AMERICAN JOURNAL OF MANAGEMENT AND IOT MEDICAL COMPUTING
Peer Reviewed, Referred & Indexed Journal
E-ISSN: 3069-0110
Vol.5, No.2(2026)
www.ajmimc.com

Self-Attentive Semantic Distillation for Autonomous Criticality Mapping in Clinical Cyber Incident Reports

Ritesh Kumar[1*], Garnepalli Pranay kumar[2], Govindaloori Naga Sai Ram[2], Radharapu Ram prasad[2]

[1]Assistant Professor, [2]UG Student,[1,2]Department of Computer Science and Engineering

[1,2]Kommuri Pratap Reddy Institute of Technology, Ghanpur, Ghatkesar, 501301, Telangana, India.

*Correspondence: Ritesh Kumar (riteshkumar237@gmail.com)

ABSTRACT

The healthcare domain has increasingly become a target for cyber threats, with large-scale exposure of patient data and a steady rise in reported security vulnerabilities impacting hospital infrastructures. The rapid growth of unstructured medical security reports makes manual severity assessment inefficient, inconsistent, and unsuitable for timely response. To overcome these limitations, this study presents an automated text analysis framework based on Natural Language Processing (NLP), utilizing a medical security dataset composed of incident records, advisories, and vulnerability disclosures. The process begins with preprocessing and Exploratory Data Analysis (EDA) to ensure data quality through normalization, cleaning, and visualization. For contextual understanding, Lightweight RoBERT (Robustly Optimized BERT) is applied to generate semantic embeddings while maintaining computational efficiency. Unlike traditional approaches, the proposed system integrates Deep Neural Network (DNN)-based feature selection with Natural Gradient Boosting (NGBoost) to enhance classification performance. For comparison, baseline models such as Stochastic Gradient Descent (SGD) and NGBoost classifiers are also evaluated. The framework performs binary classification to distinguish between normal and high-severity vulnerabilities, enabling effective prioritization of critical issues. By combining contextual embeddings with advanced feature selection, the model improves accuracy, reduces false predictions, and enables faster response in real-time scenarios. This solution provides a scalable and efficient mechanism for automated vulnerability assessment, strengthening cybersecurity defenses and supporting improved risk management in healthcare systems.

Keywords: Healthcare cybersecurity, Medical data security, Cyber threats in healthcare, Natural Language Processing (NLP), Deep Neural Networks (DNN), Natural Gradient Boosting (NGBoost).

1. INTRODUCTION

As contemporary software applications expand in scale, functionality, and interconnectivity, maintaining their reliability and robustness has become increasingly difficult, with defects remaining an unavoidable outcome of complex development processes. These defects, commonly known as software bugs, arise from issues such as incorrect memory management, runtime exceptions, logical inconsistencies, and deviations from intended execution behavior [1]. Such problems can lead not only to system instability but also to critical security vulnerabilities, especially in environments where fault tolerance is minimal. Historical incidents involving security breaches have demonstrated the severe financial and operational consequences of unresolved defects, highlighting the importance of early detection and precise severity assessment. In practice, when users encounter issues, they report them through a Bug Tracking System (BTS), providing structured and unstructured information including summaries, detailed descriptions, platform specifications, and perceived severity levels. These reports are then analyzed by developers or

triagers, who must manually inspect and prioritize a large volume of submissions, making the process time-consuming, resource-intensive, and prone to human inconsistency [2].

The impact of software bugs varies significantly depending on their effect on system functionality and user experience. Critical defects may completely disrupt application usage, major issues can impair essential features, minor bugs cause limited inconvenience, and low-priority issues typically involve cosmetic or non-functional inconsistencies [3]. Accurately assigning severity levels is therefore essential for efficient

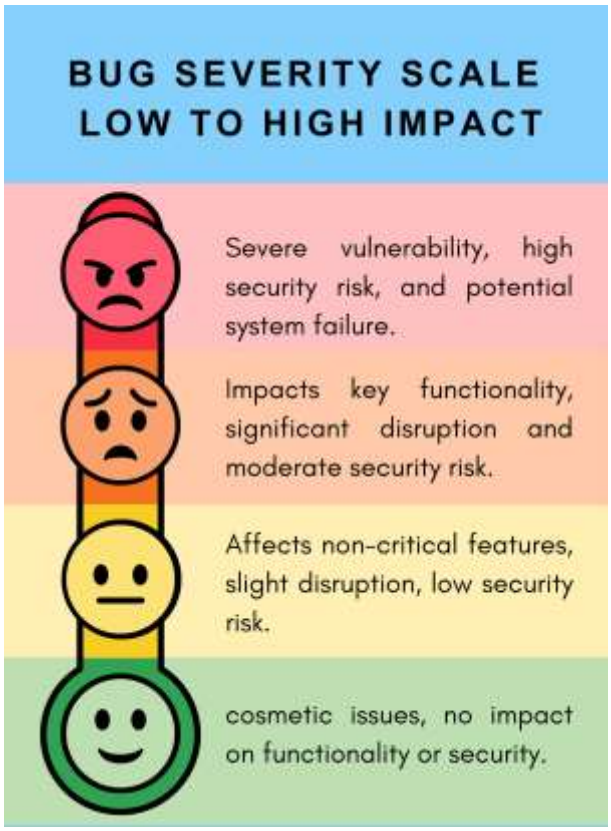


Fig. 1.1: The basic classes of bug severity.

resource allocation and timely resolution. However, manual severity classification presents several challenges, as it relies heavily on interpreting textual content within bug reports, which may be ambiguous, incomplete, or inconsistent. Furthermore, discrepancies often arise between users and developers regarding severity assignments, leading to frequent reclassification. Studies indicate that a substantial proportion of bug reports undergo severity reassignment, which increases resolution time and negatively impacts development efficiency [4, 5]. This reassignment process not only delays issue handling but also affects task distribution among development teams.

