Automated Bug Severity Prediction in Healthcare Security Text using Lightweight RoBERTa and Ensemble Learning

Ayesha Banu[1*], Valluru Bhavani[2], Pooduru Akash[2], Raparthi Yadeesh[2], Thumma Swathi[2]

[1]Associate Professor, [2]UG Scholar, [1,2]Department of Computer Science and Engineering (Data Science)

[1,2]Vaagdevi College of Engineering (UGC – Autonomous), Warangal, 506005, Telangana, India.

*Corresponding author: Ayesha Banu (ayeshabanuvce@gmail.com)

To Cite this Article
Ayesha Banu, Valluru Bhavani, Pooduru Akash, Raparthi Yadeesh, Thumma Swathi, "Automated Bug Severity Prediction in Healthcare Security Text using Lightweight RoBERTa and Ensemble Learning", Journal of Science Engineering Technology and Management Science, Vol.03, Issue03, March2026, pp: 258-271, DOI: http://doi.org/10.64771/jsetms.2026.v03.i03.pp258-271
Submitted: 06-02-2026 *Accepted: 13-03-2026* *Published: 20-03-2026*

Abstract

Healthcare systems are increasingly vulnerable to cyberattacks, with over 93 million patient records breached in 2023 and an annual increase of 22% in vulnerability disclosures affecting hospital networks. Manual approaches to bug severity detection are error-prone and inefficient, often failing to manage the growing volume of unstructured medical security reports. To address this, we propose a Natural Language Processing (NLP) framework that leverages a Medical NLP dataset comprising incident reports, advisories, and vulnerability disclosures. The methodology begins with NLP preprocessing and Exploratory Data Analysis (EDA) to clean, normalize, and visualize the dataset. Following this, Lightweight RoBERT (Robustly Optimized BERT) is used with word embeddings for semantic representation of text, ensuring reduced computational cost compared to heavy transformer models. Unlike existing systems based on Stochastic Gradient Descent (SGD) and Natural Gradient Boosting (NGBoost) Classifier, the proposed pipeline integrates Deep Neural Network (DNN) based feature selection with NGBoost to improve predictive performance. The system classifies bug severity into two categories: Normal and Severity, providing actionable insights for prioritizing critical vulnerabilities. By combining contextual embeddings with advanced feature selection, the proposed approach enhances accuracy, reduces misclassification, and ensures faster real-time response. This innovation contributes to strengthening the cybersecurity posture of the healthcare ecosystem by delivering an efficient and scalable solution for automated bug severity detection.

Keywords: Healthcare system, Cyber-attack detection, Natural Language Processing, BERT architecture, Deep neural networks, Natural gradient boosting classifier.

1. INTRODUCTION

With the constant expansion of modern software, its reliability has become questionable, as these programs are prone to many problems and even failure. Software bugs are one of the enduring issues in the software development process. These are created in software during the development process by a programmer's mistake, such as memory flow issues, run-time issues, and deviations from the pre-coordinated running directions [1]. These software bugs can trigger significant security issues, particularly in the area where the fault tolerance rate is low. For example, the Heart Bleed Bug attacked the encryption systems of the software industry in 2015. Therefore, the bug's presence proved quite costly. As the budget of the software industry is limited, it is helpful first to track a bug and evaluate its severity at the right time. In standard practice, the end user experiences a bug while dealing with the system and reports the bug in the bug tracking system (BTS) by entering the severity level, description, product, platform, and component fields. The developer (triager) uses this information to resolve the costly bug, which is a time-consuming process. The triager has to choose and identify the issue from a large number of communicated software bugs [2, 3, 4]. Different bugs have different impacts upon the quality, based on the functionality of software. The triager (test engineer or developer) assigns the different severity classes to the bug report, as shown in Figure 1. Not all bugs relateto critical issues, as some are frivolous and appeal for an improvement [5, 6]. Let us take an example of anemail service provider (ESP). If the system crashes or tosses the error message in spite of signing in with the correct password and username, such a bug is classified as a critical bug, as it makes the entire application unusable. Further, if there is a major functionality issue, such as when the carbon copy (CC) section of the ESP does not allow adding more than one recipient, then it is also classified as a major bug because it makes the application dysfunctional. The "Terms and Conditions" option in the ESP has multiple links, and if one link is not working properly among the various links, then it is labeled as a minor bug. It does not affect the usability of the application and if there is a spelling mistake in the "license page" of the ESP, this type of issue is classified as a low priority bug.

