iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VII **Month of publication:** July 2026

DOI: https://doi.org/10.22214/ijraset.2026.84185

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VII July 2026- Available at www.ijraset.com

Requirement Classification and Prioritization Using Artificial Intelligence and Natural Language Processing

Sai Sindhuja Bhukya[1], Naveen Kumar Nuthanapati[2]
[1]*Department of Computer Science and Engineering, Jawaharlal Nehru Technological University, Hyderabad, India*
[2]*Department of Computer Science and Engineering, Jawaharlal Nehru Technological University, Hyderabad, India*

Abstract: ***Requirement engineering is a foundational stage of the software development life cycle, and the accuracy with which requirements are classified directly influences downstream design, testing, and cost estimation. Software requirements are commonly separated into Functional Requirements (FR), which describe what a system must do, and Non-Functional Requirements (NFR), which describe how well the system must do it, covering attributes such as performance, security, and usability. When this separation is carried out manually, the process is slow, subjective, and prone to disagreement between analysts, particularly as project size grows. This paper presents a Requirement Classification and Prioritization Tool that combines Natural Language Processing (NLP) with machine learning to automate the FR/NFR decision. Requirement statements are cleaned and normalised, then represented numerically through two complementary techniques: Term Frequency-Inverse Document Frequency (TF-IDF) and contextual BERT embeddings. Four classifiers-Logistic Regression, Support Vector Machine, Random Forest, and a***
BERT-based model-are trained and benchmarked on a dataset of 6,086 labelled requirement statements using accuracy, precision, recall, and
F1-score. A Gradio-based interface allows a requirement to be submitted and its predicted category, confidence score, and a short explanation to be viewed immediately. The results indicate that transformer-based representations offer a modest but consistent improvement in contextual understanding over TF-IDF, while classical classifiers remain competitive and considerably cheaper to train.

Keywords-Software Requirements Engineering; Functional Requirements; Non-Functional Requirements; Natural Language Processing; Machine Learning; BERT; TF-IDF; Text Classification; Gradio.

I. INTRODUCTION

Software requirements describe the functions a system must perform and the constraints under which it must perform them, and they form the basis on which a system is designed, built, and tested [7]. Requirements are conventionally split into two broad classes: Functional Requirements (FR), which enumerate the services and operations a system offers, and Non-Functional Requirements (NFR), which capture quality attributes such as reliability, performance, security, and usability [6]. Correctly separating the two matters because FR and NFR are usually verified through different means, are prioritised differently by project managers, and feed into different parts of the design.

In practice, this separation is still carried out largely by hand. An analyst reads each statement in a specification document and assigns it to a category based on judgement and experience. As the number of requirements in a project grows into the hundreds or thousands, the manual process becomes slow, inconsistent across reviewers, and a measurable drain on project schedules. Ambiguity in natural-language requirement statements compounds the problem, since functional and nonfunctional requirements frequently share similar sentence structures and vocabulary, which makes simple rule-based separation unreliable [8].

Advances in Natural Language Processing (NLP) and Machine Learning (ML) offer a practical route around this bottleneck. NLP techniques can tokenise, normalise, and vectorise free-text requirement statements, while supervised ML models can learn to associate linguistic patterns with FR or NFR labels from historical labelled requirement sets. Several studies have investigated this direction using classical classifiers such as Support Vector Machines and Logistic Regression [1], and, more recently, transformer-based language models such as BERT that capture contextual and semantic relationships within a sentence [2, 9, 10].

This paper reports on the design, implementation, and evaluation of a Requirement Classification and Prioritization Tool that automatically labels requirement statements as FR or NFR. The tool combines two feature-extraction strategies, TF-IDF and BERT embeddings, with four classification approaches: Logistic Regression, Support Vector Machine (SVM), Random Forest, and a BERT + SVM hybrid. The trained models are exposed through an interactive Gradio interface that returns a predicted category together with a confidence score. The remainder of the paper is organised as follows. Section II reviews related work on automated requirement classification. Section III describes the proposed methodology and system architecture. Section IV presents the system design through UML diagrams. Section V discusses implementation details. Section VI reports the experimental results. Section VII summarises the testing carried out, and Section VIII concludes the paper and outlines directions for future work.