2. Related Work

Research in bug deduplication, classification, and severity prediction has evolved significantly with the integration of machine learning (ML), deep learning (DL), and natural language processing (NLP)

techniques. Early studies focused on analyzing textual bug reports and understanding their characteristics, while recent approaches emphasize hybrid models, data preprocessing strategies, and explainable systems to improve automation and decision-making in software maintenance.

2.1 Survey and Review-Based Studies

Comprehensive surveys have played a crucial role in understanding the landscape of bug analysis techniques. Qian et al. [6] presented an extensive review of bug deduplication and triage methods, categorizing approaches based on runtime information and textual analysis. Their work highlights key methodologies, datasets, and research challenges, providing a strong foundation for future advancements.

Similarly, Bhakar et al. [8] conducted a systematic review of computational intelligence techniques for severity assessment, primarily in healthcare domains. Their study emphasizes the effectiveness of ML and DL models in handling high-dimensional data, while also identifying challenges such as data imbalance, feature selection, and interpretability, which are equally relevant in software defect analysis.

Galić et al. [12] further explored deep learning techniques for severity assessment in image-based applications, demonstrating how models like CNNs, RNNs, and GANs can extract complex patterns. Although focused on medical imaging, their findings provide valuable insights applicable to other domains involving complex data representations.

2.2 Machine Learning and NLP-Based Classification

Machine learning combined with NLP has become a dominant approach for bug classification and prioritization. Tabassum et al. [7] proposed a hybrid framework integrating NLP preprocessing, feature extraction techniques such as TF-IDF and Word2Vec, and supervised ML models for multi-class bug classification and severity prediction. Their results show that combining multiple feature extraction strategies improves classification performance.

Liu et al. [10] introduced a defect detection method that integrates program dependency graph analysis with contextual feature extraction. By combining structural and contextual information and applying an optimized SVM classifier, the approach improves detection of high-severity defects, particularly those influenced by code dependencies.

2.3 Impact of Data Preprocessing and Feature Engineering

Data preprocessing plays a critical role in improving model performance. Ji et al. [9] investigated the effect of stop-word removal on bug classification using various deep learning models, including CNN, LSTM, GRU, Transformer-based architectures, and BERT. Their findings indicate that preprocessing decisions significantly influence model accuracy, as different models interpret contextual information differently. This highlights the importance of careful feature engineering and preprocessing in building robust classification systems.

2.4 Clustering and Hybrid Learning Approaches

Unsupervised and hybrid learning methods have been explored to address limitations in labeled data availability. Khandakar et al. [11] utilized k-means clustering to group severity levels, followed by supervised classification using ML and CNN models. Their approach demonstrates that combining clustering with supervised learning improves robustness, particularly in scenarios with limited labeled data.

These hybrid strategies highlight the growing trend of integrating multiple learning paradigms to enhance classification accuracy and adaptability across different datasets.

2.5 Explainable and Recommendation-Based Systems

Recent research has also focused on improving decision-making through explainability and recommendation systems. Samir et al. [13] proposed explainable models for developer recommendation and bug assignment. By analyzing historical bug data and incorporating explainability into the recommendation process, their system enhances transparency and trust in automated bug triage. This approach supports more efficient bug resolution workflows and improves collaboration among development teams.

3. PROPOSED METHODOLOGY

The architecture follows a phased pipeline that progressively refines data for accurate prediction. It begins with data ingestion and structuring using pandas, where raw medical text is cleaned and encoded into a usable format. The next phase performs EDA using NLTK and Gensim to extract linguistic and statistical insights from the data. A deep feature refinement stage then generates contextual embeddings using RoBERTa, which are further optimized through a DNN into compact feature vectors. These refined features are used to train multiple classifiers such as SGD and NB in parallel. The models are evaluated and compared using metrics and visual analysis to select the best performer. In the final phase, the system is deployed through a Django web application for real-time usage. Users can input data (single or batch), which passes through the same pipeline to generate predictions efficiently as shown in Fig. 2.

AMERICAN JOURNAL OF MANAGEMENT AND IOT MEDICAL COMPUTING
Peer Reviewed, Referred & Indexed Journal
E-ISSN: 3069-0110
Vol.5, No.2(2026)
www.ajmimc.com