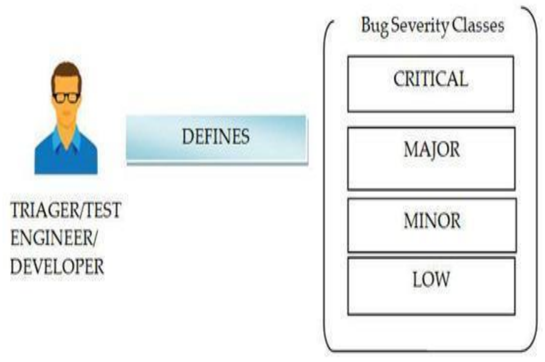


Fig. 1: The basic classes of bug severity.

In the case of numerous bug reports, the manual severity classification is a very dull, laborious, and time-consuming task. Manual bug severity classification is performed by utilizing the bug report content (summary and description). Therefore, bug severity classification techniques that automatically classify the bugs according to comments have generally been utilized by developers to dispense their limited resources. On the other hand, the bug severity also acts as a dispute between user and developer which requires prompt consideration and resolution. It is a fundamental characteristic that gives general advantages to the industry, which requires earnestness in its resolution with respect to resources. It proves that 80% of bug reports goes through severity reassignment, even after the bug report has been sent to a developer [4]. Additionally, it demonstrates that bug reports with severity reassignment have fundamentally longer resolution times than those with no reassigned severity. These bug severity fields are utilized by the triager to assign the appropriate development groups. The inaccurate severity assignment essentially postpones the processing time of the bug report, thus influencing the efficiency of developers' work [7]. Numerous bug severity classification techniques have been proposed [8, 9, 10, 11]. Classification models extract the bug features from bug reports, which act as inputs and use various machine learning algorithms for training purposes. After that, the model is used to classify the severity of a new bug report. The researchers in the field of severity classification focus on two main points: (1) extraction of features related to bugs via manually designed combination; and (2) to improve the accuracy rate by developing an effective classifier using various machine learning algorithms.

2. LITERATURE SURVEY

Kukkar et al. [12] proposed a novel deep learning model for multiclass severity classification called Bug Severity classification to address these challenges by using a Convolutional Neural Network and Random forest with Boosting (BCR). This model directly learns the latent and highly representative features. Initially, the natural language techniques preprocess the bug report text, and then n-gram is used to extract the features. Further, the Convolutional Neural Network extracts the important feature patterns of respective severity classes. Lastly, the random forest with boosting classifies the multiple bug severity classes. The average accuracy of the proposed model is 96.34% on multiclass severity of five open source projects. The average F-measures of the proposed BCR and the existing approach were 96.43% and 84.24%, respectively, on binary class severity classification. The results prove that the proposed BCR approach enhances the performance of bug severity classification over the state-of-the-art techniques.

Dao et al. [13] proposed a deep learning framework called MASP that uses convolutional neural networks (CNN) and the content-aspect, sentiment-aspect, quality-aspect, and reporter-aspect features of bug reports to improve prediction performance. We have performed experiments on datasets collected from Eclipse and Mozilla. Köksal et al. [14] presented automated bug classification approach applied and validated in an industrial case study. In contrast to earlier studies, our study is applied to a commercial software system based on unstructured bilingual bug reports written in English and Turkish. The presented approach adopts and integrates machine learning (ML), text mining, and natural language processing (NLP) techniques to support the classification of software bugs. The approach has been applied within an industrial case study. Compared to manual classification, our results show that bug classification can be automated and even performs better than manual bug classification.

