

A HYBRID DEEP LEARNING APPROACH FOR ACCURATE SECURITY REQUIREMENTS CLASSIFICATION USING SEMANTIC AND STATISTICAL FEATURE INTEGRATION

¹Madina Ali, ^{*2}Hashim Ali, ³Didar Hussain, ⁴Abdullah Sadiq

¹Department of Computer Science, Abdul Wali Khan University Mardan

^{*2}Department of Computer Science, Abdul Wali Khan University Mardan

³Department of Computer Science, Abdul Wali Khan University Mardan & Department of Computer Science, Bacha Khan University Charsadda

⁴Department of Computer Science, Bacha Khan University Charsadda

DOI: <https://doi.org/10.5281/zenodo.21777251>

Keywords

Security, Requirement Classification, Requirements Engineering, Convolutional Neural Network, XGBoost, Deep Learning, Software Engineering

Article History

Received on 02 Jan, 2026

Accepted on 14 Feb, 2026

Published on 15 Feb, 2026

Copyright @Author

Corresponding Author:

Hashim Ali

hashimali@awkum.edu.pk

Abstract

Security requirements classification is a fundamental task in requirements engineering, enabling software engineers to discover security-related requirements during the early stages of the SDLC. In large software development projects, the identification of Security Requirements is time-consuming, and error-prone. In this regard, this research aims to address this challenge by proposing a hybrid model of CNN and XGBoost for automated binary classification of Security and Non-Security Requirements. The proposed framework uses a neural network for deep feature refinement, and the final classification is performed using XGBoost. To capture the semantic information and the statistical information of the text, SBERT embeddings are combined with TF-IDF representations before feature refinement. The proposed framework is assessed with the publicly available PROMISE dataset by analyzing its accuracy, precision, recall, F1-score and AUC-ROC. Experimental results show that the proposed model brings 93% accuracy, 84% precision, 88% recall, 86% F1-score and 99% AUC-ROC, outperforming previous state-of-the-art approaches for security requirements classification. The results show that the proposed hybrid framework is an effective and reliable solution for automated security requirements classification and reduces manual effort and supports intelligent software requirements engineering.

1. Introduction

Software requirements are a crucial part of the software development life cycle (SDLC). It describes what and how the system should do. There are two types of requirements, such as FRs and NFRs. FRs describe what the system should do, and NFRs describe how the system should do. Requirements are so important that they not only fulfill what the system they also focus on the limitations of the system [1]. As requirements are crucial and mostly affect the system, there is a pressing need to identify the requirements accurately [2, 3]. For the identification and classification of the requirements, manual techniques like TF-IDF and CountVectorizer are used, whereas for automated AI-based approaches are used [4]. Although all of these approaches generated promising results, there are still some issues, such as architecture complexity, dataset limitation, imbalancing, human error, as well as non-scalability. Therefore, these approaches became insufficient with the passage of time. When AI started, they solved all the above issues and defined how software requirements are processed. Machine Learning (ML) and Deep Learning (DL) are two main branches of AI. Many successful research articles focus on ML models for requirements classification and perform well compared to traditional techniques [5]. AI has increased performance in such tasks utilizing ELMo, FastText, and transformer-based approaches [6, 7]. Among these approaches, one is BERT, which has shown outstanding performance in different RE tasks, such as ambiguity resolution, keyword extraction, and requirement classification [8, 9]. Another popular AI-based technique, Large Language Models (LLMs), has shown good performance in various RE subtasks, such as wording recognition and traceability automation [10]. Moreover, these AI-based techniques also performed well on the unlabelled datasets.

Nowadays, AI is used in more fields of software engineering, such as code generation, debugging, testing, and deployment. Several AI-based tools like GPT-4 and OpenAI Codex are available, which are used for code generation from natural language. They successfully reduced human errors and development time [11]. AI-driven testing keys also improve software program reliability and efficiency by enhancing the accuracy of issue and exposure predictions. Despite these refinements, historical SRE research has tended to overlook more recent AI-driven methods in favor of more traditional prioritization or categorization techniques. There is also a dearth of research on the combined use of ML, DL, and LLM-based methods for requirements classification [12, 13].

Due to the recent advancements in the field of deep learning, security requirements engineering is becoming different. Hybrid architectures include neural networks and ensemble in order to demonstrate how modern DL methods can detect more complex patterns in requirement texts so that the classification of security requirements can be more accurate and automated [6]. Based on this strategy, our work will be a further improvement of the hybrid DL framework, adapted specifically to the classification of security requirements with an aim of obtaining more precise and trustful automated software requirement analysis.

2. Related Work

Cuevas et al. [14] proposed an ML-based binary classifier to identify whether user stories can be implemented with AI; they are helpful in supporting Smart Product Backlogs. With the help of the Mendeley data, the authors ran LR, KNN, and SVM using TF-IDF and word/character embeddings. The highest accuracy (74-75%), was obtained with Logistic Regression and SVM, which confirms the effectiveness of the model. Muhamad et al. [15] compared the class balancing methods of requirement classification on the PROMISE dataset using Bag of Words features with SVM and different resampling algorithms that included SMOTE, Borderline-SMOTE, and SVM-SMOTE. The findings indicated that both accuracy and F1-score of SVM-SMOTE were much better than those of the baseline. Kaur et al. [18] proposed a transformer-based requirement classification model using BERT embeddings on the PROMISE dataset. This model successfully reduces misclassification and correctly classifies functional and non-functional requirements.

Rahman et al. [16] applied 15 ML models on the PROMISE_exp dataset with different vectorization approaches, including Chi-Square, BoW, Hashing, and TF-IDF for NFR classification. In these models, linear SVC with TF-IDF gained the highest F1-score of 81.5% and outperformed other models. Saleem et al. [19] applied a multi-label-based model, MLRPredictor, on the PROMISE dataset using OkapiBM and LR. Their proposed approach performs better than other ML and DL models in terms of f-measure, and hence they proved strong generalization. Hajjami et al. [20] performed comparison of two versions of ChatGPT i.e. gpt-3.5-

turbo and gpt-4 with LLMs and SVM models by using five popular datasets such as under zero-shot, few-shot modes and PROMISE. Their results show that LLMs performed better than other approaches in terms of generalization in few short but overall. Jamasb et al. [21] applied ML-based models for requirement classification using PROMISE dataset to make correct decisions. The model performed well; however, it needs to be properly tuned or use a hybrid approach for further improvement. For the automated classification of NFRs, Souza et al. [22] utilized different ML models such as NB, DT, KNN, and SVM. They used TF-IDF for feature extraction and evaluated these models on the PROMISEs dataset. In these models, SVM achieved the highest performance and outperformed other approaches. Abbas et al. [23] introduced an improved RF-based model, FNRC, for requirements classification. They applied this approach to CCHIT and PROMISE datasets and for performance optimization. This approach achieved a recall score of 85% and performed better than the traditional RF approach.