A. Motivation

Large software projects routinely generate requirement documents that run into hundreds of statements, and analysing each one manually for its FR/NFR type is repetitive and time-intensive. Because the assignment ultimately depends on an individual analyst's interpretation, the same requirement can be labelled differently by two reviewers, reducing the reliability of any downstream planning that depends on the classification. Automating this step with AI and NLP promises to reduce that inconsistency while freeing analysts to focus on higher-value review tasks.

B. Challenges in Automated Requirement Classification

Several factors make automated FR/NFR classification harder than a typical text-classification task. Functional and non-functional requirements are frequently expressed with similar linguistic constructs, so surface-level keyword matching is unreliable. Real specifications can run into thousands of statements, which rules out manual verification of every prediction. The relative proportion of FR to NFR statements in a typical dataset is also uneven, which can bias a classifier toward the majority class. Finally, some NFR statements require semantic understanding, for example distinguishing a performance constraint from a functional description of a monitoring feature. Plain keyword features cannot capture this well, which motivates the use of contextual embeddings such as BERT [9].

II. LITERATURE SURVEY

A number of studies have investigated the use of AI and NLP for classifying software requirements. Table 1 summarises six representative studies alongside the present work.

Canedo and Mendes [1] framed FR/NFR separation as a supervised text-classification task and compared Bag-of-Words, TF-IDF, and Chi-square feature selection combined with SVM,

Logistic Regression, Multinomial Na¨ıve Bayes, and KNN on the

PROMISE requirements dataset, reporting that TF-IDF paired with SVM or Logistic Regression gave the strongest results. The approach nonetheless remains dependent on hand-crafted features and captures limited context.

Airlangga [2] proposed a semi-supervised GAN-BERT model that combines labelled and unlabelled requirement statements to improve classification when annotated data is scarce, obtaining higher accuracy than purely supervised baselines at the cost of substantially higher training time and computational demand.

Khayashi et al. [3] benchmarked CNN, LSTM, and BiLSTM architectures on the PURE requirements dataset and found that such models learn semantic features automatically, reducing the need for manual feature engineering, though they still require large labelled corpora and considerable compute.

Binkhonain and Zhao [4] introduced a hierarchical classification scheme built on BERT and RoBERTa that first assigns a broad requirement category before refining it into sub-categories, improving prediction accuracy relative to flat binary classification but requiring larger, more carefully structured datasets.

Alhoshan et al. [5] explored zero-shot classification of requirements using pre-trained transformer models, allowing new requirement categories to be handled without additional training data. The method trails supervised approaches in raw accuracy and is sensitive to prompt design.

Kurtanovic and Maalej [6] used supervised classifiers,´ including SVM, Na¨ıve Bayes, and Decision Tree, over the PROMISE dataset, providing one of the earliest reliable baselines for FR/NFR separation, although reported performance depends heavily on handcrafted linguistic features and degrades on ambiguous requirement text.

Table 1: Summary of Related Work

Ref.	Technique	Dataset	Limitation
[1]	TF-IDF + SVM/LR/NB/KNN	PROMISE	Depends on handcrafted features
[2]	GAN-BERT semi-supervised model	SR Dataset	High compute and needs unlabelled data
[3]	CNN, LSTM, and BiLSTM	PURE	Needs large labelled data
[4]	BERT/RoBERTa hierarchical classification	PURE	Needs larger structured data
[5]	Zero-shot transformers	PURE	Lower accuracy and prompt-sensitive
[6]	SVM, NB, and Decision Tree	PROMISE	Struggles with ambiguous text

Across these studies, two broad conclusions recur. Classical machine-learning pipelines built on TF-IDF remain fast, interpretable, and reasonably accurate, but their performance is bounded by the quality of the hand-engineered features. Transformer-based approaches capture more of the surrounding sentence context and narrow the gap on harder, more ambiguous requirements, at the price of longer training time and a heavier hardware footprint [9, 10]. The present work builds on this observation by evaluating both families side by side: TF-IDF with Logistic Regression, SVM, and Random Forest, and BERT embeddings with an SVM classifier on the same dataset.

III. PROPOSED METHODOLOGY

A. *System Architecture*

The proposed tool is organised as a modular pipeline consisting of a requirement input module, a text pre-processing module, a feature extraction module, a classification module, a result generation module, and a Gradio-based user interface, as shown in Fig. 1. A requirement statement entered through the interface is cleaned and normalised, converted into a numeric feature vector, passed through the selected classification model, and returned to the user together with its predicted label and confidence score.