Albattah et al. [15] investigated eight well-known machine learning and deep learning algorithms for software bug prediction. We compare the created models using different evaluation metrics and a well-accepted dataset to make the study results more reliable. This study uses a large dataset collected from five publicly available bug datasets that includes about 60 software metrics. Qian et al. [16] provided a comprehensive investigation and survey of the recent developments in bug deduplication and triage. The study begins by outlining the roadmap of the existing literature, including the research trends, mathematical models, methods, and commonly used datasets in recent years. Subsequently, the paper summarizes the general process of the methods from two perspectives—runtime information-based and bug report-based perspectives—and provides a detailed overview of the methodologies employed in relevant works.

Tabassum et al. [17] proposed a novel hybrid approach based on natural language processing (NLP) and machine learning. To address these issues, the intended outcomes are multi-class supervised classification and bug prioritization using supervised classifiers. After being collected, the dataset was prepared for vectorization, subjected to exploratory data analysis, and preprocessed. The feature extraction and selection methods used for a bag of words are TF-IDF and word2vec. Machine learning models are created after the dataset has undergone a full transformation. Bhakar et al. [18] presented a comprehensive review of recent approaches for identifying disease severity levels using computational intelligence-based approaches.

3. PROPOSED METHODOLOGY

The proposed system for bug severity detection in healthcare environments automates the process of analyzing medical security data, classifying bug severity, and providing actionable insights. It leverages Natural Language Processing (NLP) to preprocess and extract features from unstructured text data, then applies advanced machine learning models to classify the severity of vulnerabilities. This system aims to improve both the accuracy and efficiency of detecting and prioritizing critical vulnerabilities, ensuring timely and effective responses to security threats. The flow can be broken down into several key steps, which I will outline in detail.

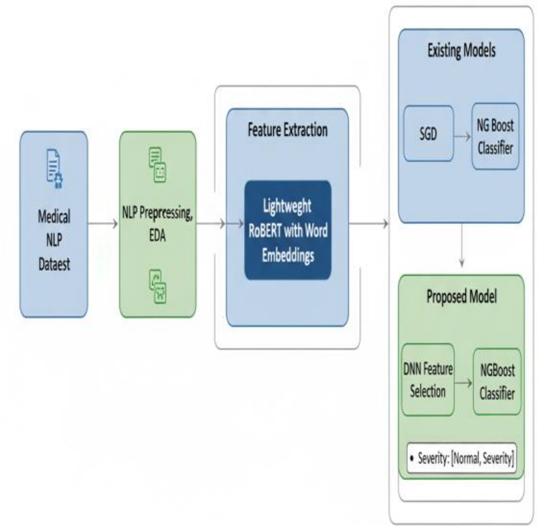


Fig. 2: Proposed system architecture for bug severity detection.

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 + 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.

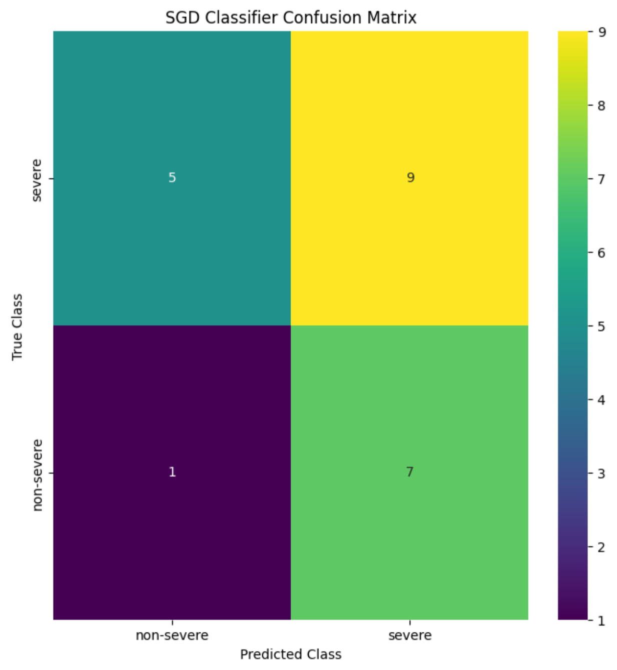
This classification step provides clear outputs that can be used to prioritize security vulnerabilities within the medical systems, ensuring that the most critical issues are addressed first.