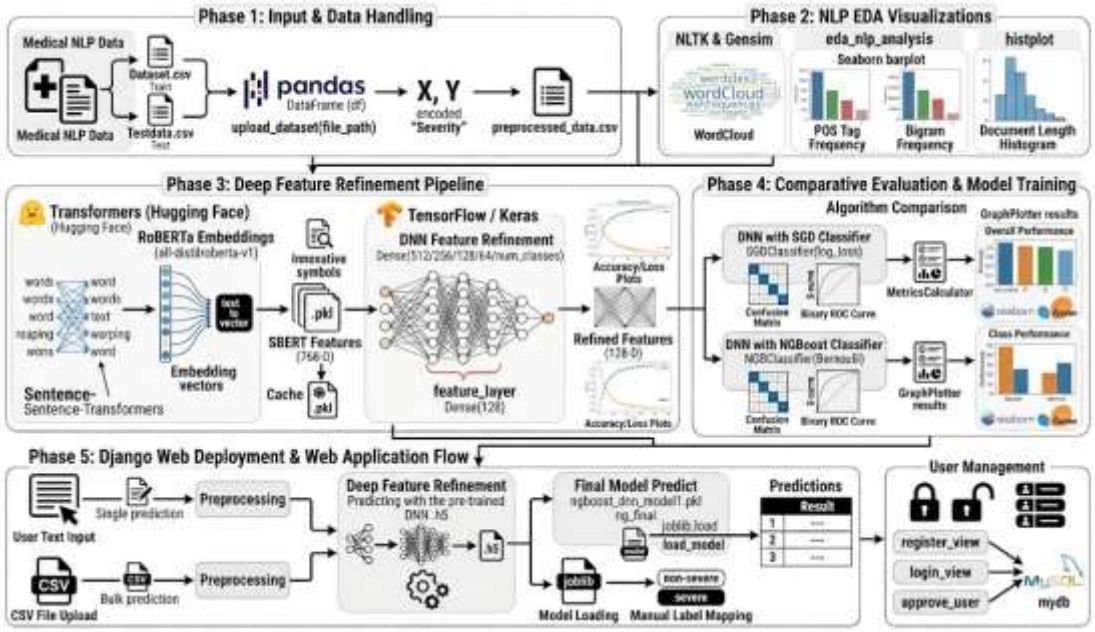


Fig. 2: Proposed System Architecture for Bug Severity Detection.

Step 1: Data Collection – Medical NLP Dataset: In the first step of the process, the system gathers raw data from various medical sources, including security incident reports, vulnerability disclosures, and advisories. This data is typically unstructured and consists of textual reports generated from different healthcare entities, including hospitals, insurance providers, and other healthcare systems. The raw data needs to be collected, centralized, and organized into a format that can be used for further processing.

- **Data Source Identification:** The data is sourced from multiple channels, such as security advisories, vulnerability reports, and incident logs that describe potential security flaws within healthcare systems.

- **Dataset Preparation:** After collection, this unstructured data is converted into a structured format, like a CSV or database, ensuring it is ready for subsequent processing steps.

The goal of this stage is to prepare the medical NLP dataset for cleaning and preprocessing in the next step, ensuring all relevant information is available for feature extraction.

Step 2: NLP Preprocessing & Exploratory Data Analysis (EDA): Once the dataset is collected, it undergoes an NLP preprocessing phase. This step involves several text-cleaning operations, where the raw textual data is normalized and made ready for analysis. Additionally, Exploratory Data Analysis (EDA) is performed to gain insights into the dataset's structure, uncover patterns, and understand its characteristics. These steps are essential for making the data suitable for feature extraction and model training.

- **Text Cleaning:** The data is processed to remove noise such as irrelevant characters, formatting issues, and stopwords. Text is then tokenized into individual words, and each word is normalized through lemmatization, which reduces it to its root form (e.g., "running" becomes "run").

- **Exploratory Data Analysis (EDA):** The system analyzes the dataset to identify trends, common phrases, and other important features. This involves generating word clouds, histograms of document lengths, and exploring the frequency of terms (such as n-grams and part-of-speech tags). These visualizations provide useful insights that guide the subsequent stages of feature extraction.

This preprocessing step ensures the data is cleaned and transformed into a form that can be used to train machine learning models, providing a solid foundation for feature extraction.

Step 3: Feature Extraction Using Lightweight RoBERT with Word Embeddings

In the feature extraction step, the system uses Lightweight RoBERT (Robustly Optimized BERT) to convert the preprocessed text into meaningful feature vectors, known as word embeddings. RoBERTa is an optimized version of BERT (Bidirectional Encoder Representations from Transformers) that can generate contextualized word representations. These embeddings capture the semantic meaning of the text, which is crucial for the accurate classification of bug severity.

- **RoBERT Model:** The Lightweight RoBERT model is applied to the preprocessed text to generate word embeddings. These embeddings help the model understand the meaning of words within their context, improving the system's ability to differentiate between normal and severe bugs based on the report descriptions.

- **Feature Representation:** The embeddings are processed and aggregated into fixed-length feature vectors, ready to be used in machine learning models. The system may also include non-textual features (such as metadata or numerical values) if available, ensuring that the model has all the relevant information for classification.

This step transforms raw text data into structured feature vectors, which will then be used to train machine learning models for severity classification.

Step 4: Model Training & Classification: Once the features have been extracted, the system applies machine learning models to classify bug severity into two categories: Normal or Severity. In this phase, both existing and proposed models are tested to evaluate which performs best on the dataset. The models trained include SGD Classifier, NGBoost, and an enhanced version of NGBoost that integrates DNN-based feature selection.

- **Existing Models (SGD & NGBoost):** The system first evaluates the performance of existing models like SGD (Stochastic Gradient Descent) and NGBoost (Natural Gradient Boosting). These classifiers are trained using the features extracted in Step 3 to determine how well they can predict the severity of bugs in medical datasets.

- **Proposed Model (DNN Feature Selection + NGBoost):** The proposed model incorporates DNN based feature selection, which enhances the feature selection process. The DNN identifies which features are most important for predicting bug severity, helping the NGBoost model perform better by focusing on the most relevant information.