Canedo et al. [24] applied different ML models such as KNN, SVM, LR, and MNB on PROMISE_exp dataset for requirements classification. With the feature selection techniques, they used GloVe and BoW embeddings, which achieved an outstanding binary precision of 92.9%, and hence successfully improved results. Hassan et al. [6] proposed a hybrid CNN+XGBoost model for the classification of security and non-security requirements. They used two popular datasets, PROMISE_exp and DOSSPRE, for this task. They achieved an accuracy of 87.3% on PROMISE_exp and 85.3% on the DOSSPRE dataset. Lunarejo et al. [25] performed a comparison of different ML and DL models for NFR subtypes classification. They applied SVM with bag of words and fast text on a Spanish-translated version of the PROMISE NFR dataset. After applying SVM with both approaches, they proved that SVM with fast text achieved the highest performance due to its simplest architecture, and it needed lightweight processing resources. Minina et al. [26] worked on the classification of security requirements in GitHub issues. They applied SmallBERT techniques on PROMISE dataset for accomplishing the said task. Their results show that the proposed model achieved highest accuracy score on GitHub data. For bug prediction in Nasa and PROMISE dataset, Haq et al. [27] implemented a hybrid approach of transfer learning and an ensemble based model. Their approach gained an outstanding AUCROC score compared to other traditional models.

In another study Patel et al. [28] evaluated a UML-based approach on PROMISE dataset for NFRs recognizing and specifying. They gained a precision score of 90.05% and hence they outperformed other models. One of the advantage of this model is to providing the facility of visualization of the NFRs on use cases diagrams. Similarly Izhar et al. [29] applied NLP with TFIDF and cosine similarity on PROMISE and EPR datasets obtained from industries for functional and non-functional requirements classification. Their proposed approach gained highest and correct performance score in this regard. Hassan et al. [30] performed non-functional requirements i.e. NFRs classification using the popular PROMISE_exp dataset. They proposed a BERT-based model for achieving the said aim. Their approach achieved an accuracy of 90%, which very categorically show that the proposed model outperformed other approaches at all in this field. Similarly in the study of Or et al. [31] the classification of FR and NFRs have been done. They applied LR with the SMOTE technique on the PROMISE dataset to complete this task. Their approach gained an accuracy of 76.16%, but they improved this score from 58.31%, which clearly show their contributions. Alhoshan et al. [32] applied LLM-based approaches, Bloom, Llama, and Gemma, on the PROMISE dataset for requirements classification. They performed a comparison of these approaches with other traditional approaches. Their obtained results show that the proposed approach performed better than other approaches.

Garcia et al. [33] performed NFRs classification using the PROMISE dataset. They applied a multi-label-based model in this regard. Their proposed approach successfully performed multi-classification with the highest score compared to other models. Xu et al. [34] proposed a BERT-based approach and its different versions for the classification of FRs and NFRs. They applied this approach to Tera, which is a variant of a very popular dataset PROMISE. The proposed BERT approach gained an 88.6% of accuracy. Their obtained results show that the proposed model outperformed other models. Martin et al. [35] performed requirements classification using two datasets, such as SecReq and PROMISE_exp datasets. All the requirements in the given datasets were security-related requirements. They applied an LLM-based model to successfully perform this task. Their proposed approach achieved the highest accuracy score in this regard. Similarly, in the work of Saleem et al. [36],

requirements classification has been done. They used three datasets, such as REQuestA, PROMISE, and Aerospace. The proposed Gemini and ChatGPT for various requirements engineering tasks. They performed well and struggled effectively; however, their approaches did not outperform other advanced models.

Silva et al. [37] used two datasets, the custom and PROMISE datasets, for clustering FRs in IOT settings. They applied the transformer-based model ReqCluster4IOT to accomplish this task. The proposed successfully performed the said task. Similarly, Ghaisas et al. [38] applied different approaches based on GenAI and NLP on the PROMISE_exp dataset. They successfully classified non-functional requirements, i.e., NFRs, into 12 classes effectively. They achieved the best results; however, the class-unbalancing issue had not resolved. Peer et al. [39] proposed a hybrid-based approach, NLP4ReF, for the classification of requirements as well as the generation of requirements. The proposed approach integrated different ML models based on NLTK with GPT 3.5. They evaluated the proposed approach on PROMISE as well as data obtained from different IoT projects. Their results show that GPT gained an outstanding performance for requirements generation and NLTK for classification. They achieved an 82.5% score over the generation of the requirements, whereas NLTK performed well for requirements classification. The given Table 2.1 summarizes the existing work discussed in the ML and DL sections.

Table 1: *Summary of the Existing Work*

Reference	Contribution	Usage Approach	Strength	Weakness
[14]	Proposed an ML-based binary classification model for identifying AI-suitable user stories.	Used the Mendeley dataset. Applied LR, KNN, and SVM with TF-IDF, and word vectors for feature extraction.	Introduced novel use of data for AI feasibility. High accuracy (74–75%) with LR and SVM.	Rely on expert labeling. KNN showed poor accuracy. Dataset is imbalanced.
[6]	Proposed hybrid based approach of CNN + XGBoost model.	Labeled PROMISE_exp and DOSSPRE datasets into SR and NSR.	Achieved highest accuracy of 87.3% on PROMISE_exp.	Focused mainly on binary classification.
[27]	Proposed a hybrid model using ensemble, transfer learning, and feature selection.	Used PROMISE and NASA datasets to evaluate performance.	Demonstrated better AUC-ROC with combined classifiers.	Focused on bug prediction, not requirements classification.
[28]	Proposed a method to recognize and specify NFRs using extended UML use-case diagrams.	Used the PROMISE dataset for empirical evaluation of the NFR identification approach.	Achieved high precision (90.05%) and provided early-stage NFR visualization.	The semi-automatic method still requires manual effort, limited to UML-based environments.
[29]	Proposed NLPReqClassifier model combining TF-IDF and Cosine Similarity.	Used the PROMISE dataset and real-world ERP data for dual evaluation of FR/NFR.	Demonstrated high adaptability in academic and industrial contexts.	Limited to classical NLP techniques; lacks comparison with modern DL models.
[16]	Evaluated 15 ML algorithms with various vectorization techniques.	Used the PROMISE_exp dataset with BoW, TF-IDF, Hashing, and Chi-Square.	Identified Linear SVC + TF-IDF as the best combination with 81.5% F1-score.	Focused only on traditional ML; DL methods not explored.
[19]	Proposed MLRPredictor using OkapiBM and logistic regression for multi-label classification.	Evaluated on the PROMISE dataset with comparative analysis.	Outperformed state-of-the-art models in macro F-1 score and demonstrated generalizability.	Complex transformation of data; lacks deep model integration like Transformers.