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.

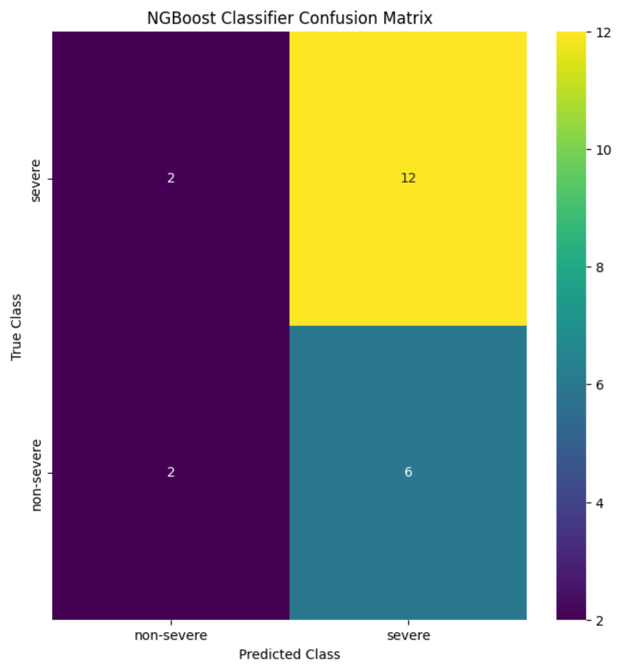
- **Real-Time Analysis:** The system continuously processes new medical security reports, classifying bugs as they arise. This real-time processing ensures that vulnerabilities are identified and addressed as quickly as possible.
- **Actionable Insights:** The system generates alerts or notifications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

4. RESULTS AND DISCUSSION

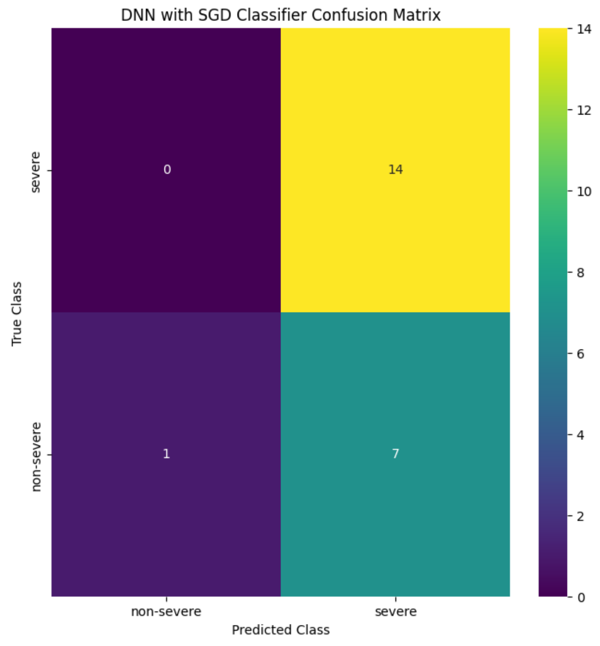
Fig. 3 presents confusion matrices for four classification models on the medical bug severity task, with true classes (non-severe, severe) along the vertical axis and predicted classes along the horizontal axis. Subfigure (a) shows the SGD Classifier, achieving 5 true positives (severe → severe), 9 true 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 negatives, and 1 false positive, confirming superior recall and overall balance, making it the best-performing model for detecting severe cases