The goal of this step is to fine-tune the models to ensure they can classify the severity of bugs accurately and efficiently, with the proposed model expected to outperform the traditional models in terms of prediction accuracy.

Step 5: Bug Severity Classification: After the models have been trained, they are used to classify bug severity in new, unseen data. The classification process involves using the trained classifiers (either the NGBoost or the enhanced DNN and NGBoost model) to predict whether a bug is of normal severity or requires urgent attention.

- **Prediction:** The models are tested on the validation dataset, where they predict the severity of each vulnerability or bug described in the medical reports.

- **Output:** The predictions are made into binary labels: Normal for low-severity bugs and Severity for high-priority bugs. These labels are used to determine which bugs need immediate attention and which can be handled later.

This classification step provides clear outputs that can be used to prioritize security vulnerabilities within the medical systems, ensuring that the most critical issues are addressed first.

Step 6: Real-Time Response & Actionable Insights: Finally, once the bug severity is classified, the system generates real-time actionable insights to help cybersecurity teams prioritize their responses. These insights are based on the severity classifications from the previous step.

- **Real-Time Analysis:** The system continuously processes new medical security reports, classifying bugs as they arise. This real-time processing ensures that vulnerabilities are identified and addressed as quickly as possible.

- **Actionable Insights:** The system generates alerts or notifications indicating which bugs are of high severity, suggesting that they should be addressed immediately. The insights also include recommendations for fixing or mitigating the vulnerabilities, enabling cyber security teams to act swiftly.

This final step ensures that healthcare organizations can quickly respond to security threats, reducing the risk of data breaches or other cyber security issues that could compromise patient safety or data privacy.

Proposed DNN with NGBoost

The Proposed DNN with NGBoost framework integrates the power of Deep Neural Networks (DNNs) for learning high-level, non-linear representations from Lightweight RoBERTa word embeddings and the robustness of Natural Gradient Boosting (NGBoost) to provide probabilistic classification. This combined approach allows the model to not only classify bug severity as Severe or Non-Severe but also quantify the uncertainty of its predictions. The DNN handles the complex, hierarchical feature learning from the text data, while NGBoost optimizes the model using natural gradients, making it more stable and accurate in high-dimensional settings like medical NLP. This hybrid model provides a probabilistic output, improving decision-making by informing stakeholders not just about the severity but also the confidence of the prediction.

Input Representation (Lightweight RoBERTa Embeddings): The input to the proposed model consists of Lightweight RoBERTa embeddings derived from preprocessed medical bug reports, advisories, or

vulnerability descriptions. These embeddings provide rich, high-dimensional representations of the text, capturing the semantic context of each word and phrase. For example, words like "critical," "exploit," and "patch" will have specific contextual meanings that are important for distinguishing between Severe and Non-Severe bugs.

DNN Feature Extraction: The Lightweight RoBERTa embeddings are passed through the Deep Neural Network (DNN), which consists of multiple hidden layers designed to learn complex patterns in the data. Each layer in the DNN applies a linear transformation followed by a non-linear activation function (such as ReLU), enabling the network to capture intricate semantic relationships. The deeper the layers, the more abstract the features the DNN can learn. This step is crucial for understanding the deeper structure of the reports and recognizing patterns indicative of bug severity.

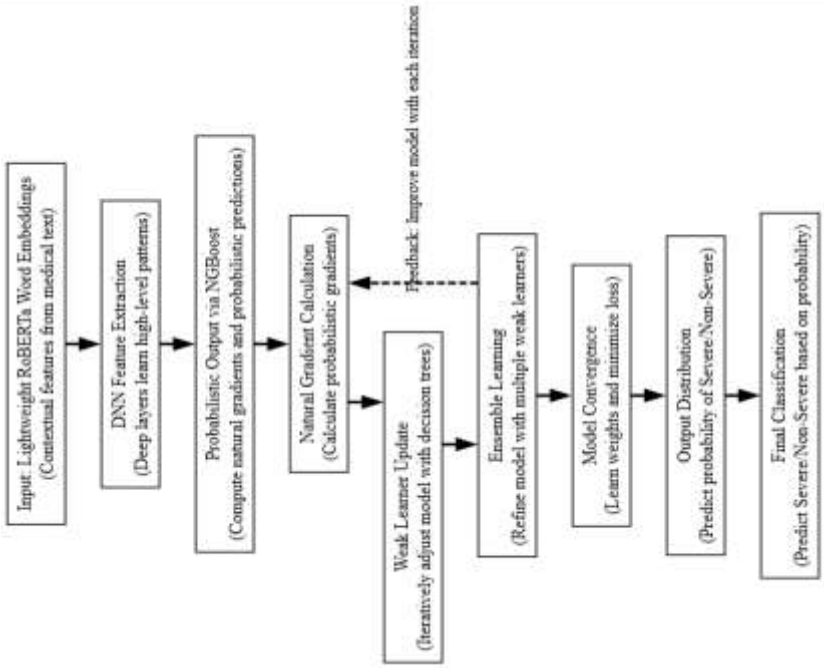


Fig. 3: Internal working of DNN with NGBoost.

Natural Gradient Calculation in NGBoost: NGBoost computes natural gradients to adjust model weights efficiently, focusing on the distribution of predicted values rather than just minimizing errors in deterministic predictions.This provides more stable updates in the presence of complex or noisy features like medical terminology, which are often observed in bug reports.