Reference	Contribution	Usage Approach	Strength	Weakness
[30]	Proposed a BERT-based model to link NFRs with sustainable environmental factors.	Used the extended PROMISE_exp dataset with a binary classification.	Achieved 90% accuracy, improved all key performance metrics over previous models.	Focused only on sustainability aspects, limiting the general NFR classification utility.
[31]	Addressed class imbalance in requirement classification using SMOTE-Tomek and logistic regression	Used the PROMISE dataset, applied SMOTE-Tomek to handle imbalance and stratified K-fold for evaluation.	Significant improvement in classification accuracy 76.16% compared to baseline 58.31%. The model is interpretable.	Focused only on binary classification (FR/NFR). Limited to LR without comparison to advanced DL models
[32]	First comprehensive experimental study evaluating generative LLMs (Bloom, Gemma, Llama) for requirements classification	Used PROMISE NFR, Functional-Quality, and SecReq datasets to compare generative vs. non-generative models across 400 experiments	Showed Bloom excelled in precision, Gemma in recall, and Llama gave balanced results; asserted that prompt design strongly affects LLM.	Generative LLMs generally underperformed non-generative models (e.g., All-Mini, SBERT) in multi-class classification.
[33]	Proposed a multi-label classification method to categorize NFRs into specific quality attributes.	Used the PROMISE dataset (326 NFRs, 358 FRs) from 15 student projects for training and testing.	Enables fine-grained classification of NFR sub-types, enhancing requirement clarity.	Dataset size affects scalability and model generalization.
[34]	Applied BERT to classify software requirements as FR/NFR.	Used the tera-PROMISE dataset with BERT and its variants.	Achieved 88.6% accuracy with BERT.	Larger models didn't outperform base BERT.
[35]	Demonstrated superior performance of LLMs over traditional ML models.	Used PROMISE_exp and SecReq datasets; applied various strategies with 9 LLMs.	High accuracy and generalization without feature engineering.	Performance depends heavily on the effective-prompt design.
[36]	Evaluated ChatGPT and Gemini on diverse RE tasks and datasets.	Used 4 datasets, including PROMISE, REQuestA, Pure, and Aerospace (NER).	Demonstrated LLM potential in RE tasks, especially in QA.	Underperformed compared to state-of-the-art in classification.
[38]	Highlights NLP and GenAI challenges and solutions in enterprise-scale RE tasks.	Used PROMISE_exp dataset with 969 NFRs labeled into 12 classes to demonstrate RE tasks using NLP techniques.	Offered practical insights, emphasizes real-world applicability with NFR classification.	The dataset remains imbalanced; focus is broader, not limited to experimental evaluation.
[39]	Developed NLP4ReF, a tool using NLTK and GPT-3.5 for classifying and generating	Tested on real-world IoT projects and PROMISE datasets. Used TF-IDF + SVM	GPT achieved 82.5% relevance in generating new requirements. NLTK	GPT use raises privacy and cost concerns, limited by token size and LLM

Reference	Contribution	Usage Approach	Strength	Weakness
	software requirements.	and Word2Vec in the NLTK version; GPT version used semantic prompts via API for generation and classification.	provided solid classification. Output generation is fast and reusable across domains.	noise. NLTK-generated language is less coherent. Evaluation limited to one case.
[23]	Proposed FNRC, an improved RF with Accuracy Sliding Window (ASW) for classifying FR and NFRs.	Applied to the PROMISE and CCHIT datasets. Used standard preprocessing (TF-IDF, stemming), and pruned RF trees using ASW to boost accuracy.	Achieved up to 85% recall and good performance across domains. Reduced model complexity and improved interpretability.	Struggles with multi-NFR statements. Some NFR types (e.g., Availability) had low precision. Manual tuning is needed for thresholds.
[24]	Evaluated multiple ML models (MNB, LR, kNN, SVM) using different feature extraction and selection techniques.	Used PROMISE_exp dataset. Tested BoW and GloVe embeddings with CHI ² and ANOVA.	High binary precision (92.9%) using MNB + BoW. ANOVA improved results in multi-class scenarios.	Limited to PROMISE_exp. Low performance (73.1% accuracy). Did not use DL models.

3. Proposed Methodology

We proposed a hybrid-based ML and DL framework based on CNN-XGBoost, in which SBERT and TF-IDF features are refined through a neural network before final classification using XGBoost. This approach was applied to the PROMISE dataset and achieved promising results. It covers all about the proposed approach and the PROMISE dataset, such as dataset collection, preprocessing, feature extraction, feature refinement, splitting strategy, model training, and evaluation. The detailed methodology is presented in the following Figure.

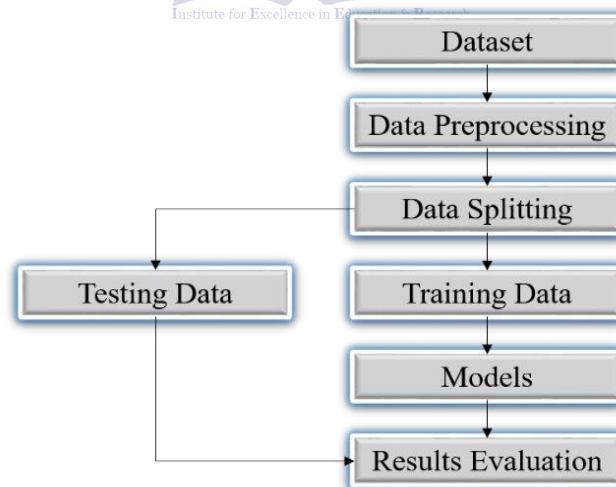


Figure 1 Proposed Methodology

Dataset Description

For this research task, we have used the publically available dataset known as PROMISE. These requirements, which were collected from actual development settings, are both functional and non-functional software specifications. The textual nature of the dataset makes it a very good fit for NLP, ML and DL research applications. One of the biggest advantages of the PROMISE dataset is that it is labeled already, which makes the supervised learning process a lot easier. The original PROMISE dataset contains 969 requirement instances with several attributes, including serial number (S.No), file number, requirement text, and requirement type (class label). For this study, records belonging to class F were removed, resulting in a final dataset of 525 instances used for binary classification.

Data Preprocessing

The PROMISE dataset was preprocessed to ensure the efficient training, assessment, and prediction in the proposed security requirement classification framework, while ensuring it remains clean, reliable, and of high quality. These pre-processing methods not only enhanced the quality of features, but also reduced computation costs and enhanced the learning ability. All things considered, preprocessing made sure that the dataset was transformed into a transparent, dependable, resilient, and high-quality that would be appropriate for effective training, evaluation, and prediction throughout security requirement classification framework.

Dataset Splitting Strategy

The dataset was divided into 80% training and 20% testing using the `train_test_split()` function with stratified sampling (`stratify=labels`) and `random_state=42` to ensure reproducibility. To perform additional evaluation improvement, a three-fold test split cross-validation is performed. Finally, the model performance is evaluated on unseen data through the use of a test set. This stratified multi-stage splitting strategy allows the study to provide stability of the evaluation, fairness, and realistic deployment conditions.