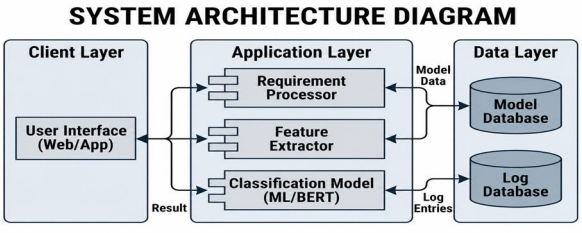


Figure 1: System architecture of the requirement classification tool.

B. *Data Collection and Pre-processing*

The dataset used for training and evaluation contains 6,086 requirement statements drawn from publicly available requirement specification sources, labelled as either FR or NFR. Of these, 3,964 statements (65.1%) are functional and 2,122 (34.9%) are non-functional, giving a mild class imbalance that is accounted for during evaluation through class-wise precision and recall rather than accuracy alone. Before feature extraction, every requirement statement is converted to lower case, stripped of special characters and extra whitespace, and normalised, which improves the quality of the resulting feature vectors and the stability of model training.

C. Feature Extraction

Two complementary representations are used to convert cleaned requirement text into numeric form. TF-IDF weighs each term in a requirement statement by how distinctive it is across the dataset, giving higher weight to terms that are frequent within a statement but rare across the corpus. It is inexpensive to compute and pairs naturally with classical linear and tree-based classifiers. BERT embeddings, in contrast, are produced by a pre-trained bidirectional transformer that encodes each word in the context of the full sentence, allowing the representation to capture semantic relationships. This helps distinguish, for example, a performance constraint from a functional description that happens to use similar words [9].

D. Classification Models

Four classification approaches are trained and compared. Logistic Regression is used as a fast, interpretable linear baseline that estimates the probability of a requirement belonging to the FR or NFR class. Support Vector Machine (SVM) constructs a maximum-margin boundary between the two classes and is well suited to the high-dimensional sparse vectors produced by TFIDF. Random Forest combines the predictions of many decision trees to reduce variance and resist over-fitting on the training data. Finally, a BERT + SVM hybrid uses contextual BERT embeddings as input features to an SVM classifier, combining the semantic strength of BERT with the discriminative boundary of SVM [10].

E. Result Generation and User Interface

Once a classifier produces a prediction, the result generation module computes a confidence score from the model's class probabilities and prepares a short natural-language explanation of the outcome. These are displayed to the user through a Gradio-based web interface, which accepts a free-text requirement statement, lets the user choose which trained model to use, and returns the predicted label, confidence score, and explanation in real time.

IV. SYSTEM DESIGN

The internal behaviour of the tool is documented using standard UML diagrams: a data flow diagram that traces how a requirement statement moves through the pipeline, and use case, class, activity, and sequence diagrams that describe, respectively, the interactions available to the user, the static structure of the code, the flow of execution, and the order in which the internal modules communicate.

A. Data Flow Diagram

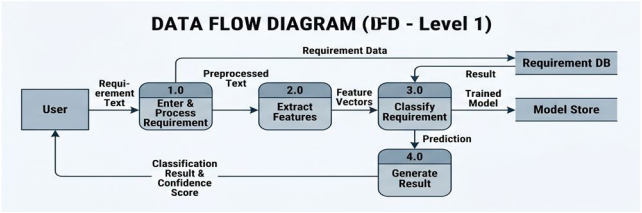


Figure 2: Level-1 data flow diagram of the proposed system.

B. Use Case Diagram

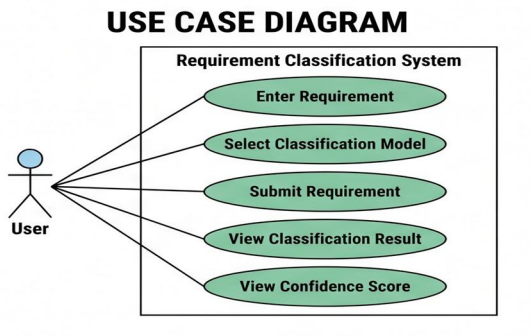


Figure 3: Use case diagram for the requirement classification system.