Update Weak Learners: In NGBoost, weak learners (such as decision trees) are iteratively trained, where each new learner corrects the errors made by the previous learners. NGBoost works by updating these trees based on the natural gradient computed in the previous step. The focus on probabilistic adjustments means the model progressively improves its ability to handle uncertainty, especially when distinguishing subtle variations in bug severity.

Ensemble Learning: As boosting progresses, the ensemble of decision trees becomes stronger, with each new weak learner contributing to the final prediction. Through this process, the model refines its understanding of which features (from the embeddings and the DNN) most effectively separate severe bugs from non-severe ones.

Output Distribution from NGBoost: Once the model converges, NGBoost outputs a probability distribution over the classes (Severe, Non-Severe). This probabilistic output gives not just the predicted label but also the confidence level of the prediction. For example, if a report is predicted as Severe, NGBoost might output a probability of 0.85, indicating 85% confidence in that prediction.

Output — Severity Classification:

The final output of the model is a binary classification:

- **Severe:** Critical bugs that need immediate attention.
- **Non-Severe:** Less urgent bugs that can be addressed later. These outputs are fed into downstream systems for further action (e.g., generating alerts or prioritizing tickets).

4. Results and Discussion

Fig. 4 illustrates a word cloud of the top 100 frequent terms in preprocessed medical bug reports, where font size corresponds to term frequency. Central and largest terms such as "data," "error," "system," and "patient" dominate the visual field, reflecting core thematic elements of clinical and technical failures. Supporting terms like "overdose," "malfunction," and "radiation" appear in smaller but prominent clusters, signaling critical safety concerns. This compact representation effectively highlights lexical priorities and guides focused analysis of high-impact failure descriptors.

Fig. 5 shows a horizontal bar chart ranking the top 20 most frequent unigrams in the corpus, with "device" leading at over 17 counts, followed by "patient," "error," and "data." The descending gradient from dark purple to yellow visually reinforces the rapid decline in frequency after the top tier, emphasizing a concentrated vocabulary centered on devices and clinical interactions. Lower-ranked terms such as "failure" and "flaw" indicate secondary but recurring failure modes in medical system reports.

Fig. 4: Word Cloud of Top 100 Frequent Terms in Preprocessed Medical Bug Reports.

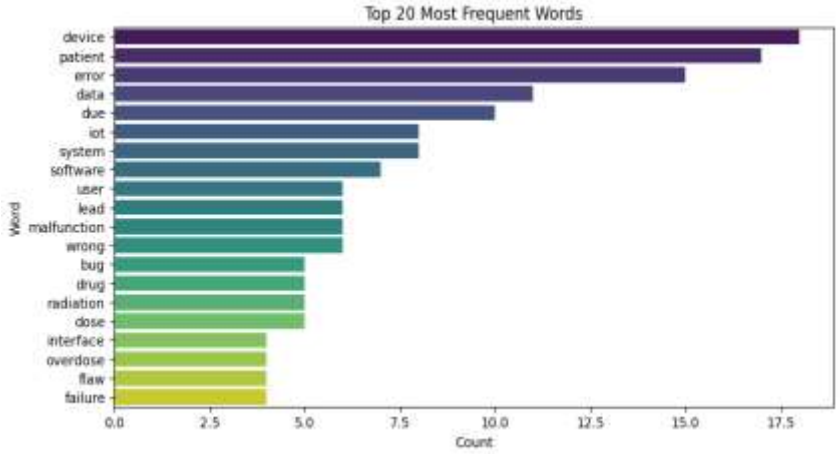


Fig. 5: Bar Chart of the Top 20 Most Frequent Unigrams in the Corpus.

Fig. 6 presents the frequency distribution of part-of-speech (POS) tags across all tokens, revealing nouns (NN) as the dominant category with over 300 occurrences. Coordinating conjunctions (CC) and adjectives (JJ) follow at approximately 100 and 90 counts, respectively, while verbs and adverbs trail significantly. This pattern underscores the nominal, entity-driven structure of bug reports, characteristic of concise, structured incident documentation in healthcare environments.