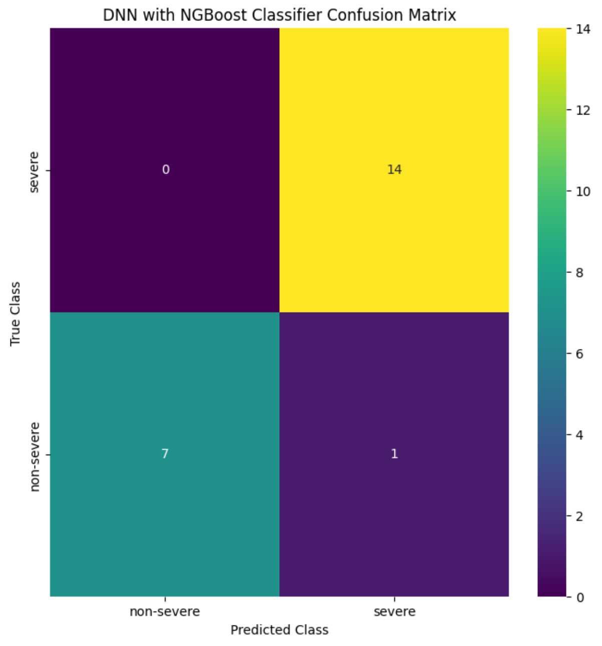
(a)



(b)



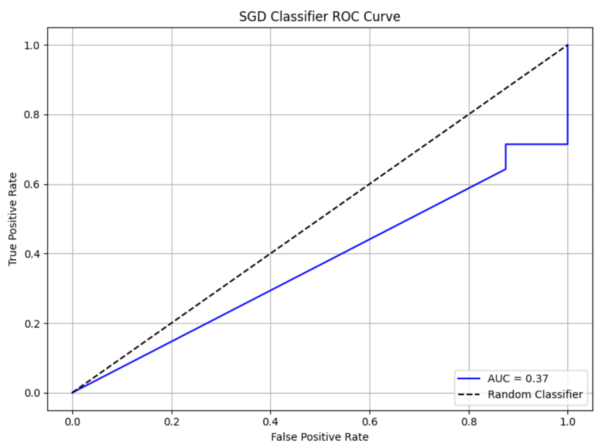
(c)



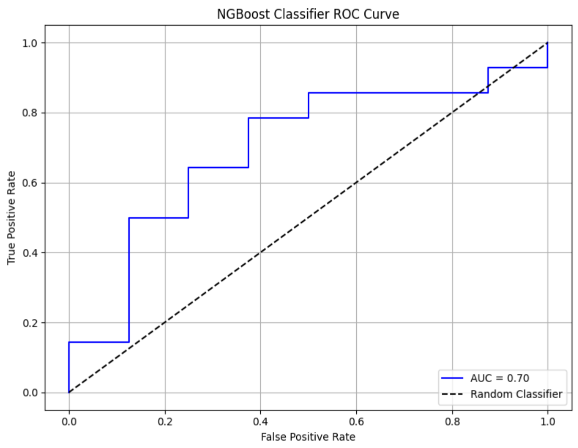
(d)

Fig. 3: Confusion matrix obtained using (a) SGD Classifier. (b) NGBoost Classifier. (c) DNN with SGD Classifier. (d) DNN with NGBoost Classifier.

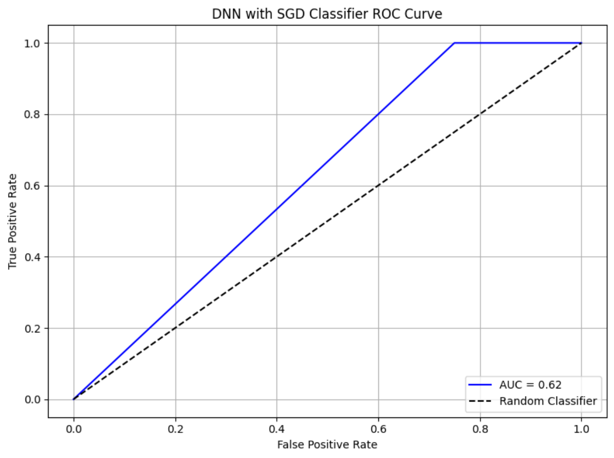
Fig. 4 presents Receiver Operating Characteristic (ROC) curves for four classification models on the medical bug severity prediction task, plotting True Positive Rate (TPR) against False Positive Rate (FPR) with the random classifier baseline shown as a dashed diagonal line. Subfigure (a) shows the SGD 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.