Proposed Model

The framework combines deep feature refinement with the XGBoost to develop a robust hybrid model. By combining these methods, the model is better able to extract statistical textual patterns and contextual semantic information from requirement statements. Figure 2 shows the general workflow and architectural layout of the suggested hybrid framework. The suggested methodology is made up of several interrelated steps that work together to efficiently categorize software needs into security-related and non-security-related groups.

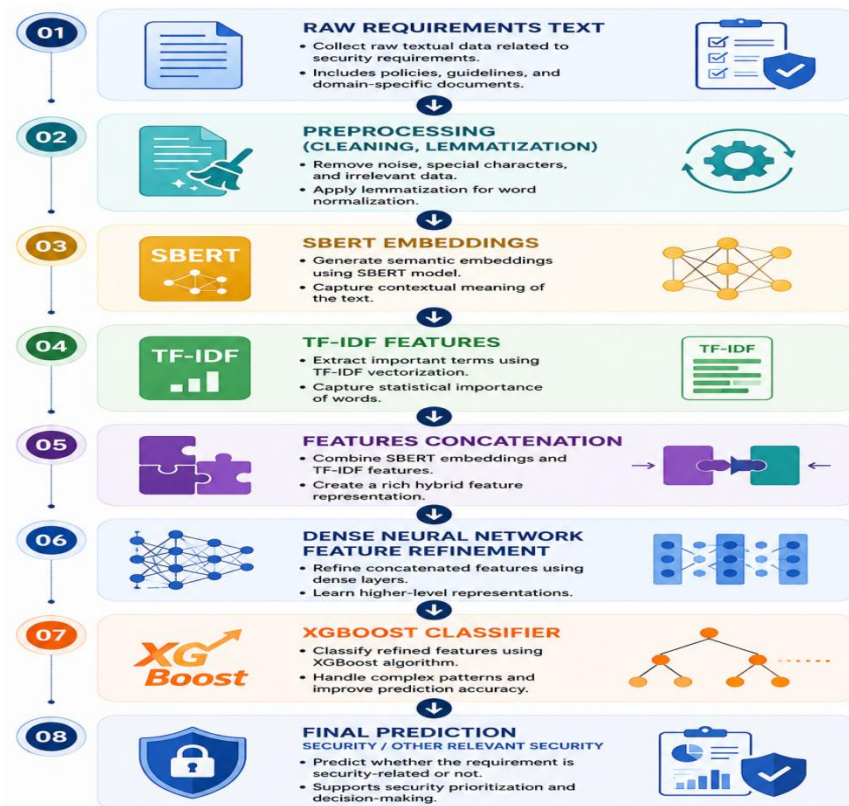


Figure 2 Proposed Solution Architecture

SBERT Embeddings

Dense semantic embeddings are produced from the textual requirements using SBERT. Unlike traditional embedding techniques, SBERT produces meaningful sentence-level vector representations. These embeddings allow the model to understand the semantic meaning and contextual dependencies within the requirement text. As a result, the proposed approach becomes more capable of distinguishing between SR and NSR based on contextual information.

TF-IDF Feature Extraction

In addition to semantic embeddings, the proposed model utilized TF-IDF for statistical feature extraction. TF-IDF helps identify domain-specific keywords and statistically important terms associated with security requirements. While SBERT captures semantic meaning, TF-IDF captures frequency-based textual importance. Therefore, combining these two representations enables the model to learn richer and more discriminative textual patterns.

Fully Connected Neural Network-Based Feature Refinement

The proposed framework follows the approved CNN-XGBoost architecture. The neural network learned refined feature representations from the concatenated SBERT and TF-IDF feature vectors. The objective of the neural network was to learn compact latent representations from the combined SBERT and TF-IDF features. Those vectors fed to the fully connected neural network for feature refinement.

XGBoost-Based Classification

The XGBoost performed well and generated outstanding results in terms accuracy, f1 score, precision, recall and AUC-ROC. For this task this model was chosen due to the outstanding performance in handling the imbalanced datasets, its robustness against noisy features, and its ability to model non-linear relationships within the extracted refined feature vectors. When the XGBoost model trained and tested on the extracted refined features, they gained an outstanding performance and hence they outperformed other models.

Evaluation Metrics

In this research work, several widely used and well-known performance evaluation measures were used to evaluate the effectiveness of the proposed hybrid CNN-XGBoost framework. These measures are accuracy, precision, recall and F1-score. These evaluation metrics were used in the proposed security requirement classification framework to assess the model's ability to extract security-related and non-security-related requirements.

Accuracy

Classification accuracy is the ratio of occurrences that are appropriately categorized by the classification learner. Means the ratio of suitably predicted samples to the total number of examples.

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

F-Score

F-measure is the harmonic mean of precision and recall. For the best performance, it is one, and for the foulest performance, it is zero.

$$F1 = \frac{\text{Precision} * \text{recall}}{\text{precision} + \text{recall}} \quad (2)$$

Precision

Precision or positive predictive value. Precision is the ratio of true positives (absence classified as absence) to all instances classified as positive (total samples classified as absence).

$$\text{Precision} = \frac{TP}{TP+FP} \quad (3)$$

Recall

Recall in ML measures the ability of a model to correctly identify all actual positive cases. It tells us how many real positives were captured by the model [40].

$$\text{Recall} = \frac{TP}{TP+FN} \quad (4)$$

4. Results and Discussion

The proposed model was implemented in this platform using the popular programming language Python. For this task, a popular dataset PROMISE was used, and for embeddings, TF-IDF and SBERT were used. The dataset was split in a ratio of 80:20, where 80% of the data was used for model training and 20% for model testing. To understand the above, the class specifications of the PROMISE data set have been graphically represented in Figure 3 below.

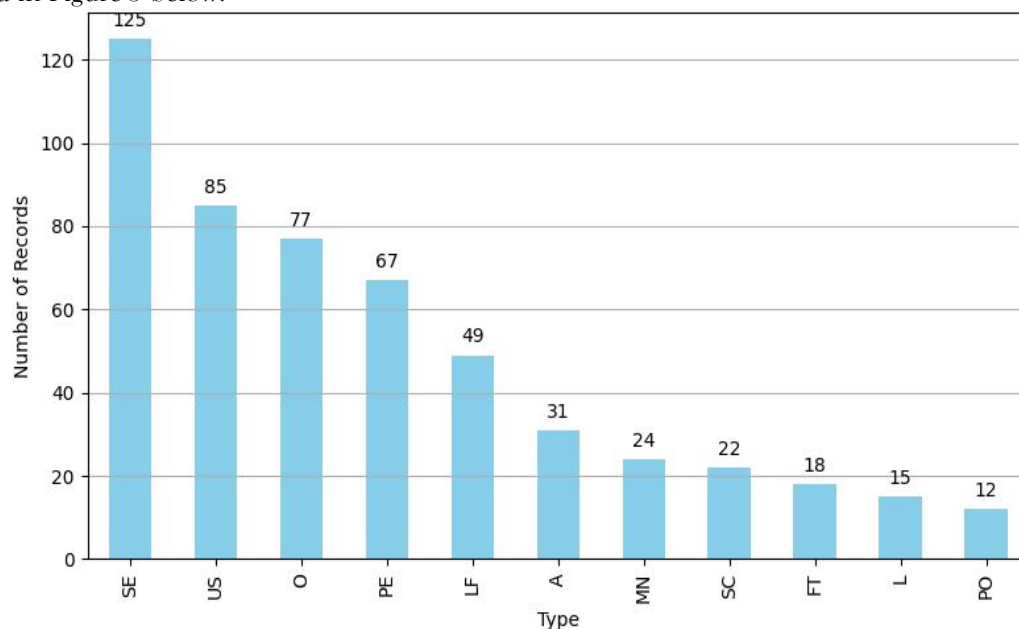


Figure 3 Class Distribution in Type

The data set as presented in Figure 3 has an imbalanced distribution of the various types of requirements. The high records in the SE class dominate the highest followed by US, O and PE, implying dominance of these classes in the data set. On the other hand, other classes, like PO, L, and FT, have lower samples, which is a large difference between the classes.