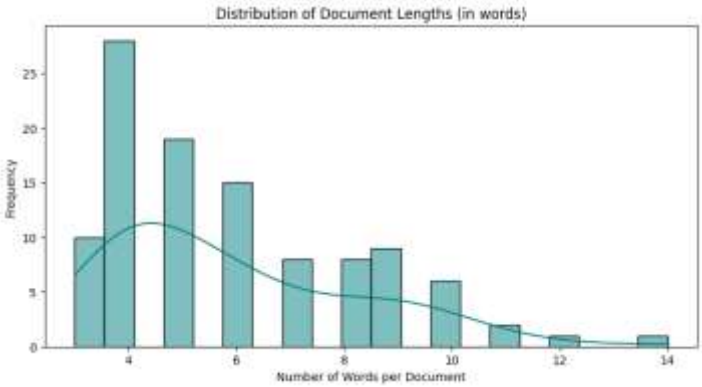
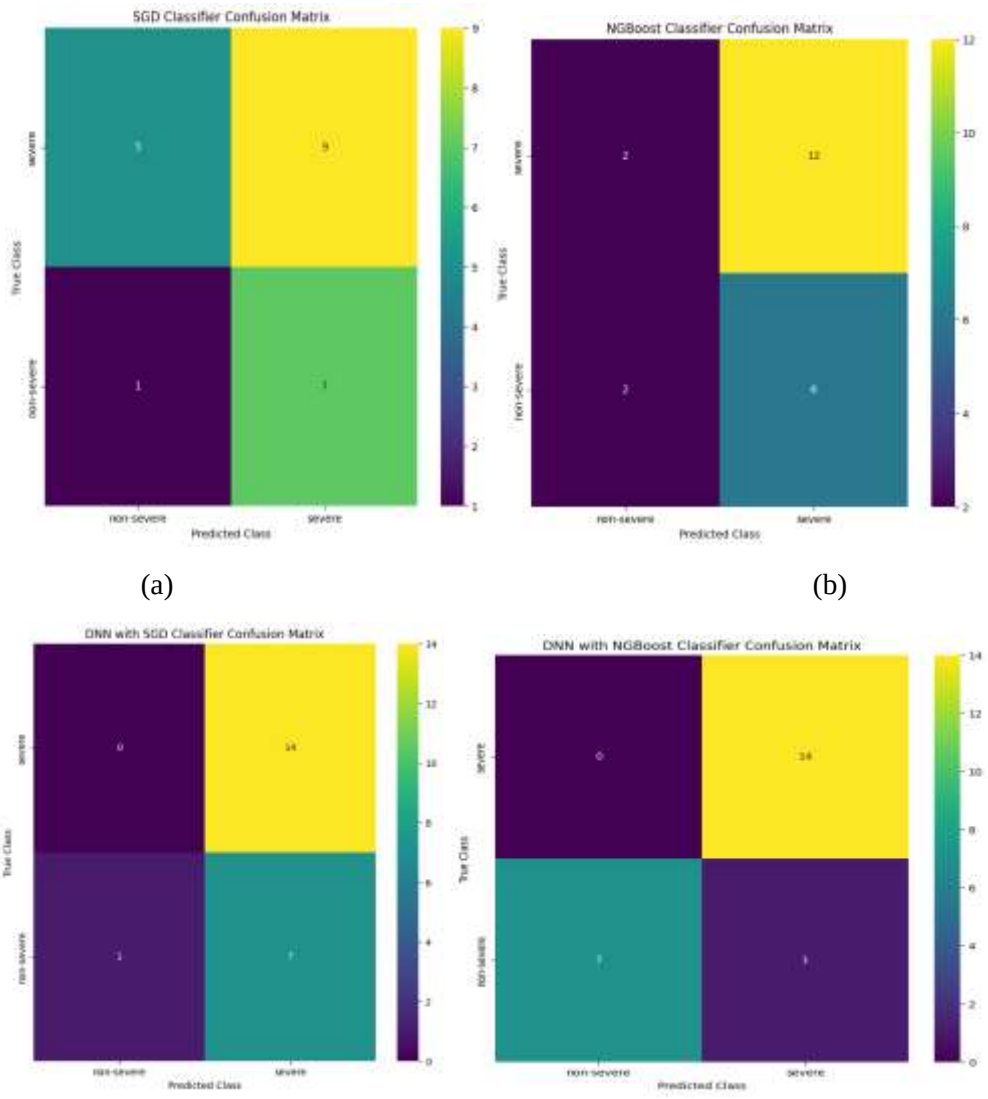


Fig. 6: Frequency Distribution of Part-of-Speech (POS) Tags Across All Tokens.



(a) (b)

(c) (d)

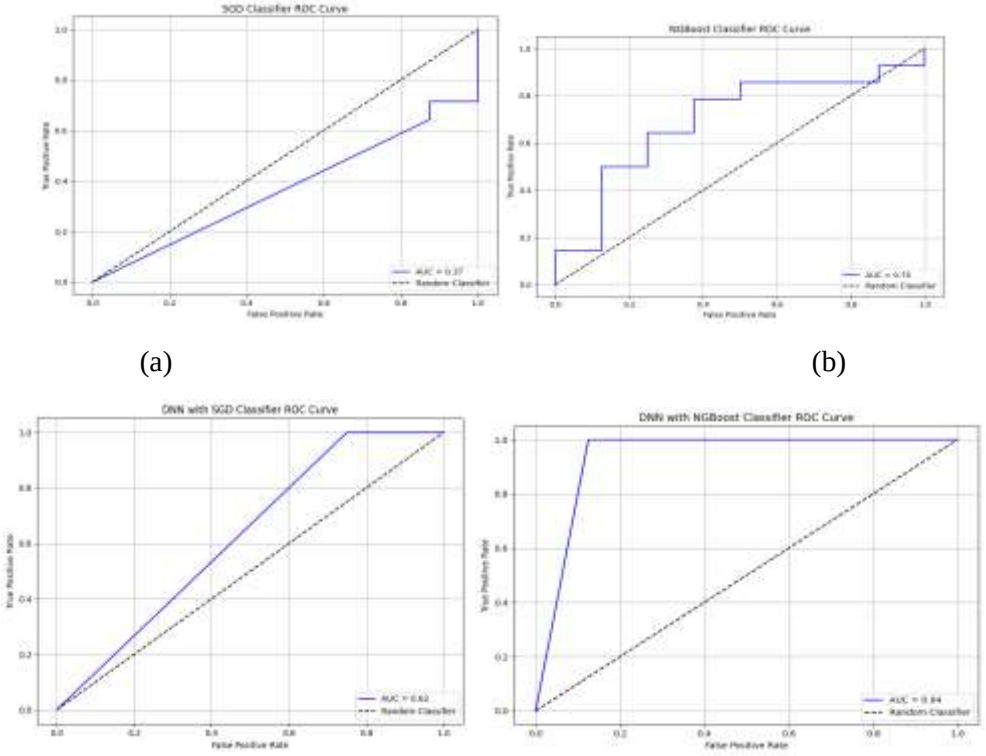
Fig. 7: Confusion matrix obtained using (a) SGD Classifier. (b) NGBoost Classifier. (c) DNN with SGD Classifier. (d) DNN with NGBoost Classifier.