C. Class Diagram

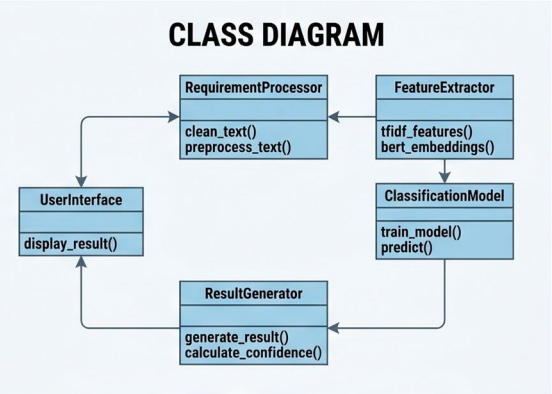


Figure 4: Class diagram showing the RequirementProcessor, FeatureExtractor, ClassificationModel, ResultGenerator, and UserInterface classes.

D. Activity Diagram

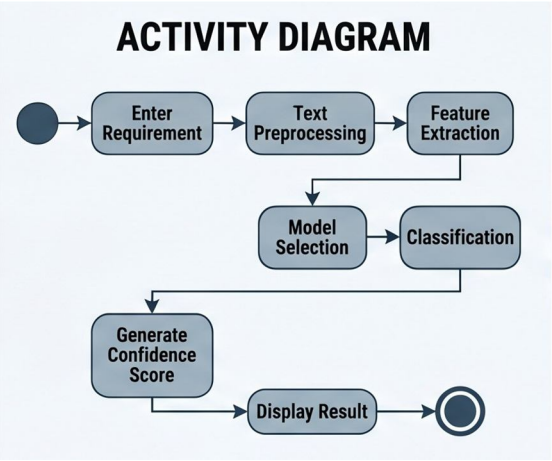


Figure 5: Activity diagram describing the end-to-end classification workflow.

E. Sequence Diagram

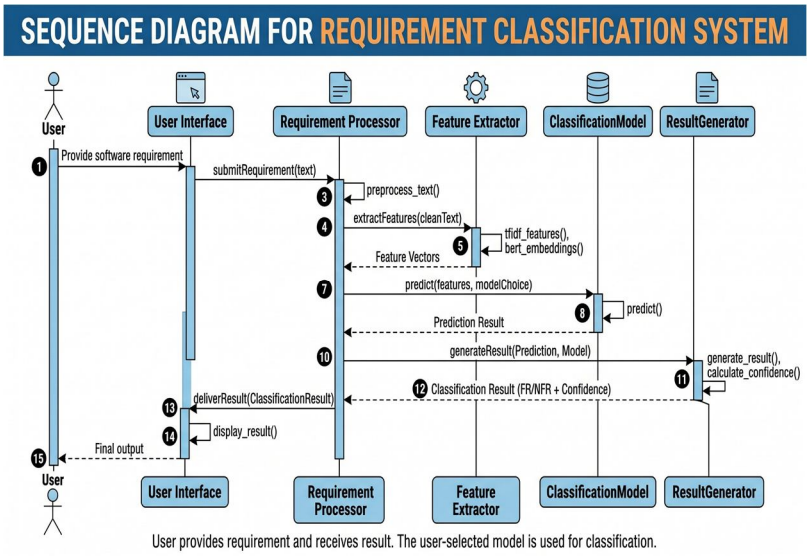


Figure 6: Sequence diagram showing module-to-module interaction during a single classification request.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VII July 2026- Available at www.ijraset.com

V. IMPLEMENTATION

The system is implemented in Python 3.11. Pandas and NumPy handle dataset loading and numerical operations; NLTK performs tokenisation and text cleaning; Scikit-Learn provides the TF-IDF vectoriser together with the Logistic Regression, SVM, and Random Forest classifiers; Hugging Face Transformers and PyTorch load the pre-trained BERT model and compute contextual embeddings; Joblib serialises trained models to disk; Matplotlib produces confusion matrices and classification-report plots; and Gradio renders the interactive web interface. Table 2 summarises the software and hardware environment used for development and evaluation.

Table 2: Software and Hardware Environment

Component	Specification
Operating System	Windows 10 / 11
Programming Language	Python 3.11
ML / NLP Libraries	Scikit-Learn, NLTK
Deep Learning Libraries	PyTorch, Hugging Face Transformers
User Interface	Gradio
Processor	Intel Core i5 or higher
RAM	8 GB minimum
Storage	20 GB minimum

The overall workflow proceeds as follows: the dataset is gathered and cleaned; requirement text is pre-processed; TF-IDF and BERT features are extracted; the four models are trained; each model is evaluated against held-out test data; trained models are serialised for reuse; and the Gradio interface is connected to the saved models so that a user-submitted requirement can be classified in real time with its associated confidence score.