This disproportion indicates that the model can be biased with majority classes in case it is not addressed. In Figure 4, the normalized confusion matrix is given, which shows the correlation between the actual class and the predicted class of Security and Non-Security requirements. Performance in classification across the classes is better understood by use of a normalized matrix in that correct predictions are indicated and patterns of misclassification are noted.

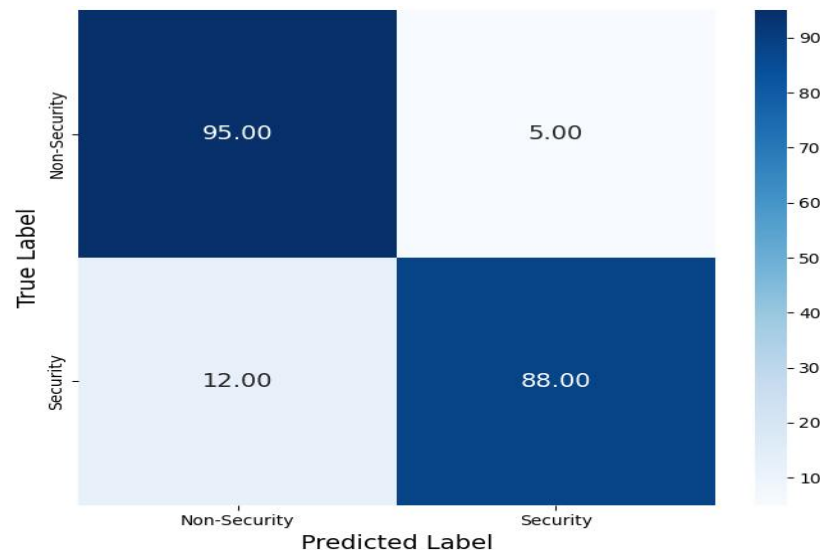


Figure 4 Normalized Confusion Matrix for Security Requirements Classification

The proposed model has a good performance in classification in both classes as depicted by Figure 4. The true negative rate of using the model is high because 95% of the Non-Security requirements are truly classified and only 5% are falsely classified as being Security. Similarly 88% Security requirements are appropriately determined and 12% are Non-Security. The existing CNN+XGBoost model proposed in the original study [6] is a hybrid learning framework developed for the classification of software requirement statements into SR and NSR. The CNN and XGBoost together could take advantage of the deep feature learning, ensemble classification benefits, and more. Based on the reported results, 10s epochs and batch size of 32 were considered as an optimum set. In this setup, the suggested hybrid system attained a accuracy of $\sim 87.3\%$, with a precision and recall of 85.2% and 89.9%, respectively, and F1-score of 87.3%.

In this study, the new CNN+XGBoost model has a number of critical improvements relative to the current hybrid architecture. The initial significant enhancement is in the preprocessing pipeline wherein the text cleaning process was improved to eliminate noise including the punctuations and irrelevant symbols. The procedure of lemmatization was used to simplify words to their root words so that that a single term could be represented in different variations. Also, the standardization of the requirement statements was done to get a standard structure before tokenization occurred. These changes enhanced the capacity of CNN to identify long-range dependencies and subtle semantics of the textual information. However, this was improved by employing a fully connected neural network with dense layers, batch normalization, and dropout to learn more discriminative latent feature representations from the concatenated SBERT and TF-IDF features. Additionally, a second dense layer with 128 neurons was employed to learn more compact and discriminative feature representations. Also, dropout was added in order to reduce overfitting in the model. In order to obtain a clear picture of the inner architecture of the proposed neural network that is employed to refine the features, the elaborated layer-wise architecture is summarized in Table 2.

Table 2: *CNN Model Summary*

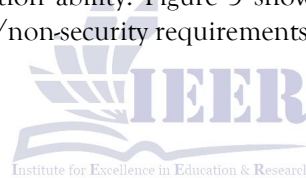
Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 1711)	0
Dense (256, ReLU)	(None, 256)	438,272
batch_normalization (BatchNormalization)	(None, 256)	1,024
dropout (Dropout)	(None, 256)	0
Dense (128, ReLU)	(None, 128)	32,896
dropout_1 (Dropout)	(None, 128)	0
Dense (1, Sigmoid)	(None, 1)	129

As it could be observed, the XGBoost classifier utilized in the proposed hybrid model uses a specific set of hyperparameters, which are provided in Table 3.

Table 3: *XGBoost Model Summary*

Hyperparameter	Value
Objective	binary:logistic
eval_metric	logloss
n_estimators	200
max_depth	10
learning_rate	0.01
subsample	0.9
random_state	42

The XGBoost classifier was configured to solve a binary classification problem using the binary:logistic objective function, as shown in Table 3. Thus, the hybrid CNN-XGBoost framework was able to improve the classification performance and generalization ability. Figure 5 shows the overall architecture of the proposed hybrid classification approach for security/non-security requirements of a software.