Fig. 7 presents confusion matrices for four classification models on the medical bug severity task, with true classes (non-severe, severe) along the vertical axis and predicted classes along the horizontal axis. Subfigure (a) shows the SGD Classifier, achieving 5 true positives (severe → severe), 9 true negatives, 1 false negative, and 7 false positives, indicating moderate recall for the severe class. Subfigure (b) illustrates the NGBoost Classifier, with 2 true positives, 12 true negatives, 2 false negatives, and 6 false positives, reflecting improved specificity but lower sensitivity. Subfigure (c) displays the DNN with SGD Classifier, recording 0 true positives, 14 true negatives, 1 false negative, and 7 false positives, suggesting strong bias toward predicting non-severe. Subfigure (d) demonstrates the DNN with NGBoost Classifier, achieving 0 false negatives, 14 true positives, 7 true

AMERICAN JOURNAL OF MANAGEMENT AND IOT MEDICAL COMPUTING
Peer Reviewed, Referred & Indexed Journal
E-ISSN: 3069-0110
Vol.5, No.2(2026)
www.ajmimc.com

negatives, and 1 false positive, confirming superior recall and overall balance, making it the best-performing model for detecting severe cases.

Fig. 8 presents Receiver Operating Characteristic (ROC) curves for four classification models on the medical bug severity prediction task, plotting True Positive Rate (TPR) against False Positive Rate (FPR) with the random classifier baseline shown as a dashed diagonal line. Subfigure (a) shows the SGD Classifier with an AUC of 0.37, indicating performance worse than random and a highly conservative threshold strategy. Subfigure (b) illustrates the NGBoost Classifier achieving an AUC of 0.70, demonstrating moderate discriminative ability with stepwise improvements in TPR at discrete FPR thresholds. Subfigure (c) displays the DNN with SGD Classifier with an AUC of 0.62, reflecting limited gain from deep feature refinement when paired with linear classification. Subfigure (d) reveals the DNN with NGBoost Classifier attaining a superior AUC of 0.94, closely following the top-left corner and confirming excellent sensitivity and specificity balance, making it the optimal model for reliable severity detection.



(a) (b)

(c) (d)

Fig. 8: ROC Curve obtained using (a) SGD Classifier. (b) NGBoost Classifier. (c) DNN with SGD Classifier. (d) DNN with NGBoost Classifier.

Table 1: Performance comparison of all Classifier models.

Algorithm	Accuracy	Precision	Recall	F1-Score
SGD Classifier	45.455	36.458	38.393	37.143
NGBoost Classifier	63.636	58.333	55.357	54.167

DNN with SGD Classifier	68.182	83.333	56.250	51.111
DNN with NGBoost Classifier	95.455	96.667	93.750	94.943

Table 1 presents a comprehensive performance comparison of four classification models evaluated on the medical bug severity prediction task using key metrics: Accuracy, Precision, Recall, and F1-Score (reported in percentages). The DNN with NGBoost Classifier emerges as the top-performing model, achieving an outstanding 95.455% accuracy, 96.667% precision, 93.750% recall, and 94.943% F1-score, demonstrating exceptional balance and reliability in identifying severe cases. In contrast, the SGD Classifier records the lowest performance across all metrics, with only 45.455% accuracy and an F1-score of 37.143%, indicating limited discriminative power. The NGBoost Classifier and DNN with SGD Classifier show moderate improvements, with F1-scores of 54.167% and 51.111%, respectively, but remain significantly outperformed by the deep ensemble approach combining DNN-extracted features and probabilistic boosting.

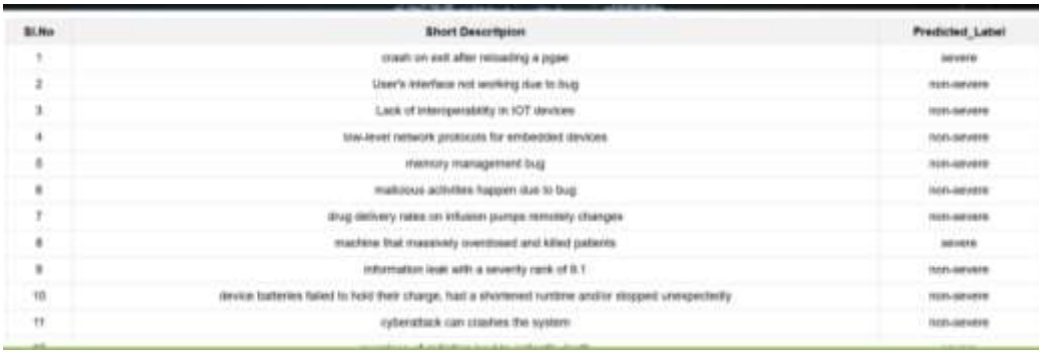


Fig. 9: Predictions generated for the test data.