VI. EXPERIMENTAL RESULTS AND DISCUSSION

The dataset was split into training and test partitions, and each model was evaluated on the held-out test set of 1,218 requirement statements, including 793 FR and 425 NFR statements. Accuracy, precision, recall, and F1-score were used as evaluation metrics. Table 3 reports macro-averaged results for all four models.

Table 3: Performance Comparison of Classification Models

Model	Acc.	Prec.	Rec.	F1
Logistic Regression (TF-IDF)	0.84	0.85	0.80	0.82
SVM (TF-IDF)	0.86	0.85	0.83	0.84
Random Forest (TF-IDF)	0.84	0.84	0.81	0.82
BERT + SVM	0.86	0.85	0.84	0.84

Logistic Regression achieves the highest recall on the FR class, 0.94, but is noticeably weaker at recognising NFR statements, with a recall of 0.66. This suggests that a purely linear decision boundary over TF-IDF features under-fits the more varied vocabulary used in non-functional requirements. SVM and Random Forest both improve NFR recall over Logistic Regression while keeping FR performance largely intact, and SVM achieves the best overall accuracy among the TF-IDF-based models. The BERT + SVM hybrid matches SVM's overall accuracy while further improving NFR recall to 0.76, consistent with the expectation that contextual embeddings help distinguish NFR statements that share surface vocabulary with functional ones. Overall, the results support the conclusion drawn from the literature in Section II: TF-IDF-based classical models remain a strong, inexpensive baseline, while BERT-based representations provide a measurable, if incremental, improvement in handling the harder non-functional class [9, 10].
Fig. 7 shows the Gradio interface used to interact with the trained models. Figs. 8 and 9 show representative functional and non-functional predictions. Figs. 10–13 show the classificationreport outputs for the evaluated models.

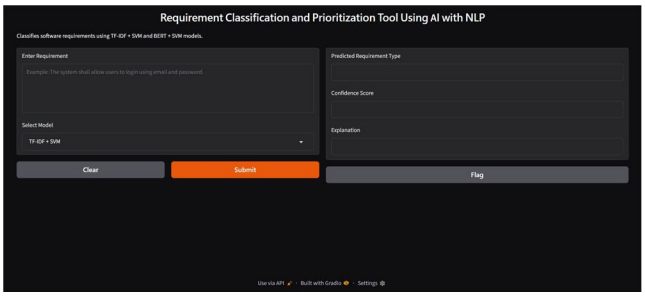


Figure 7: Gradio interface for real-time requirement classification.

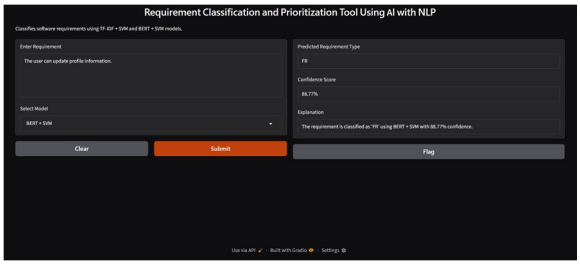


Figure 8: Sample prediction for a functional requirement.

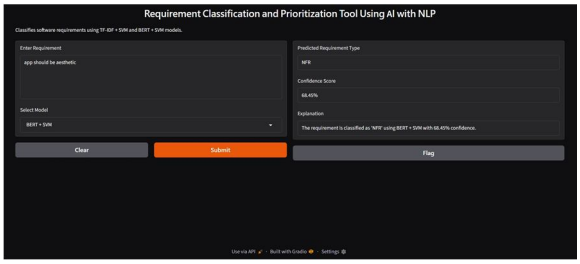


Figure 9: Sample prediction for a non-functional requirement.