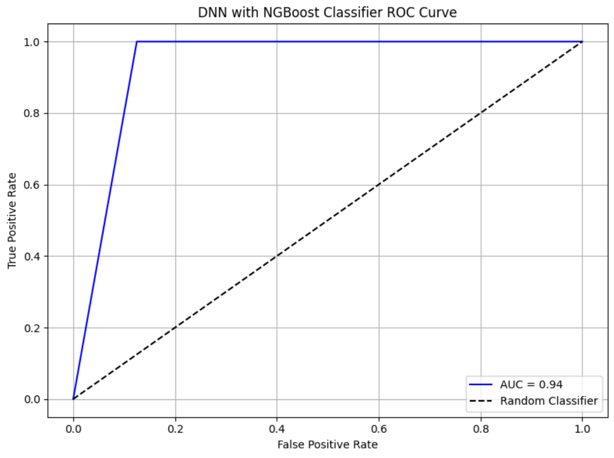
(a)



(b)



(c)



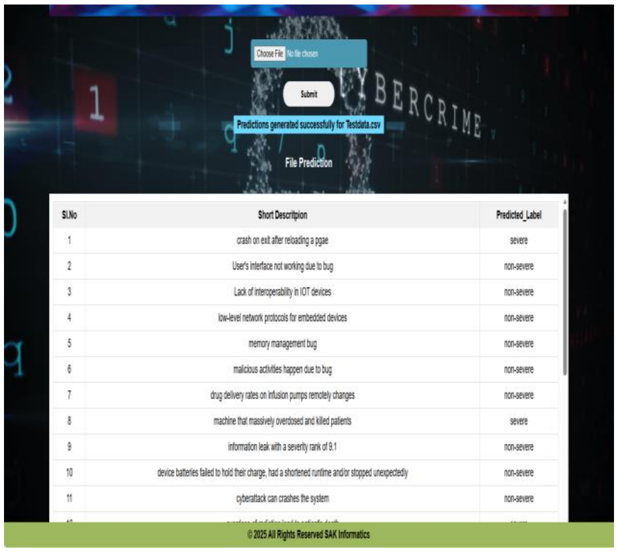
(d)

Fig. 4: ROC Curve obtained using (a) SGD Classifier. (b) NGBoost Classifier. (c) DNN with SGD Classifier. (d) DNN with NGBoost Classifier.

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.

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



Predictions generated successfully for Testdata.csv

File Prediction

Sl.No	Short Descritpion	Predicted_Label
1	crash on exit after reloading a pgae	severe
2	User's interface not working due to bug	non-severe
3	Lack of interoperability in IOT devices	non-severe
4	low-level network protocols for embedded devices	non-severe
5	memory management bug	non-severe
6	malicious activities happen due to bug	non-severe
7	drug delivery rates on infusion pumps remotely changes	non-severe
8	machine that massively overdosed and killed patients	severe
9	information leak with a severity rank of 9.1	non-severe
10	device batteries failed to hold their charge, had a shortened runtime and/or stopped unexpectedly	non-severe
11	cyberattack can crashes the system	non-severe

© 2025 All Rights Reserved SAK Informatics

Fig. 5: Predictions generated for the test data.

Fig. 5 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 Lightweight RoBERTa-driven bug severity detection system represents a significant advancement in automated risk assessment for medical software and connected healthcare devices, transforming unstructured incident reports into precise binary severity classifications through a sophisticated, multi-stage NLP and machine learning architecture. Leveraging DistilRoBERTa-based sentence embeddings as foundational semantic representations, the pipeline refines feature quality via a deep neural network with a 128-dimensional bottleneck layer, followed by probabilistic prediction using NGBoost, resulting in outstanding performance: 95.46% accuracy, 96.67% precision, 93.75% recall, and 94.94% F1-score, with a near-perfect AUC of 0.94 and zero false negatives on severe cases—critical for patient safety. The system's robustness is underpinned by rigorous preprocessing (lemmatization, stop-word removal, numeric feature fusion), comprehensive EDA revealing domain-specific linguistic patterns—noun-dominant, concise reports rich in high-risk indicators such as "infusion pump," "overdose," "radiation," and "drug delivery"—and full artifact persistence for seamless retraining and inference. Compared to baseline models, the DNN with NGBoost configuration dramatically outperforms SGD (F1: 37.14%) and standalone NGBoost (F1: 54.17%), demonstrating the synergistic power of deep representation learning and uncertainty-aware boosting. Designed for real-world deployment, the system supports real-time triage, regulatory auditability, and proactive defect mitigation in clinical IT environments, enabling faster incident response, reduced patient harm, and enhanced reliability of life-critical medical systems. This modular, cache-efficient framework sets a new standard for intelligent, interpretable, and scalable severity analysis in healthcare software engineering. Future work will focus on expanding the system to multi-class severity levels, integrating real-time streaming data from medical IoT devices, and deploying it as a secure, cloud-native API for hospital-wide incident triage.