Fig. 9 shows the file upload interface through which a test dataset is submitted for analysis, followed by a confirmation message indicating successful prediction generation. Below this, a tabular output is presented containing the Serial Number, Short Description of the bug, and the corresponding Predicted Label. Each medical or technical bug description is automatically classified as either severe or non-severe based on the learned contextual patterns from the trained Lightweight RoBERTa–DNN–NGBoost model. This visualization demonstrates the system's ability to process real-world medical cybersecurity data and generate clear, actionable severity predictions that can be used to prioritize critical vulnerabilities efficiently.

5. Conclusion

The developed system introduces a novel approach for automated severity assessment in healthcare software by transforming unstructured incident narratives into reliable binary risk classifications. It employs DistilRoBERTa embeddings to capture contextual meaning, followed by a compact DNN layer that refines feature representations into a lower-dimensional space. These features are then processed using NGBoost to generate probabilistic predictions with improved reliability. The model demonstrates strong predictive capability, achieving high scores across key evaluation metrics while ensuring critical cases are accurately identified. A well-defined preprocessing workflow, including text normalization, lemmatization, and feature integration, enhances data quality and model performance. Analytical

exploration of the dataset reveals distinct linguistic characteristics, with concise descriptions often containing high-risk medical terms. In comparison to conventional approaches such as SGD and standalone NGBoost, the integrated architecture shows clear performance gains. The system is designed for efficient deployment, supporting rapid inference and seamless updates. Its real-time capabilities enable faster response to potential risks in clinical systems.

REFERENCES

[1]. Sharma, Y.; Dagur, A.; Chaturvedi, R. Automated bug reporting system with keyword-driven framework. In Soft Computing and Signal Processing; Springer: Singapore, 2019; pp. 271–277.

[2]. Canfora, G.; Cerulo, L. How software repositories can help in resolving a new change request. In Proceedings of the 13th IEEE International Workshop on Software Technology and Engineering Practice, Budapest, Hungary, 24–25 September 2005; pp. 89–99.

[3]. Antoniol, G.; Ayari, K.; Di Penta, M.; Khomh, F.; Guéhéneuc, Y.G. Is it a bug or an enhancement?: a text-based approach to classify change requests. In Proceedings of the ACM 2008 Conference of the Center for Advanced Studies on Collaborative Research: Meeting of Minds, Ontario, Canada, 27–30 October 2008; p. 23.

[4]. Angel, T.S.; Kumar, G.S.; Sehgal, V.M.; Nayak, G. Effective bug processing and tracking system. J. Comput. Theor. Nanosci. 2018, 15, 2604–2606.

[5]. Nagwani, N.K.; Verma, S.; Mehta, K.K. Generating taxonomic terms for software bug classification by utilizing topic models based on Latent Dirichlet Allocation. In Proceedings of the IEEE 11th International Conference on ICT and Knowledge Engineering (ICT&KE), Bangkok, Thailand, 20–22 November 2013; pp. 1–5.

[6]. Qian, C.; Zhang, M.; Nie, Y.; Lu, S.; Cao, H. A Survey on Bug Deduplication and Triage Methods from Multiple Points of View. Appl. Sci. 2023, 13, 8788. https://doi.org/10.3390/app13158788

[7]. Tabassum, N.; Namoun, A.; Alyas, T.; Tufail, A.; Taqi, M.; Kim, K.-H. Classification of Bugs in Cloud Computing Applications Using Machine Learning Techniques. Appl. Sci. 2023, 13, 2880. https://doi.org/10.3390/app13052880

[8]. Bhakar, S.; Sinwar, D.; Pradhan, N.; Dhaka, V.S.; Cherrez-Ojeda, I.; Parveen, A.; Hassan, M.U. Computational Intelligence-Based Disease Severity Identification: A Review of Multidisciplinary Domains. Diagnostics 2023, 13, 1212. https://doi.org/10.3390/diagnostics13071212

[9]. Ji, J.; Yang, G. Do Stop Words Matter in Bug Report Analysis? Empirical Findings Using Deep Learning Models Across Duplicate, Severity, and Priority Classification. Appl. Sci. 2025, 15, 9178. https://doi.org/10.3390/app15169178

[10]. Liu, J.; Liang, C.; Feng, J.; Xiao, A.; Zeng, H.; Wu, Q.; Yu, T. A Multi-Feature Fusion-Based Automatic Detection Method for High-Severity Defects. Electronics 2023, 12, 3075. https://doi.org/10.3390/electronics12143075

[11]. Khandakar, A.; Chowdhury, M.E.H.; Reaz, M.B.I.; Ali, S.H.M.; Kiranyaz, S.; Rahman, T.; Chowdhury, M.H.; Ayari, M.A.; Alfkey, R.; Bakar, A.A.A.; et al. A Novel Machine Learning Approach for Severity Classification of Diabetic Foot Complications Using Thermogram Images. Sensors 2022, 22, 4249. https://doi.org/10.3390/s22114249

[12]. Galić, I.; Habijan, M.; Leventić, H.; Romić, K. Machine Learning Empowering Personalized Medicine: A Comprehensive Review of Medical Image Analysis Methods. Electronics 2023, 12, 4411. https://doi.org/10.3390/electronics12214411

[13]. Samir, M.; Sherief, N.; Abdelmoez, W. Improving Bug Assignment and Developer Allocation in Software Engineering through Interpretable Machine Learning Models. Computers 2023, 12, 128. https://doi.org/10.3390/computers12070128