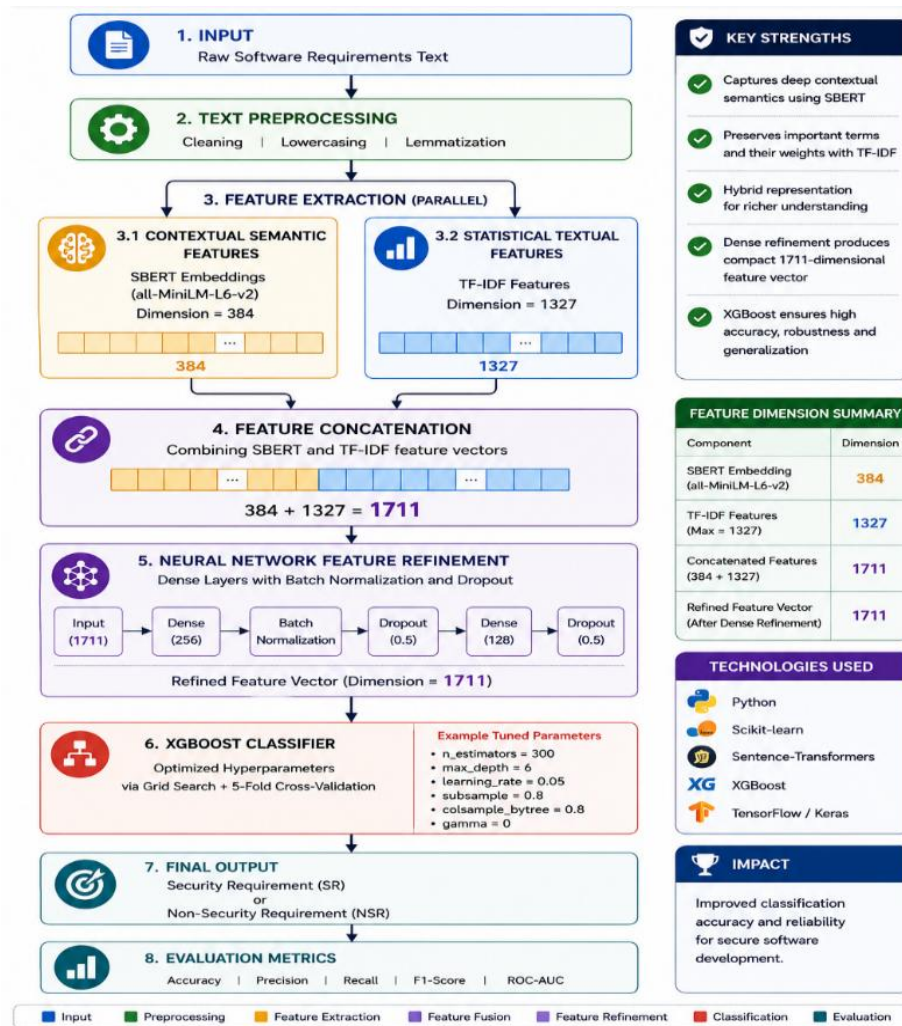


Figure 5 Proposed Hybrid Model Architecture for Security Requirement Classification

The proposed model starts with raw software requirement statements as depicted in Figure 5, which is processed to text preprocessing which involves cleaning, lowercasing and lemmatizing as a measure to minimize noise and normalize the input. The hybrid CNN-XGBoost framework achieved with an accuracy of 93.33%, precision of 84.62%, recall of 88.00%, F1-score of 86.27%, and AUC-ROC of 99.05%, which is a better model than the baseline model mentioned in the original paper. In order to present the improvement of the hybrid model with the help of the enhanced hybrid model, Figure 6 shows the performance of the updated model.

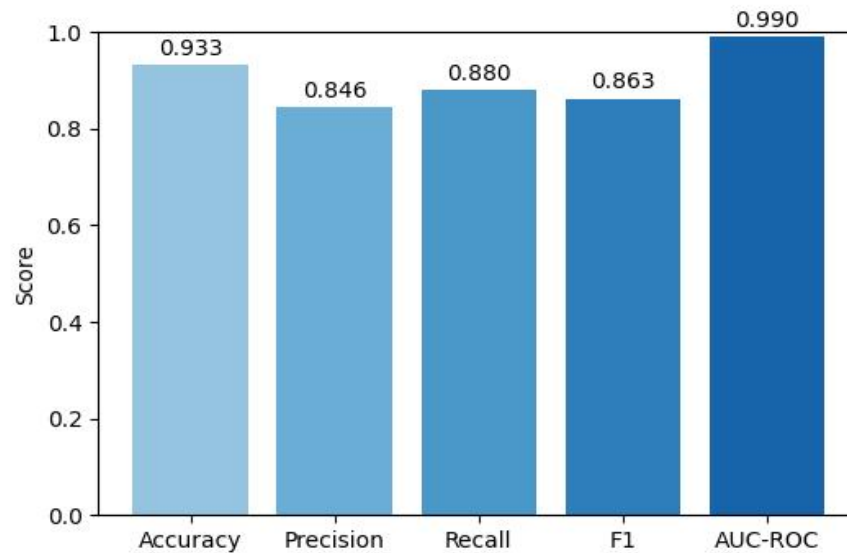


Figure 6 Updated Model's Performance

To evaluate the extent to which the updated CNN component of the hybrid model is effectively learning, the training and validation loss curves can be plotted, as shown in Figure 6. These loss curves demonstrate the quality of reducing the error during the training and the quality of generalizing the learned information to previously unseen data.

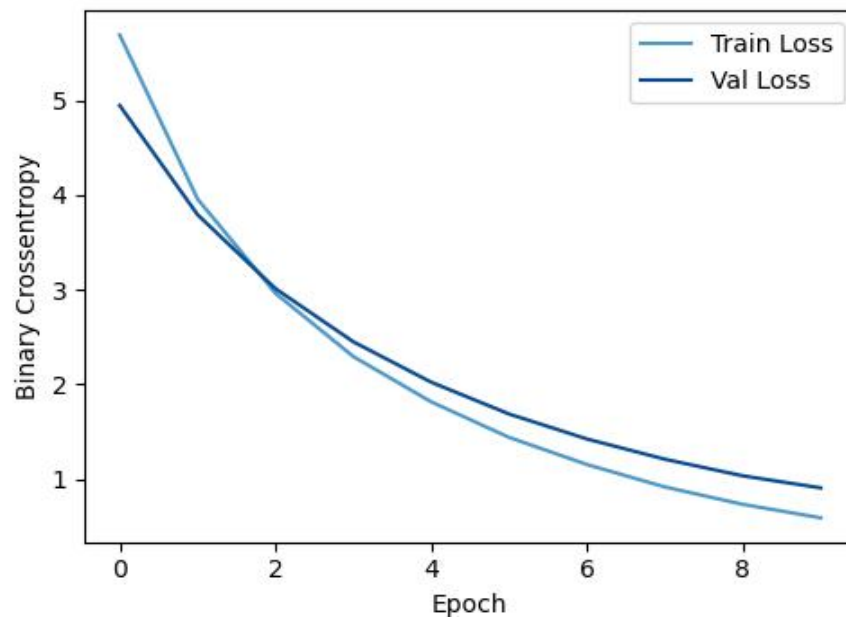


Figure 7 Training and Test Loss of the Proposed Hybrid CNN-XGBoost Framework

As a continuation of the loss analysis and further evaluation of the model's learning behavior, Figure 7 illustrates the training and validation accuracy curves of the proposed CNN + XGBoost model. These curves show that the model is effective in the proper classification of data in consecutive training epochs. Through the trends of analysis of the accuracy, one can conclude whether the model is improving steadily, being stable, and whether it is capable of generalizing to the unseen data throughout the training process.