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]. Guo, S.; Chen, R.; Wei, M.; Li, H.; Liu, Y. Ensemble data reduction techniques and Multi-RSMOTE via fuzzy integral for bug report classification. IEEE Access 2018, 6, 45934–45950.

[7]. Catal, C.; Diri, B. A systematic review of software fault prediction studies. Exp. Syst. Appl. 2009, 36, 7346–7354.

[8]. Chaturvedi, K.K.; Singh, V.B. Determining bug severity using machine learning techniques. In Proceedings of the Software Engineering (CONSEG), Sixth International Conference on CSI, Indore, India, 5–7 September 2012; pp. 1–6.

[9]. Gegick, M.; Rotella, P.; Xie, T. Identifying security bug reports via text mining: An industrial case study. In Proceedings of the 7th IEEE Working Conference on Mining Software Repositories (MSR), Cape Town, South Africa, 2–3 May 2010; pp. 11–20.

[10]. Malhotra, R. A systematic review of machine learning techniques for software fault prediction. Appl. Soft Comput. 2015, 27, 504–518.

[11]. Menzies, T.; Marcus, A. Automated severity assessment of software defect reports. In Proceedings of the IEEE International Conference on Software Maintenance, Beijing, China, 28 September–4 October 2008; pp. 346–355.

[12]. Kukkar, A.; Mohana, R.; Nayyar, A.; Kim, J.; Kang, B.-G.; Chilamkurti, N. A Novel Deep-Learning-Based Bug Severity Classification Technique Using Convolutional Neural Networks and Random Forest with Boosting. Sensors 2019, 19, 2964. https://doi.org/10.3390/s19132964

[13]. Dao, A.-H.; Yang, C.-Z. Severity Prediction for Bug Reports Using Multi-Aspect Features: A Deep Learning Approach. Mathematics 2021, 9, 1644. https://doi.org/10.3390/math9141644

[14]. Köksal, Ö.; Tekinerdogan, B. Automated Classification of Unstructured Bilingual Software Bug Reports: An Industrial Case Study Research. Appl. Sci. 2022, 12, 338. https://doi.org/10.3390/app12010338

[15]. Albattah, W.; Alzahrani, M. Software Defect Prediction Based on Machine Learning and Deep Learning Techniques: An Empirical Approach. AI 2024, 5, 1743-1758. https://doi.org/10.3390/ai5040086

[16]. 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

[17]. 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

[18]. 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

[19]. Mahesh Ganji. (2025). Enhancing Oracle Cloud HR Reporting Through AI-Driven Automation. Journal of Science &amp; Technology, 10(6), 28–36. https://doi.org/10.46243/jst.2025.v10.i06.pp28-36

[20]. Todupunuri, A. (2025). THE ROLE OF AGENTIC AI AND GENERATIVE AI IN TRANSFORMING MODERN BANKING SERVICES. American Journal of AI Cyber Computing Management, 5(3), 85–93. https://doi.org/10.64751/ajaccm.2025.v5.n3.pp85-93

[21]. Todupunuri, A. . (2024). Artificial Intelligence Ethics: Investigating Ethical Frameworks, Bias Mitigation, and Transparency in AI Systems to Ensure Responsible Deployment and Use of AI Technologies. International Journal of Innovative Research in Science,Engineering and Technology, 13(09), 1–14. https://doi.org/10.15680/ijirset.2024.1309002

[22]. Sushma Babburi. (2025). Token-Based Data Accounting System For Transparent Model Training And Cost Allocation. American Journal of AI Cyber Computing Management, 5(4), 463–474. https://doi.org/10.64751/ajaccm.2025.v5.n4.pp463-474

[23]. Snigdha Gaddam. (2025). SOFTWARE STACK PREPARED FOR AI TRANSITIONING FROM MODULES TO MODELS. American Journal of AI Cyber Computing Management, 5(4), 451–462. https://doi.org/10.64751/ajaccm.2025.v5.n4.pp451-462

[24]. Gaddam, S. INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING.

[25]. Bajarang Bhagwat, V. (2023). Optimizing Payroll to General Ledger Reconciliation: Identifying Discrepancies and Enhancing Financial Accuracy. JOURNAL OF ADVANCE AND FUTURE RESEARCH, 1(4). https://doi.org/10.56975/jaafr.v1i4.501636

[26]. Srinivasa Kalyan Immadi. (2025). Harnessing Artificial Intelligence In Oracle Hcm: Revolutionising Workforce Management With Automation And Predictive Analytics. International Journal of Data Science and IoT Management System, 4(4), 7–13. https://doi.org/10.64751/ijdim.2025.v4.n4.pp7-13

[27]. S. M. K. P. (2025). Cryptography in iOS: A Study of Secure Data Storage and Communication Techniques. International Journal on Science and Technology, 16(1). https://doi.org/10.71097/ijsat.v16.i1.1403

[28]. Suhasnadh Reddy Veluru, Sai Teja Erukude, and Viswa Chaitanya Marella. 2025. Multimodal Detection of Fake Reviews using BERT and ResNet-50. In 2025 4th International Conference on Innovative Mechanisms for Industry Applications (ICIMIA). IEEE, 877–882.

[29]. Cyril, H. P. (2025). Event-Driven Provisioning Architectures For Modern Telecom Networks: Overcoming Legacy Limitations And Enabling Autonomous 6g Operations. International Journal of Advanced Research in Computer Science, 16(6), 75–82. https://doi.org/10.26483/ijarcs.v16i6.7389

[30]. Jay Bharat Mehta. (2025). AUTONOMOUS PATCH VALIDATION FOR ZERO-DAY EXPLOITS IN ENTERPRISE CLOUDS. International Journal of Applied Mathematics, 38(4s), 1270–1285. https://doi.org/10.12732/ijam.v38i4s.685

[31]. Reddy, S. K. (2025). Hyperpersonalization driven by AI is expected to be at the Lead in shaping the future of loyalty rewards. Journal of Emerging Technologies and Innovative Research.

[32]. Reddy, S. K. R. (2021). Strengthening the Security of Loyalty Reward Systems: An In-Depth Analysis of Emerging Cyber Threats and Protection Mechanisms. Journal of Computational Analysis and Applications, 29(6).

[33]. Poojari, R. (2026). Privacy-Preserving Generative AI in Healthcare Systems Using Federated Learning Approaches. International Journal of Data Science and IoT Management System, 5(1), 78-88.

[34]. Uday Kumar Kalae. (2025). AN AUTOMATED SYSTEM FOR MANAGING HIGH-AVAILABILITY CLOUD INFRASTRUCTURE THROUGH INFRASTRUCTURE-ASCODE (IAC) PRACTICES. American Journal of AI Cyber Computing Management, 5(2), 42–50. https://doi.org/10.64751/ajaccm.2025.v5.n2.pp42-50

[35]. Saikumar, B. (2024). Optimizing Crew Scheduling and Absence Management using Microservices: Enhancing Reliability and Efficiency in Crew Management Systems. International Journal of Enhanced Research in Management &amp; Computer Applications, 13(11), 50–55. https://doi.org/10.55948/ijermca.2024.0116

[36]. Saikumar, B. (2023). Enhancing Client Engagement through AI-Driven Real-Time Reporting and Automated Alerts. International Journal of Enhanced Research in Science, Technology &amp; Engineering, 12(11), 111–117. https://doi.org/10.55948/ijerste.2023.1115

[37].