Training TFIDF_Logistic_Regression...

	precision	recall	f1-score	support
FR	0.84	0.94	0.89	793
NFR	0.86	0.66	0.75	425
accuracy			0.84	1218
macro avg	0.85	0.80	0.82	1218
weighted avg	0.85	0.84	0.84	1218

Figure 10: Classification report for Logistic Regression using TF-IDF features.

	precision	recall	f1-score	support
FR	0.87	0.92	0.89	793
NFR	0.84	0.74	0.79	425
accuracy			0.86	1218
macro avg	0.85	0.83	0.84	1218
weighted avg	0.86	0.86	0.86	1218

Figure 11: Classification report for SVM using TF-IDF features.

```
Training TFIDF_Random_Forest...
              precision    recall  f1-score   support

          FR       0.85      0.92      0.88       793
         NFR       0.83      0.70      0.76       425

    accuracy                           0.84      1218
   macro avg       0.84      0.81      0.82      1218
weighted avg       0.84      0.84      0.84      1218
```

Figure 12: Classification report for Random Forest using TF-IDF features.

```
              precision    recall  f1-score   support

          FR       0.88      0.91      0.89       793
         NFR       0.82      0.76      0.79       425

    accuracy                           0.86      1218
   macro avg       0.85      0.84      0.84      1218
weighted avg       0.86      0.86      0.86      1218
```

Figure 13: Classification report for the BERT + SVM model.

VII. SOFTWARE TESTING

Each module of the tool was tested individually through unit testing and in combination through integration testing to confirm that pre-processing, feature extraction, classification, and result generation operate correctly together. Functional testing verified that requirement statements are accepted, processed, and classified with a confidence score, and performance testing confirmed that predictions are returned within a short, near-instantaneous response time. Table 4 lists a representative subset of the test cases executed against the Gradio interface, all of which passed.

Table 4: Representative Test Cases

Test ID	Description	Expected Result
TC-01	Valid functional requirement	Predicted as FR
TC-02	Valid non-functional requirement	Predicted as NFR
TC-03	Empty input	Validation message displayed
TC-04	Input with special characters	Processed without error
TC-05	Multiple sequential predictions	Correct independent output each time