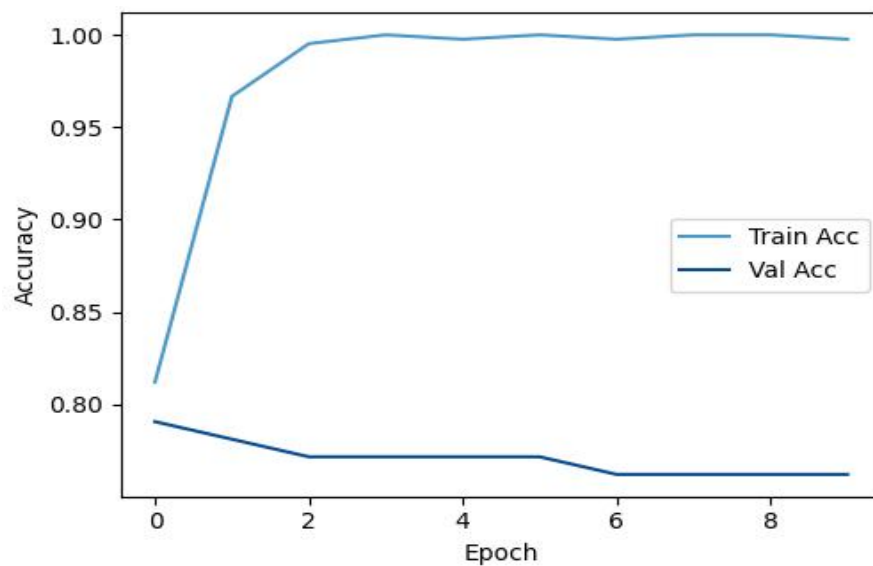


Figure 8. Training and Test Accuracy of the Proposed Hybrid CNN-XGBoost Framework

The trends of the accuracy curves in Figure 8 demonstrate that training accuracy is highly upwards with an initial steep growth which levels off almost immediately to near-perfect accuracy indicating that the model has acquired the underlying data patterns.

Comparison

The comparison includes more than just evaluation metrics; it also covers architectural, pre-processing, and optimization changes made to the new framework. The detailed comparison between the existing and updated Hybrid CNN-XGBoost models based on multiple evaluation metrics and architectural aspects is given in Table 4.

Table 4: Comparison of Existing and Updated CNN+XGBoost Models

Aspect	Existing Model [6]	Updated Model
Text Preprocessing	Basic cleaning	Advanced cleaning
Feature Representation	CNN feature extraction	SBERT + TF-IDF with neural feature refinement
Neural Network Architecture	Shallow CNN architecture	Fully connected neural network
Activation Function	ReLU	ReLU
Regularization	Minimal or no dropout	Dropout applied to reduce overfitting
Epochs Used	20 epochs	10 epochs
Batch Size	32	32
XGBoost Parameters	Default/limited tuning	Tuned <code>n_estimators</code> , <code>max_depth</code> , and <code>learning_rate</code>
Data Splitting	Random train-test split	Stratified 80:20 train-test split
Accuracy	87.30%	93.33%
Precision	84.00%	84.62%
Recall	85.00%	88.00%
F1-Score	77.00%	86.27%
AUC-ROC	Not reported	99.05%

The updated Hybrid CNN-XGBoost model outperformed the existing architecture in all major evaluation metrics as can be seen in Table 4, demonstrating the effectiveness of the proposed enhancements. One of the key reasons for this improvement is the inclusion of advanced preprocessing techniques and improved feature representation methods. The experimental results overall demonstrate the effectiveness of the developed improvements and prove that the enhanced hybrid CNN-XGBoost model is a more accurate, robust and efficient solution for automated security requirement classification.

5. Conclusion

The aim of this study was to help in making the automated security requirement classification to be effective by introducing the hybrid framework which is the combination of a neural feature refinement network and XGBoost, which is used to classify the extracted features. The findings of the study confirm the given methodology can be used to address various limitations of the current methods. The use of both methods of hybrid feature representation simultaneously is another significant contribution. TF-IDF captured the frequency and importance of terms in the requirement statements. SBERT embeddings captured the semantic meaning and contextual information of the requirement statements. The combination of these two representations produced a rich and useful feature space which assisted the model to tell the difference between the security requirements and non-security requirements. The model was able to have a solid convergence and good performance by thoroughly parameter tuning both the neural feature refinement network and XGBoost sub parts and by also ensuring stratified data splitting. The model achieved an accuracy of 93.33% and the values of its precision, recall, and F1-score were all very good. The score of the model in terms of AUC-ROC of 99.05% also indicates that the model can easily distinguish the difference between classes. These findings indicate that this model is far superior to the baseline model and other approaches which have been outlined in the literature. The study is significant not only due to the technical contribution of the study, but also because of the practical contribution of the study.

6. Future Work

Despite the promising performance of the proposed framework, several opportunities remain for future research. Future studies should evaluate the model on larger and more diverse industrial datasets to further validate its generalizability. In addition, incorporating transformer-based language models and XAI techniques may further improve semantic understanding, model interpretability, and classification performance. Finally, optimizing the hybrid architecture for computational efficiency and integrating it into intelligent requirements engineering tools would enhance its practical applicability in real-world software development environments.

References

- [1] B. A. Suris, A. Thobirin, S. Surono, and M. N. A. Abdunazar, "Enhancing Multi-Class Classification of Non-Functional Requirements Using a BERT-DBN Hybrid Model," *INTENSIF: Jurnal Ilmiah Penelitian dan Penerapan Teknologi Sistem Informasi*, vol. 9, pp. 195-210, 2025.
- [2] Q. Zhang, C. Fang, Y. Xie, Y. Zhang, Y. Yang, W. Sun, et al., "A survey on large language models for software engineering," *arXiv preprint arXiv:2312.15223*, 2023.
- [3] K. Ronanki, C. Berger, and J. Horkoff, "Investigating chatgpt's potential to assist in requirements elicitation processes," in *2023 49th Euromicro conference on software engineering and advanced applications (SEAA)*, 2023, pp. 354-361.
- [4] A. Hemmat, M. Sharbaf, S. Kolahdouz-Rahimi, K. Lano, and S. Y. Tehrani, "Research directions for using LLM in software requirement engineering: A systematic review," *Frontiers in Computer Science*, vol. 7, p. 1519437, 2025.
- [5] C. Gao, X. Hu, S. Gao, X. Xia, and Z. Jin, "The current challenges of software engineering in the era of large language models," *ACM Transactions on Software Engineering and Methodology*, vol. 34, pp. 1-30, 2025.
- [6] S. Hassan, Q. Li, M. Zubair, R. A. Alsowail, and M. Umair, "An Improved Hybrid Deep Learning Approach for Security Requirements Classification," *CMCCOMPUTERS MATERIALS & CONTINUA*, vol. 82, pp. 4041-4067, 2025.
- [7] H. Mokayed, T. Z. Quan, L. Alkhaled, and V. Sivakumar, "Real-time human detection and counting system using deep learning computer vision techniques," in *Artificial Intelligence and Applications*, 2023, pp. 205-213.
- [8] N. Tasneem, H. B. Zulzalil, and S. a. Hassan, "Enhancing agile software development: A systematic literature review of requirement prioritization and reprioritization techniques," *IEEE Access*, 2025.
- [9] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, et al., "Large language models for software engineering: A systematic literature review," *ACM Transactions on Software Engineering and Methodology*, vol. 33, pp. 1-79, 2024.

- [10] D. Luitel, S. Hassani, and M. Sabetzadeh, "Improving requirements completeness: Automated assistance through large language models," *Requirements Engineering*, vol. 29, pp. 73-95, 2024.
- [11] F. A. Boamah, X. Jin, S. Senaratne, and S. Perera, "Transition from traditional knowledge retrieval into AI-powered knowledge retrieval in infrastructure projects: A literature review," *Infrastructures*, vol. 10, p. 35, 2025.
- [12] V. Shankar, "Machine Learning for Linux Kernel Optimization: Current Trends and Future Directions," *International Journal of Computer Sciences and Engineering*, vol. 13, pp. 56-64, 2025.
- [13] W. Alhoshan, A. Ferrari, and L. Zhao, "Zero-shot learning for requirements classification: An exploratory study," *Information and Software Technology*, vol. 159, p. 107202, 2023.
- [14] M. Gaona-Cuevas, V. B. Guerrero, and F. H. Vera-Rivera, "The smart product backlog: A classification model of user stories," *IEEE Access*, 2024.
- [15] F. P. B. Muhamad, E. Mulyani, M. S. Bunga, and A. F. Mushafa, "A Class Balancing Methods Comparison in Software Requirement Classification Using a Support Vector Machine," 2023.
- [16] A. Rahman, A. Nayem, and S. Siddik, "Non-Functional Requirements Classification Using Machine Learning Algorithms," *International Journal of Intelligent Systems and Applications*, vol. 15, pp. 56-69, 2023.
- [17] H. Ali, "Cloud-Based Machine Learning for Scalable Classification of Software Requirements: Insights from the PROMISE Dataset," *Available at SSRN 5370198*.
- [18] K. Kaur and P. Kaur, "BERT-CNN: improving BERT for requirements classification using CNN," *Procedia Computer Science*, vol. 218, pp. 2604-2611, 2023.
- [19] S. Saleem, M. N. Asim, L. Van Elst, M. Junker, and A. Dengel, "MLR-predictor: a versatile and efficient computational framework for multi-label requirements classification," *Frontiers in Artificial Intelligence*, vol. 7, p. 1481581, 2024.
- [20] A. El-Hajjani, N. Fafin, and C. Salinesi, "Which ai technique is better to classify requirements? an experiment with svm, lstm, and chatgpt," *arXiv preprint arXiv:2311.11547*, 2023.
- [21] B. Jamasb, R. Akbari, and S. R. Khayami, "On the applicability of explainable artificial intelligence for software requirement analysis," *arXiv preprint arXiv:2302.05266*, 2023.
- [22] D. A. C. de Souza, D. Albuquerque, E. Dantas Filho, M. Perkusich, and A. Perkusich, "Using Machine Learning for Non-Functional Requirements Classification: A Practical Study," in *Workshop Brasileiro de Engenharia de Software Inteligente (ISE)*, 2023, pp. 35-38.
- [23] J. Abbas, A. Ahmad, S. M. Shaheed, R. Fatima, S. Shah, M. Elaffendi, et al., "Classification and Comprehension of Software Requirements Using Ensemble Learning," *Computers, Materials & Continua*, vol. 80, 2024.
- [24] E. Dias Canedo and B. Cordeiro Mendes, "Software requirements classification using machine learning algorithms," *Entropy*, vol. 22, p. 1057, 2020.
- [25] M. I. Limaylla-Lunarejo, N. Condori-Fernandez, and M. R. Luaces, "Evaluating Non-Functional Requirements Classification for Spanish Text: Traditional vs. Deep Learning Approaches."
- [26] P. Minina, A. Sadovykh, and V. Ivanov, "Detecting Security Requirements in GitHub Issues-Novel Dataset and SmallBERT-based model," in *2023 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2023, pp. 315-318.
- [27] Q. M. ul Haq, F. Arif, K. Aurangzeb, N. ul Ain, J. A. Khan, S. Rubab, et al., "Identification of software bugs by analyzing natural language-based requirements using optimized deep learning features," *Computers, Materials & Continua*, vol. 78, pp. 4379-4397, 2024.
- [28] K. Patel, T. Trivedi, and U. Shah, "User Centered Non-Functional Requirements Specification-An Extended Use-Case Diagram," *Procedia Computer Science*, vol. 235, pp. 240-249, 2024.
- [29] R. Izhar, K. Cosh, L. Ramingwong, S. Ramingwong, and S. N. Bhatti, "An Efficient Methodology for the Categorization of Software Requirements Using Natural Language Processing and Similarity Analysis," *IEEE Access*, 2025.

- [30] S. Hassan, Q. Li, M. Zubair, R. A. Alsowail, and M. A. Qureshi, "Unveiling the correlation between nonfunctional requirements and sustainable environmental factors using a machine learning model," *Sustainability*, vol. 16, p. 5901, 2024.
- [31] B. Or, "Improving Requirements Classification with SMOTE-Tomek Preprocessing," *arXiv preprint arXiv:2501.06491*, 2025.
- [32] W. Alhoshan, A. Ferrari, and L. Zhao, "How Effective are Generative Large Language Models in Performing Requirements Classification?," *arXiv preprint arXiv:2504.16768*, 2025.
- [33] S. M. García, C. A. Fernández-y-Fernández, and E. R. Pérez, "Classification of non-functional requirements using convolutional neural networks," *Programming and Computer Software*, vol. 49, pp. 705-711, 2023.
- [34] H. Xu and G. Xu, "Applying the Bert Algorithm for Software Requirements Classification."
- [35] M. de Sá Martin, "Evaluating the Potential of Large Language Models in Security-Related Software Requirements Classification," PUC-Rio, 2025.
- [36] S. Saleem, M. N. Asim, L. V. Elst, and A. Dengel, "Generative language models potential for requirement engineering applications: insights into current strengths and limitations," *Complex & Intelligent Systems*, vol. 11, p. 278, 2025.
- [37] B. C. d. SILVA, "ReqCluster4IoT: um método de agrupamento de requisitos para aplicações de IoT," 2025.
- [38] S. Ghaisas and A. Singhal, "Dealing with Data for RE: Mitigating Challenges while using NLP and Generative AI," in *Handbook on Natural Language Processing for Requirements Engineering*, ed Springer, 2025, pp. 457-486.
- [39] J. Peer, Y. Mordecai, and Y. Reich, "NLP4ReF: Requirements Classification and Forecasting: From Model-Based Design to Large Language Models," in *2024 IEEE Aerospace Conference*, 2024, pp. 1-16.
- [40] A. K. Dwivedi, "Performance evaluation of different machine learning techniques for prediction of heart disease," *Neural Computing and Applications*, vol. 29, pp. 685-693, 2018.

