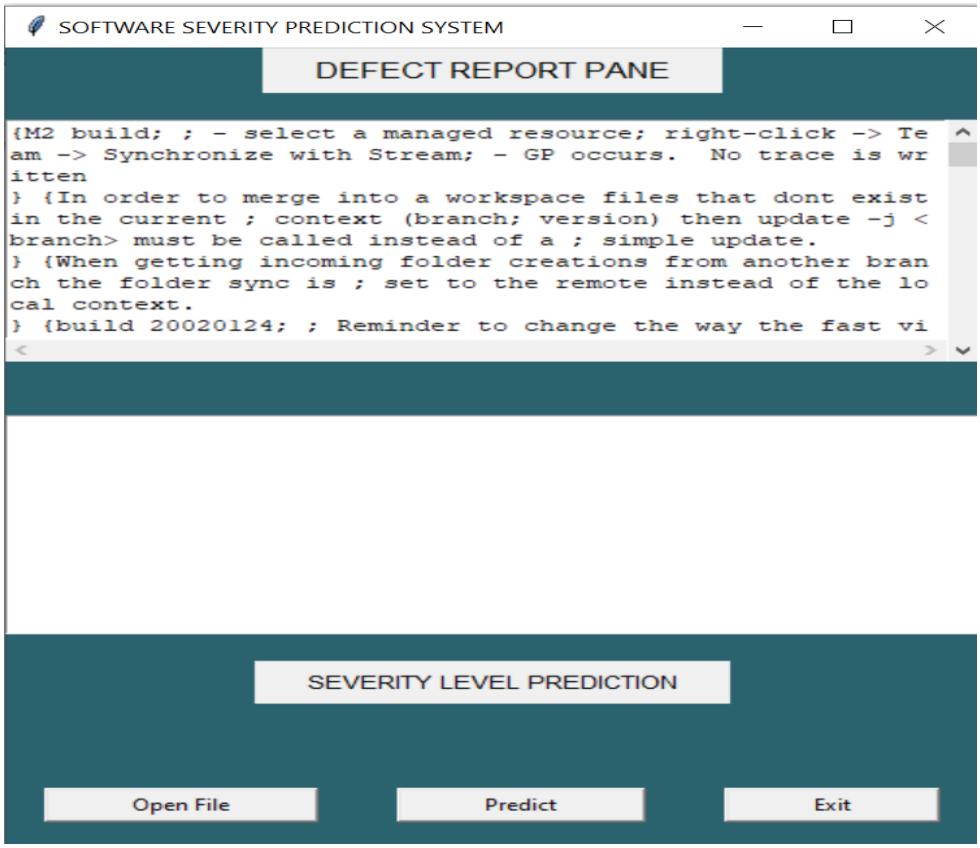

Figure 9: Defect Report Pane Showing Uploaded Defect Report Instances

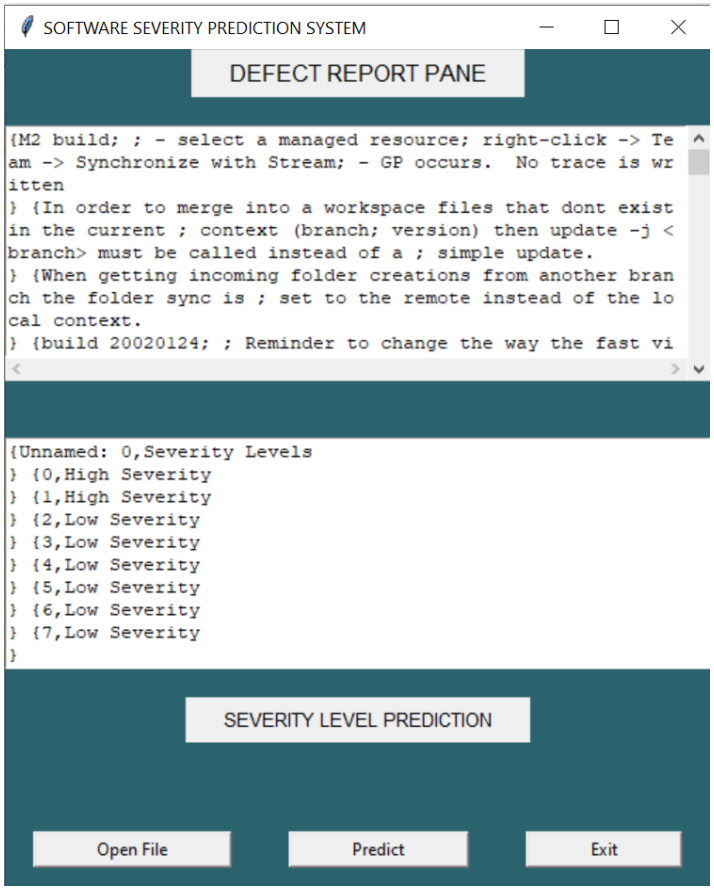

Figure 10: Typical Prediction Interphase

Performance Metrics of the Ensemble Learner

The performance of the model was evaluated based on the traditional metrics of the machine learning methodology. The weighted averages of the metrics are as presented in Table 1, while the metrics considered are described as follows:

i. **Accuracy**: The accuracy metric returns the complete exactness of the model, which is a function of the fraction of the total defect reports correctly predicted by the solution (Eq. 1).
ii. **Precision**: This metric measures the proportion of predictions correctly classified as either high or low (Eq. 2).
iii. **Recall:** Recall is also referred to as the sensitivity of the solution and indicates the proportion of instances that are correctly predicted (Eq. 3).
iv. **F1-Measure**: The metric score moderates precision and recall into a single metric and is widely regarded as a better metric for model evaluation (Eq. 4).

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$\text{Precision} = \frac{TP}{TP+FP} \quad (2)$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad (3)$$

$$F_1 = 2 * \frac{\text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}} \quad (4)$$

Where TP is the true positive, TN is the true negative, FP is the false positive, and FN is the false negative rate.

Table 1: Result of the Ensemble Model

S/N	Evaluator	Weighted Average
1	Accuracy	0.9890
2	Precision	0.9891
3	Recall	0.9710
4	F1 Measure	0.9889

In Table 1, regarding the weighted averages of the major indicators, the model's performance is cutting-edge. The class of defect reports with low severity was accurately anticipated to be of low severity and the class of defect reports with high severity was correctly expected to be of high severity, demonstrating the model's high degree of reliability. The F1 measure, which is a more trustworthy evaluation metric than accuracy and calculating the Harmonia mean of precision and recall, also produced a cutting-edge result.

5 Conclusion

A machine learning-based ensemble model was conceptualized and implemented to predict the severity level of software defects identified during software testing. Sentiment analysis using natural language processing was employed to determine the ground truth of the supervised machine learning methodology. The compound score computed from the positive, negative, and neutral polarity scores of 4321 defect instances was used to determine the dependent variable of the model. A 300-no predictor variable was extracted from each of the 4321 defect instances as the independent variable, which was later oversampled through the computation of synthetic versions to increase the representations of high-severity instances, totaling 1743 as compared to the 2578 low-severity instances. A state-of-the-art performance metrics was achieved with Accuracy, Precision, Recall, and F1-measure metrics which returned a 0.9890, 0.9891, 0.9710, and 0.9889 weighted averages respectively. The results show the proficiency of the conceptual framework in accurately determining the severity level of software defects in the natural language processing of its reports. Technology Acceptance Model was used to gauge the acceptability perception of industry experts on the solution. The perceived ease of use and perceived usefulness constructed returned both an impressive perception state among the 14 industry experts who tested the solution.

This study collectively contribute to the ever-evolving landscape of software defect prediction, highlighting the significance of advanced techniques, ensemble learning, and machine learning algorithms in enhancing the accuracy and ultimately ensuring the reliability of software systems. The insights gained from this study holds substantial promise for the ongoing improvement of software-development processes and quality assurance measures.

This study has been able to design and implemented a natural language processing and machine-learning-based conceptual framework for the prediction of software defect severity. The hybridization of synthetic minority oversampling, sentiment analysis, and word-embedding feature engineering techniques for software quality assurance were likewise achieved. Furthermore, the study implemented a defect severity predictor solution for the Quality Assurance software industry.

Therefore, the implementation of a resampling technique is highly recommended for software defect severity prediction studies because of the lower representation of defect instances in repositories compared to non-defect software metrics.

References

[1] Adeleye, O. S., & Adeshina, A. M. (2023). Predictive Modelling of Software Defect Severrity using Vote Ensemble Learning: A conceptual Framework. *Nigerian Journal of High Performanc Computing Applications (NJHPCA)*, 15-23.

[2] Anada, H., & Ueshige, Y. (2021). Anonymous deniable predicate authentication scheme with revocability. *Journal of Internet Services and Information Security*, *11*(3), 1-15.

[3] Baarah, A. L. A. D. D. I. N., Aloqaily, A. H. M. A. D., Salah, Z. A. H. E. R., & Alshdaifat, E. (2021). Sentiment-based machine learning and lexicon-based approaches for predicting the severity of bug reports. *Journal of Theoretical and Applied Information Technology*, *99*(6), 1386-1401.

[4] Baggyalakshmi, N., Jokani, N. R., & Revathi, R. (2024). Managing and Billing Software for Hypermarket. *International Academic Journal of Innovative Research*, *11*(1), 17–26. https://doi.org/10.9756/IAJIR/V11I1/IAJIR1103

[5] Devi, B. A., Jaganathan, S., Shah, P. K., & Venkatapathy, N. (2024). Prediction of Premature Retinopathy Fundus Images Using Dense Network Model for Intelligent Portable Screening Device. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 15*(2), 170-182. https://doi.org/10.58346/JOWUA.2024.I2.012

[6] Dung, N. T. (2023). Smart Urban Development Solutions in Vietnam. *International Academic Journal of Science and Engineering*, *10*(1), 01–06. https://doi.org/10.9756/IAJSE/V10I1/IAJSE1001

[7] Firoozi, H., & Imanieh, M. (2017). Improvement The Efficiency of Thin Film CIGS Solar Cells by Changing the ZnO Layers using Silvaco Software. *International Academic Journal of Science and Engineering*, *4*(1), 65–73.

[8] Jacob, R. J., Kamat, R. J., Sahithya, N. M., John, S. S., & Shankar, S. P. (2021, October). Voting based ensemble classification for software defect prediction. In *2021 IEEE Mysore Sub Section International Conference (MysuruCon)* (pp. 358-365). IEEE. https://doi.org/10.1109/MysuruCon52639.2021.9641713

[9] Khan, B., Naseem, R., Shah, M. A., Wakil, K., Khan, A., Uddin, M. I., & Mahmoud, M. (2021). Software defect prediction for healthcare big data: an empirical evaluation of machine learning techniques. *Journal of Healthcare Engineering*, *2021*(1), 8899263. https://doi.org/10.1155/2021/8899263

[10] Khan, M. Z. (2020). Hybrid ensemble learning technique for software defect prediction. *International Journal of Modern Education & Computer Science*, *12*(1), 1-10. https://doi.org/10.5815/ijmecs.2020.01.01

[11] Munirathnam, R. (2020). Maximizing Efficiency with Portfolio Management Metrics: Driving Project Success through Data-Driven Strategies. *International Journal of Advances in Engineering and Emerging Technology*, *11*(2), 125–140.

[12] Ranjkesh, M., & Ziabari, M. (2016). Persian Language telephone and Microphone Speaker identification using neural networks. *International Academic Journal of Science and Engineering*, *3*(2), 6–12.

[13] Realinho, V., Romao, T., & Dias, A.E. (2020). A Language for the End-user Development of Mobile Context-Aware Applications. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 11*(1), 54-80.

[14] Sharma, S., Bellinger, C., Krawczyk, B., Zaiane, O., & Japkowicz, N. (2018, November). Synthetic oversampling with the majority class: A new perspective on handling extreme imbalance. In *2018 IEEE international conference on data mining (ICDM)* (pp. 447-456). IEEE. https://doi.org/10.1109/ICDM.2018.00060

[15] Yang, Z., Jin, C., Zhang, Y., Wang, J., Yuan, B., & Li, H. (2022). Software defect prediction: an ensemble learning approach. In *Journal of Physics: Conference Series* (Vol. 2171, No. 1, p. 012008). IOP Publishing. https://doi.org/10.1088/1742-6596/2171/1/012008

Authors Biography

Dr.A.M. Adeshina is a scholar currently with the Faculty of Information Science and Technology, Multimedia University, Malaysia. He has his core research areas in Artificial Intelligence, High-Performance Computing, GPU-based Algorithms, Neuro informatics, Data and Cyber Security. He gained his B.Sc. (Hons.), Master's, and PhD degrees from the University of Ilorin, Nigeria, Multimedia University, Malaysia, and the University Tun Hussein Onn Malaysia respectively. He was a Postdoctoral Fellow with the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia from 2014 to 2016. Over the years, he has been quite active in researching and applying High-Performance Computing and Artificial Intelligence approaches to the development of software solutions in medical and paramedical domains, including researching best approaches to reducing algorithmic complexities, and improving performances and bottlenecks of some of the algorithms in healthcare. Some of his findings have been presented at international conferences and his contributions are frequently being acknowledged in various capacities, published in reputable international journals, and included in most of the notable databases.

Olabode Saheed Adeleye completed his B.Sc. degree in Computer science at the Federal University of Agriculture Abeokuta in 2017, and M.Sc. degree in Computer Science at Crescent Abeokuta, Nigeria in 2024. Mr. Olabode is an active member of the High Performance Computing Research Laboratory, Nigeria and he is currently working as a Word press Developer at Reliance Infosystems Limited, Nigeria. He has his core area of research interest in Machine Learning, Artificial Intelligence and Software Development. Mr. Olabode Saheed Adeleye has taken part in some software development projects. Outside of work, he enjoys watching football and meeting people.

Dr. Siti Fatimah Abdul Razak received her B. Sc (Hons) with Education where she majors in Mathematics and Information Technology, and Master of Information technology majoring in Science and System Management from Universiti Kebangsaan Malaysia in year 2004. She completed her doctorate studies in Information Technology from Multimedia University. She is also currently an Assistant Professor in Faculty of Information Science and Technology, Multimedia University. Apart from her administration and teaching responsibilities, she is also supervising postgraduate students and undergraduate final year projects. Her research interest includes vehicle safety applications, rule mining, machine learning, Internet-of Things, information systems development and educational technology. She is currently a member IEEE and the Centre for Intelligent Cloud Computing (CICC), a research centre in Multimedia University. She is also a registered Professional Technologist with the Malaysia Board of Technologists (MBOT).