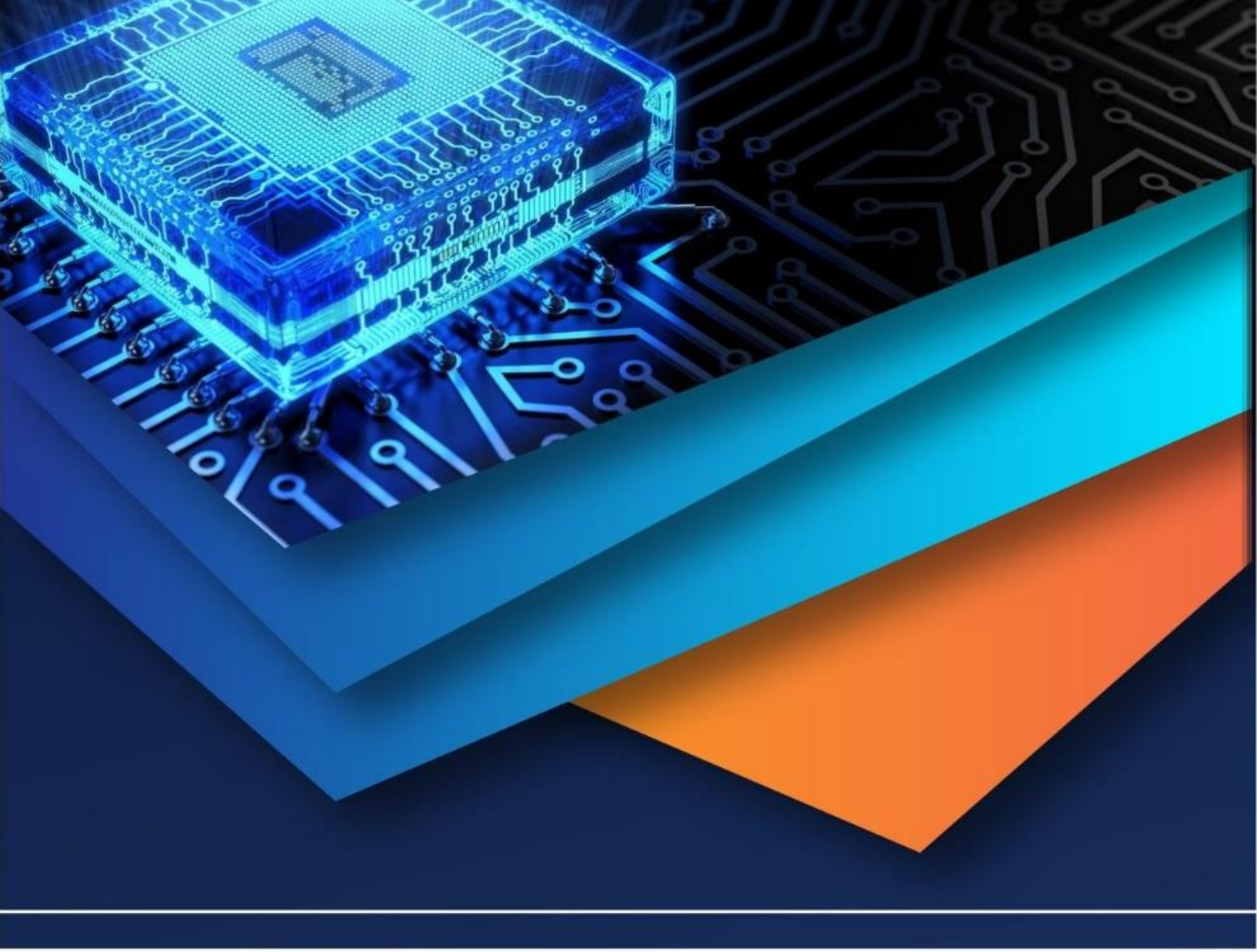
VIII. CONCLUSION AND FUTURE SCOPE

This paper presented a Requirement Classification and Prioritization Tool that applies NLP pre-processing together with TF-IDF and BERT feature extraction to automatically separate software requirements into Functional and Non-Functional categories. Four classifiers, Logistic Regression, SVM, Random Forest, and BERT + SVM, were trained and compared, and the results show that all four achieve usable accuracy, with SVM and BERT + SVM offering the best balance between functional and non-functional recall. A Gradio-based interface makes the tool directly usable by software analysts and project managers, returning a prediction, a confidence score, and a short explanation in real time. Future work could extend the tool with an intelligent prioritisation module that ranks requirements into High, Medium, and Low priority; refine NFR predictions into finer subcategories such as performance, security, and usability; incorporate explainable-AI techniques to highlight the words that most influenced a prediction; evaluate

newer transformer variants such as RoBERTa or DeBERTa; and integrate the tool with industrial requirement-management platforms such as Jira or IBM DOORS, potentially exposing it as a cloud-hosted REST
API for use within larger software-engineering workflows [4, 5].

REFERENCES

[1] E. D. Canedo and B. C. Mendes, "Software Requirements Classification Using Machine Learning Algorithms," Entropy, vol. 22, no. 9, p. 1057, 2020.
[2] G. Airlangga, "Enhancing Software Requirements Classification with Semisupervised GAN-BERT Technique," Journal of Electrical and Computer Engineering, vol. 2024, art. 4955691, 2024.
[3] F. Khayashi, B. Jamasb, R. Akbari, and P. Shamsinejadbabaki, "Deep Learning Methods for Software Requirement Classification: A Performance Study on the PURE Dataset," arXiv preprint arXiv:2211.05286, 2022.
[4] M. Binkhonain and L. Zhao, "A Machine Learning Approach for Hierarchical Classification of Software Requirements," arXiv preprint arXiv:2302.12599, 2023.
[5] 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.
[6] Z. Kurtanovic and W. Maalej, "Automatically Classifying Functional and´ Non-Functional Requirements Using Supervised Machine Learning," in Proc. IEEE 25th Int. Requirements Engineering Conf. (RE), 2017, pp. 490–495.
[7] L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E.
Chioasca, and R. T. Batista-Navarro, "Natural Language Processing for Requirements Engineering: A Systematic Mapping Study," ACM Computing Surveys, vol. 54, no. 3, pp. 1–41, 2021.
[8] Z. Abad, K. Shakeri Hossein, O. Karras, P. Ghazi, M. Glinz, G. Ruhe, and K. Schneider, "What Works Better? A Study of Classifying Requirements," arXiv preprint arXiv:1707.02358, 2017.
[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in Proc. NAACL-HLT, 2019.
[10] S. Eltahier, "BERT Fine-Tuning for Software Requirement Classification," Information, vol. 16, no. 11, p. 981, 2025.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY