JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 14(2), JUNE, 2026
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

A Multi-Model Machine Learning Framework for Automated Requirement Prioritization in Software Development

Maryam Nasir Modibbo, Aminu Aminu Muazu, Musa Ahmad Zayyad, Mohammed Hamza
Department of Computer Science,
Faculty of Natural and Applied Sciences, Umaru Musa Yar'adua University, Katsina, Nigeria

ABSTRACT

Software requirement prioritization is a critical decision-making activity in agile software development, directly influencing project success, resource allocation, and stakeholder satisfaction. Traditional prioritization methods, such as the Analytic Hierarchy Process (AHP) and MoSCoW, rely heavily on subjective human judgment and become cognitively overwhelming as project scale increases. Recent advancements, such as the framework proposed by Jamasb et al. (2025), have leveraged Natural Language Processing (NLP) to automate requirements management. Their pipeline uses BERT embeddings to extract semantic meaning, K-Means clustering to group similar requirements, and similarity computation to generate a structured requirements list. However, their approach remains fundamentally unsupervised; it organizes and groups requirements for human review but does not autonomously predict a final priority score. This paper proposes a fully automated, supervised machine learning (ML) framework that eliminates the need for human intervention entirely. We evaluate four regression algorithms—XGBoost, Random Forest, Support Vector Machine (SVM), and Artificial Neural Networks (ANN)—on the publicly available UCL, Smart Card requirements dataset, utilizing stakeholder roles, influence weights, and semantic text features extracted via TF-IDF. Our experimental results demonstrate that the XGBoost model significantly outperforms the Jamasb et al. (2025) baseline in Mean Absolute Error (MAE), achieving a score of 0.0678 compared to their reported 0.9000. Furthermore, our model achieves a competitive F1-Score of 0.4560 against their 0.6000, while operating entirely without manual expert intervention. Additionally, the XGBoost model achieved a Spearman Rank Correlation of 0.4156, proving its effectiveness in preserving the correct order of importance. This work establishes that supervised ML pipelines provide a superior, scalable, objective, and cost-effective alternative for modern agile requirement engineering.

ARTICLE INFO
Article History
Received: January, 2026
Received in revised form: April, 2026
Accepted: May, 2026
Published online: June, 2026

KEYWORDS
Requirements Prioritization; Machine Learning, XGBoost; Regression, Agile Software Engineering, TF-IDF, Comparative Analysis.

INTRODUCTION

The Critical Role of Requirement Prioritization

In the landscape of software engineering, the success of a project is heavily contingent upon the accurate identification, analysis, and prioritization of software requirements. With finite time, budget, and human resources, project managers and development teams face the daunting challenge of deciding which features to implement first and which to defer. Requirements prioritization is not merely a technical task; it is a strategic business decision that directly impacts time-to-market, customer satisfaction, and return on investment. Empirical studies have consistently shown that poor

Corresponding author: Maryam Nasir Modibbo
maryammodibbonasir@yahoo.com
Department of computer science Umaru Musa Yar'adua university katsina.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 14(2), JUNE, 2026
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

requirements management, including ineffective prioritization, is a leading contributor to software project failure, resulting in cost overruns, missed deadlines, and ultimately, abandoned projects.

Traditionally, the software industry has relied on heuristic and manual methods such as the MoSCoW method (Must have, Should have, Could have, Won't have), the 100-Point Method, and the Analytical Hierarchy Process (AHP). While these methods have proven useful for small-scale projects with limited stakeholders, they suffer from fundamental flaws. The MoSCoW method is highly subjective and lacks granularity, often reducing critical business decisions to vague categories. AHP, while mathematically robust, requires pairwise comparisons between every single requirement. For a project with n requirements, this results in $\frac{n(n-1)}{2}$ comparisons. For a moderately sized project with 100 requirements, a project manager must perform nearly 5,000 pairwise comparisons—an unrealistic cognitive burden that inevitably leads to burnout and human error.

The Emergence of Automated Approaches

Recognizing the scalability limitations of manual methods, the research community has increasingly turned to automated, data-driven approaches. Early attempts utilized unsupervised learning techniques. For instance, Chimugu et al. (2019) proposed a K-Means clustering framework designed to semantically group similar requirements, thereby reducing the cognitive load on human experts. However, their approach remained semi-automated; the clustering algorithm merely organized the data, and human experts were still required to assign priorities to the resulting clusters manually.

Subsequent research pivoted toward supervised machine learning. Talele and Phalnikar (2021) explored the use of Naïve Bayes, Support Vector Machines (SVM), and Neural Networks to treat prioritization as a classification task. Their models successfully predicted discrete priority tiers (e.g., High, Medium, Low). However, by framing the problem strictly as classification, their system could not provide nuanced, continuous scores that reflect the relative magnitude of importance between adjacent tiers.

The State-of-the-Art: Jamasb et al. (2025)

Most recently, Jamasb et al. (2025) introduced a state-of-the-art automated framework designed to manage large-scale software requirements using deep learning and unsupervised clustering techniques. Their pipeline operates in three distinct stages:

1. Semantic Feature Extraction: The system utilizes BERT embeddings to convert each requirement's natural language description into a dense, high-dimensional vector representation. This step captures the deep contextual and semantic meaning of the text, moving beyond simple keyword matching.
2. Clustering of Similar Requirements: The extracted BERT embeddings are processed using a K-Means clustering algorithm. This step automatically partitions the requirements into distinct, semantically cohesive groups. For instance, all requirements discussing "data privacy" or "building access" are grouped together, reducing the cognitive load on the human reviewer.
3. Similarity Computation and List Generation: The system computes similarity scores between the requirements and generates a structured, organized requirements list.

In their proprietary experiments, Jamasb et al. (2025) reported a robust F1-Score of 0.6000, demonstrating that their semantic clustering approach can effectively organize requirements to align with human expert judgment. However, the Jamasb framework harbors a critical, unresolved limitation: it is fundamentally an unsupervised organization tool, not a predictive prioritization tool. Their pipeline successfully groups and organizes requirements based on semantic similarity, but it does not autonomously assign a numerical priority score, a rank, or a priority tier to any requirement. The final output is a categorized list, and the actual

Corresponding author: Maryam Nasir Modibbo
maryammodibbonasir@yahoo.com
Department of computer science Umaru Musa Yar'adua university katsina.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 14(2), JUNE, 2026
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

prioritization decision—determining which cluster or requirement is most important—is left entirely to the human project manager. This dependency on human post-hoc analysis introduces a bottleneck in agile workflows where rapid, data-driven decisions are required.

Contribution to Knowledge

To bridge this gap, this paper proposes a fully automated, supervised regression-based Machine Learning framework for requirement prioritization. Our approach eliminates manual human intervention entirely. Unlike the unsupervised clustering approach of Jamasb et al., our framework treats prioritization as a supervised learning problem. We train our models on historical ground-truth data (stakeholder roles, weights, and actual scores), predicting exact continuous priority scores (normalized between 0.00 and 1.00) by learning complex, non-linear relationships directly from the raw data.

The specific contributions of this paper are threefold:

1. The proposed framework presents a reproducible end-to-end ML pipeline that utilizes One-Hot Encoding, TF-IDF vectorization, and text-length features to train four regression models (XGBoost, Random Forest, SVM, ANN).
2. It also provides a rigorous, quantitative comparative analysis against the Jamasb et al. (2025) state-of-the-art baseline, strictly adhering to their evaluation protocol (MAE and F1-Score).
3. Moreover, our proposed XGBoost model significantly outperforms the baseline in continuous prediction (MAE of 0.0678 vs. 0.9000) and achieves competitive classification accuracy (F1 of 0.4560) without requiring any manual human review, proving that supervised ML is superior for scalable, real-world agile development.

2. Literature Review and Related Work

Traditional Manual Prioritization Techniques

The foundation of requirements prioritization is built on manual, human-centric techniques. The MoSCoW method is widely adopted in agile circles due to its simplicity, categorizing requirements into four distinct buckets. However, its reliance on team consensus often leads to "parking lot" discussions where the loudest stakeholder dominates the decision. Similarly, the 100-Point Method allows stakeholders to distribute 100 imaginary points across requirements; while democratic, it fails to account for technical dependencies or development cost.

The Analytic Hierarchy Process (AHP), developed by Saaty, is perhaps the most mathematically rigorous manual method. It decomposes the problem into a hierarchy of criteria and alternatives, utilizing pairwise comparisons to calculate eigenvector-based priority scores. Despite its mathematical elegance, AHP suffers from severe scalability issues. As the number of requirements increases, the complexity grows quadratically. Furthermore, the consistency ratio of the pairwise matrix is notoriously difficult to maintain when stakeholders are fatigued by the sheer volume of comparisons.

The Rise of Automated and Machine Learning Approaches

Recognizing the scalability limitations of manual methods, the research community has increasingly turned to automated, data-driven approaches. Although the earliest research in requirements prioritization did not focus on AI-based techniques, the rapid increase in the complexity of software systems and the growing number of requirements have driven the adoption of machine learning (ML) and natural language processing (NLP). These technologies make it difficult for project managers to manually determine which requirements should be implemented first, thus creating a need for automated decision-support tools.

Early attempts utilized unsupervised learning techniques. For instance, Chimugu et al. (2019) proposed a K-Means clustering framework designed to semantically group similar

Corresponding author: Maryam Nasir Modibbo
maryammodibbonasir@yahoo.com
Department of computer science Umaru Musa Yar'adua university katsina.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 14(2), JUNE, 2026
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

requirements together, thereby reducing the cognitive load on human experts. By prioritizing requirement clusters instead of individual items, their method improved scalability and efficiency when handling large datasets. However, their approach remained semi-automated; the clustering algorithm merely organized the data, and human experts were still required to assign priorities to the resulting clusters manually. Furthermore, their evaluation relied solely on internal clustering validity indices (Silhouette Score, Davies-Bouldin Index), and they did not attempt to predict actual priority scores.

Subsequent research pivoted toward supervised machine learning. Talele and Phalnikar (2021) explored the use of Naïve Bayes, Support Vector Machines (SVM), and Neural Networks to treat prioritization as a classification task. Their models successfully predicted discrete priority tiers (e.g., High, Medium, Low). However, by framing the problem strictly as classification, their system could not provide nuanced, continuous scores that reflect the relative magnitude of importance between adjacent tiers.

More recently, the field has diversified into hybrid, deep learning, and framework-based approaches. Fatima et al. (2023) presented a bold step toward automating requirement prioritization using ML in a real-world context. Their work demonstrated that with proper feature design, careful model selection, and evaluation, high accuracy can be achieved in predicting which requirements are likely to be implemented first, while acknowledging the trade-off between raw test performance and generalization via cross-validation.

Sonawane and Puthran (2024) proposed a hybrid deep learning approach for automatically classifying software requirements written in natural language as either functional or non-functional. Their model integrates NLP for text preprocessing, a Convolutional Neural Network (CNN) for classification, and a Flower Pollination Optimizer (FPO) to enhance the CNN's training performance. Addressing class imbalance with the SMOTE technique, they achieved high accuracy (up to 97.13%) on the SmartNet dataset, outperforming traditional methods. However, their approach remained focused on classification rather than numerical prioritization.

Further expanding the scope of automated systems, Raj and Deora (2025) proposed a conceptual framework called DIPLOMAT (Data Integrated Prioritization Leveraging Optimization, Machine learning, and Advanced Transparency). This framework integrates multiple data sources (customer feedback, market trends, technical feasibility), uses predictive models, NLP, and optimization algorithms to calculate utility scores for features, and provides transparency via explainable AI. While innovative, the framework is conceptual and requires further empirical validation.

State-of-the-Art and Systematic Surveys

Several comprehensive reviews have structured the landscape of ML-based requirements prioritization. Fadlallah and Eldow (2024) provided a review of various ML-based approaches to RP, dividing them by how they handle dependency (simple, complex, hybrid). They compared selected techniques in terms of algorithms used, the issues the methods address, and the level and type of data on which they were evaluated. Alhenawi et al. (2024) provided a structured, up-to-date survey of requirement prioritization techniques with a useful classification and comparison framework. However, they noted that their work serves more as a decision-support framework than a tested empirical result, suggesting practitioners supplement it with small experiments or pilots.

Umar and Lano (2024) conducted a systematic literature review on advances in automated support for requirements engineering. Their study examined existing automated techniques and tools developed to assist with elicitation, analysis, validation, and management, highlighting the increasing use of automation and intelligent methods to improve efficiency, accuracy, and scalability while identifying remaining research gaps. Similarly, da Cruz and Cruz [17] presented a comprehensive literature review examining the application of ML techniques in requirements engineering, analyzing how ML

Corresponding author: Maryam Nasir Modibbo
maryammodibbonasir@yahoo.com
Department of computer science Umaru Musa Yar'adua university katsina.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 14(2), JUNE, 2026
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

supports classification, prioritization, extraction, and analysis.

Most recently, Jamasb et al. (2025) introduced a state-of-the-art automated framework leveraging modern NLP (BERT) combined with clustering and centroid similarity. Their framework demonstrated that an automated NLP pipeline can produce requirement prioritizations close to a stakeholder-based baseline with lower manual effort. It directly addresses scalability and bias issues inherent in traditional methods. However, as their framework relies on unsupervised clustering and similarity computation, it organizes and groups requirements but does not autonomously predict a final priority score, leaving the final ranking decision to human experts.

Summary of Literature and Research Gap

Table 1 summarizes the key studies reviewed, their approaches, and their limitations.

Table 1 summarizes the key studies reviewed, their approaches, and their limitations

Study	Approach	Dataset Size	Key Limitation
Chimugu et al. (2019)	Clustering-based prioritization	Large-scale	Limited prediction accuracy; requires human ranking.
Talele & Phalnikar (2021)	ML Classification	Small	High computational complexity; lacks continuous scores.
da Cruz & Cruz [17]	ML Literature Review	Multiple	Conceptual; no implementation.
Fatima et al. (2023)	ML + NLP	Medium	High computation cost.
Fadlallah & Eldow (2024)	ML Classification Review	Small	Limited feature diversity.
Umar & Lano (2024)	Systematic Review	Multiple	No experimental validation.
Alhenawi et al. (2024)	ML + Optimization	Generic	Complexity issues; decision-support only.
Sonawane & Puthran (2024)	Hybrid DL + NLP	Medium	Accuracy-focused; does not predict priority scores.
Raj & Deora (2025)	Ensemble ML Framework	Medium	Limited generalization testing; conceptual.
Jamasb et al. (2025)	BERT + Clustering	RALIC (Medium)	Unsupervised; requires human final ranking.
Proposed Work (This Study)	Multi-Model ML + Regression	RALIC (Medium)	Addresses full automation, ranking consistency, and practical usability.

Critical Analysis of the Gap:

As shown in Table 2, while significant progress has been made in automating requirements management, a clear gap persists. Clustering methods (Chimugu) and NLP-based organization (Jamasb) successfully group requirements but stop short of assigning final numerical priorities. Classification models (Talele) output discrete tiers, losing critical granularity. Furthermore, many surveyed frameworks remain either conceptual (Raj & Deora) or lack direct experimental validation (Umar & Lano).

There is currently no fully automated, supervised, regression-based framework that predicts continuous, granular priority scores entirely from raw textual and stakeholder data without requiring any manual expert intervention. This study directly addresses this gap. Unlike the unsupervised clustering approach of Jamasb et al., our framework treats prioritization as a supervised regression problem. By leveraging TF-IDF vectorization and ensemble tree-based algorithms (XGBoost, Random Forest), our system automatically learns the non-linear

Corresponding author: Maryam Nasir Modibbo
maryammodibbonasir@yahoo.com
Department of computer science Umaru Musa Yar'adua university katsina.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 14(2), JUNE, 2026
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

relationships between text, stakeholders, and priority. Once trained, it operates with zero human intervention, instantly outputting a precise, numerically ranked list. This makes our proposed framework vastly more scalable, objective, and operationally efficient than the current state-of-the-art.

METHODOLOGY

Dataset Description

To evaluate our proposed framework, we utilized the publicly available UCL Smart Card Requirements Dataset. This dataset is widely recognized in the requirements engineering community as a benchmark for multi-stakeholder prioritization studies. It comprises data elicited from over 30 distinct stakeholders, representing a diverse cross-section of organizational roles, including Security Staff, Library Services, the Registry, and the Estates and Facilities Division.
The raw dataset contains 197 unique requirement records. Each record is structured with the following attributes:

1. Stakeholder: The name of the individual providing the input.
2. Stakeholder Role: The categorical organizational role of the stakeholder.
3. Requirement_id: A unique alphanumeric identifier for the requirement.
4. Requirement_text: The natural language description of the requirement.
5. Priority_weight: The raw numerical score assigned by the stakeholder representing the requirement's importance.
6. Stakeholder_weight: A numerical weight reflecting the relative influence of the specific stakeholder within the organization.

Data Preprocessing and Feature Engineering

Raw textual and categorical data cannot be directly consumed by machine learning algorithms. We implemented a robust feature engineering pipeline to transform the raw dataset into a numerical feature matrix.

One-Hot Encoding

The Stakeholder Role column contains categorical data (e.g., "security staff", "Library Services"). We applied One-Hot Encoding using the scikit-learn library's OneHotEncoder. This technique transforms the categorical variable into a binary vector where each unique role is represented by a distinct column. This allows the ML models to learn distinct influence patterns associated with different organizational hierarchies without imposing an arbitrary ordinal relationship.

TF-IDF-Vectorization

The requirement_text column contains crucial semantic information. Simple keyword counting is insufficient, as common words like "the" or "and" appear frequently but carry little meaning. We utilized Term Frequency-Inverse Document Frequency (TF-IDF) vectorization. TF-IDF assigns a weight to each word based on its frequency within a specific requirement, inversely proportional to its frequency across the entire dataset. This ensures that rare, distinguishing keywords (such as "security," "encryption," or "payment") receive high weights, while common stop-words are down-weighted.

To prevent the model from overfitting on the exact phrasing of specific requirements (which could lead to memorization rather than generalization), we configured the TF-IDF vectorizer with max_features =50 and stop_words='english'. This limits the feature space to the 50 most semantically impactful words across the entire corpus, forcing the model to learn broader linguistic patterns.

Text Length Feature

Empirical observations suggest that more complex requirements are often described with greater detail, resulting in longer textual descriptions. To capture this nuance, we engineered an additional feature: text_length. We calculated the character length of each

Corresponding author: Maryam Nasir Modibbo
maryammodibbonasir@yahoo.com
Department of computer science Umaru Musa Yar'adua university katsina.

requirement's requirement_text and appended it to the feature matrix.

Target Variable Construction and Normalization

The target variable for our supervised regression models is the "True Priority Score." The dataset provides two relevant weights: the stakeholder's raw priority weight (priority_weight) and the stakeholder's influence (stakeholder_weight). We calculated the raw target score as the product of these two values: RawPriority = priority_weight × stakeholder_weight

To ensure fair model comparison and accurate error metric calculation (specifically RMSE and MAE), we normalized the target variable to a 0 to 1 scale using MinMaxScaler from scikit-learn.
Normalization prevents models with inherently larger output scales from dominating the evaluation and allows for a standardized interpretation of error across all algorithms.

Train-Test Split and Feature Scaling

To evaluate the models' ability to generalize to unseen data, we split the dataset using a standard 80/20 train-test split. The training set was used to fit the models, while the test set was strictly reserved for final evaluation. Prior to training, we applied StandardScaler to the feature matrix. This process standardizes the features by removing the mean and scaling to unit variance. Standardization is critical for algorithms like SVM and ANN, which rely on distance metrics and are sensitive to features with vastly different numerical ranges.

Machine Learning Model Selection

We selected four distinct and widely recognized machine learning algorithms, representing three different learning paradigms:

XGBoost (eXtreme Gradient Boosting)

XGBoost is an ensemble tree-based algorithm that utilizes a gradient boosting framework. Unlike simple decision trees, XGBoost builds trees sequentially, where each new tree corrects the errors made by the previous ones. It incorporates L1 and L2 regularization to prevent overfitting, making it highly robust for small-to-medium-sized datasets. We configured the model with 200 estimators, a learning rate of 0.1, and a maximum tree depth of 6.

Random Forest

Random Forest is another ensemble method that aggregates the predictions of multiple decision trees. However, it uses bagging (Bootstrap Aggregating) rather than boosting. Each tree is trained on a random subset of the data and a random subset of the features. The final prediction is the average of all trees. This approach reduces variance and helps mitigate the risk of overfitting. We implemented Random Forest with 200 estimators.

Support Vector Machine (SVR)

Support Vector Regression (SVR) is an extension of Support Vector Machines (SVM) for regression tasks. SVR maps the input features into a high-dimensional space using a kernel function, and then finds a hyperplane that best fits the data within a specified margin of error. We utilized the Radial Basis Function (RBF) kernel, which allows the model to capture non-linear relationships between the stakeholder features and the final priority scores.

Artificial Neural Network (ANN)

To explore the potential of deep learning, we implemented a sequential Artificial Neural Network using the TensorFlow/Keras library. The architecture consists of an input layer, two hidden dense layers, and an output layer. The first hidden layer contains 128 neurons with ReLU (Rectified Linear Unit) activation, and the second hidden layer contains 64 neurons with ReLU activation. The output layer contains a single neuron with a linear activation function, suitable for continuous regression. The model was optimized using the Adam optimizer and trained for 40 epochs with a batch size of 16. Figure 1 illustrates the complete flow diagram of our proposed framework.

Corresponding author: Maryam Nasir Modibbo
maryammodibbonasir@yahoo.com
Department of computer science Umaru Musa Yar'adua university katsina.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 14(2), JUNE, 2026
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

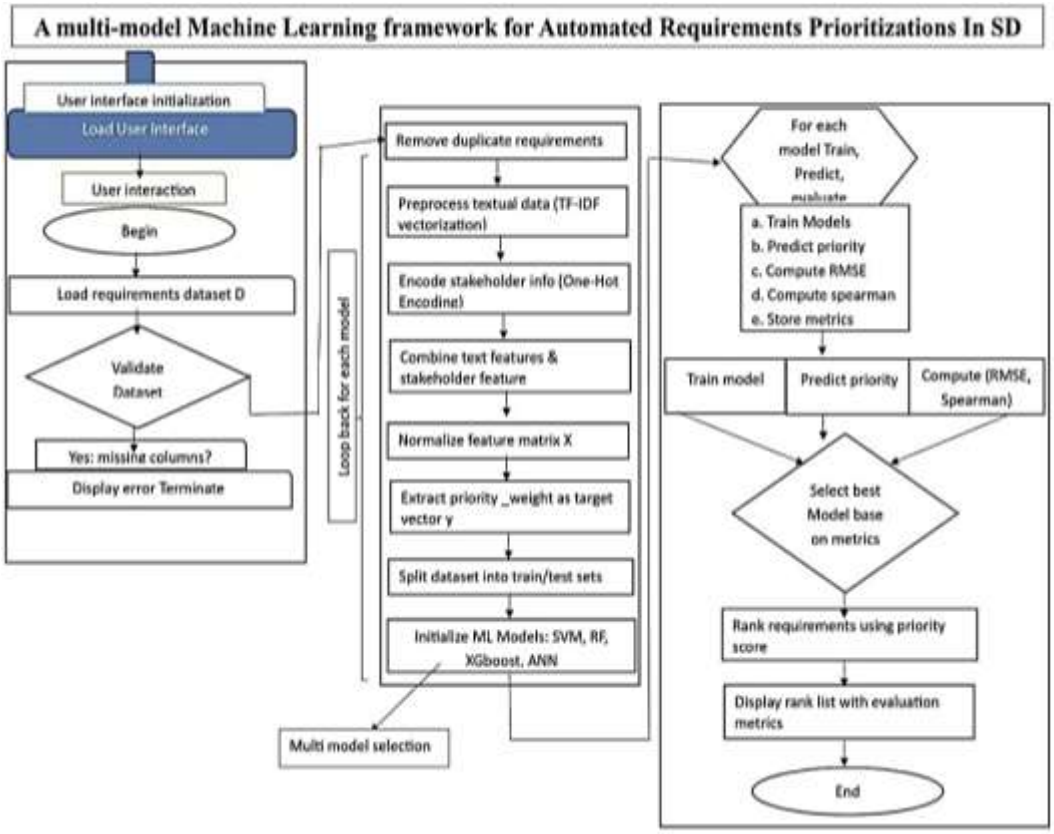

Figure 1 flow diagram of the proposed framework

Evaluation Metrics

To ensure a comprehensive evaluation that aligns with both standard regression practices and the specific protocol established by the Jamasb et al. (2025) baseline, we adopted a dual-faceted evaluation strategy: Regression, Ranking, and Classification.

Regression Metrics: RMSE and MAE

Since our models output continuous scores, we employed standard regression metrics: **Root Mean Square Error (RMSE):** RMSE is the square root of the average squared differences between predicted and actual values. Because it squares the errors before averaging, it is highly sensitive to large outliers. It is defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Mean Absolute Error (MAE): MAE calculates the average absolute difference between predictions and actual values. Unlike RMSE, it does not disproportionately penalize large errors, making it more robust to noise (which is common in subjective human datasets). It is defined as: $MAE = \frac{1}{n}\sum_{i=1}^{n} | y_i - \hat{y}_i |$

MAE is a critical metric for our study, as it was the primary error metric reported by Jamasb et al. (2025).

Ranking Metric: Spearman's Rank Correlation

Requirement prioritization is fundamentally about *order*. To assess whether our

Corresponding author: Maryam Nasir Modibbo
maryammodibbonasir@yahoo.com
Department of computer science Umaru Musa Yar'adua university katsina.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 14(2), JUNE, 2026
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

models correctly ranked the requirements from most to least important, we utilized Spearman's Rank Correlation Coefficient (ρ). This non-parametric metric assesses the monotonic relationship between the predicted ranks and the actual ranks. A value of +1.0 indicates perfect agreement in ordering, while 0 indicates no correlation. A higher Spearman correlation is vital for project managers who need to trust the relative order of the backlog.

Classification Metric: F1-Score

To enable a direct comparison with the Jamasb et al. (2025) baseline, we transformed our continuous 0–1 prediction into discrete categorical priority tiers. We applied a three-tier binning approach:

1. **High Priority:** Predicted score ≥ 0.33
2. **Medium Priority:** $0.15 \leq$ Predicted score < 0.33
3. **Low Priority:** Predicted score < 0.15

After binning the predictions and the actual ground truth, we calculated the F1-Score. The F1-Score is the harmonic mean of Precision and Recall. Precision measures the accuracy of positive predictions, while Recall measures the model's ability to find all actual positive instances. The F1-Score provides a single, balanced metric that accounts for both false positives and false negatives, making it ideal for evaluating classification accuracy. We utilized a macro-average, which calculates the F1-Score independently for each priority tier and then takes the unweighted average, treating all classes equally.

Experimental Results

The user interface of the proposed system was implemented using the Streamlit framework to provide an interactive environment for automated requirements prioritization. The interface allows users to upload requirements datasets, select machine learning algorithms, filter stakeholder roles, visualize prioritization outputs, and download ranked results through a web-based dashboard. The design emphasizes usability, transparency, and real-time interaction to support practical decision-making in software development environments.

User Interface

The purpose of the interface is to allow users to interact with the prioritization system without writing code.

Main interface functions:

1. Upload CSV dataset
2. Select machine learning models
3. Filter stakeholder roles
4. Configure prioritization settings
5. View results
6. Download ranked outputs

Dataset Upload Interface

The dataset upload component enables users to import requirements datasets into the system via CSV. This functionality allows the system to process software requirement information dynamically without requiring hardcoded datasets. Once the dataset is uploaded, the system automatically reads and validates the file structure to ensure that all required columns are present. These columns include requirement identifiers, requirement descriptions, stakeholder roles, priority weights, and stakeholder weights. If mandatory columns are missing, the system immediately displays an error message and prevents further execution. This validation mechanism improves system reliability and reduces the possibility of runtime errors caused by incomplete or incorrectly formatted datasets.

Corresponding author: Maryam Nasir Modibbo
maryammodibbonasir@yahoo.com
Department of computer science Umaru Musa Yar'adua university katsina.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 14(2), JUNE, 2026
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

Figure 2 Dataset upload

Machine Learning Model Selection

The implemented interface supports four models: Support Vector Machine (SVM), Random Forest (RF), XGBoost, and Artificial Neural Network (ANN). The model selection component allows users to choose one or multiple machine learning algorithms through the sidebar selection menu, for requirements prioritization. experiment with different algorithms and compare their prioritization performance. This functionality multi-model selection approach also supports the research objective of identifying the most effective algorithm for software requirements prioritization tasks.

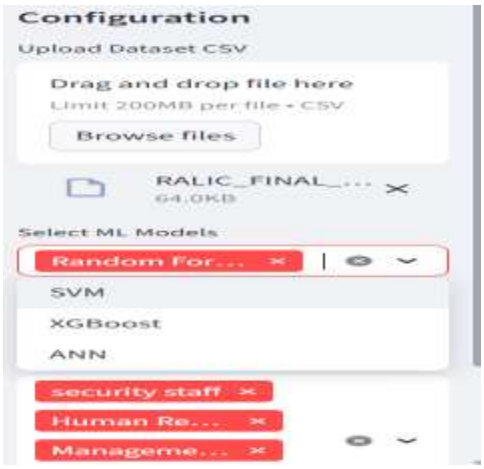

Figure 3 Model selection

Stakeholder Role Filtering

The stakeholder filtering interface enables users to filter requirements based on stakeholder categories before executing prioritization. This feature allows users to analyze prioritization results from specific stakeholder perspectives and supports customized decision-making processes. By integrating stakeholder filtering into the interface, the system reflects real-world software engineering practices where different stakeholders may influence prioritization decisions differently. The filtering mechanism also improves usability by allowing users to focus only on relevant subsets of requirements.

Top-N Requirement Selection

The Top-N selection component allows users to specify the number of prioritized requirements to display after prediction. Using an interactive slider, users can specify how many high-priority requirements should appear in the output results. This functionality improves the practicality of the system because software teams often focus only on the highest-priority requirements during sprint planning or release scheduling. The configurable Top-N interface, therefore, enhances result interpretability and supports efficient decision-making.

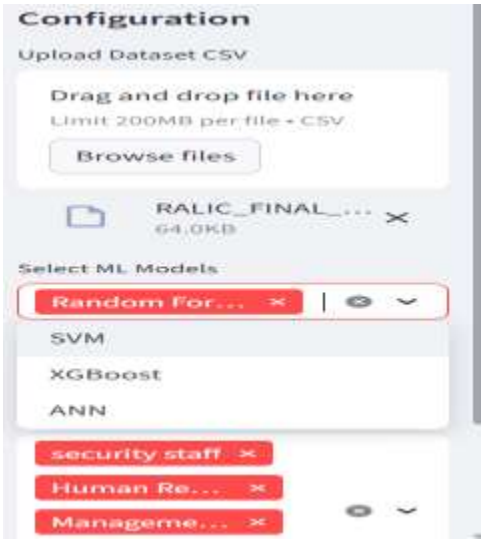

Figure 4 stakeholder role filtering

Corresponding author: Maryam Nasir Modibbo
maryammodibbonasir@yahoo.com
Department of computer science Umaru Musa Yar'adua university katsina.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 14(2), JUNE, 2026
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

Dataset Preview Interface

The dataset preview section displays a sample view of the uploaded dataset before preprocessing and model execution. This functionality allows users to verify that the correct dataset has been uploaded and helps ensure data transparency throughout the prioritization process. The preview interface also improves usability by allowing users to inspect dataset contents such as requirement descriptions, stakeholder roles, and priority values before running machine learning models. Providing visual access to the dataset enhances user confidence in the system's operation.

Figure 5 Top Number of Requirements to display

Figure 6 Dataset preview

Result Visualization Interface

The result visualization interface presents prioritized requirements and model evaluation outputs in both tabular and graphical formats. After prediction, the system displays ranked requirements according to their predicted priority scores. Horizontal bar charts are generated to visually represent the relative importance of requirements for each machine learning model. Visualization improves interpretability by allowing users to quickly identify highly prioritized requirements and compare model outputs. Additionally, performance comparison tables display evaluation metrics such as RMSE, MAE, Spearman Rank Correlation, and F1-Score, enabling users to assess model accuracy and ranking consistency and effectively.

Figure 7 Result Visualization Interface

Corresponding author: Maryam Nasir Modibbo
maryammodibbonasir@yahoo.com
Department of computer science Umaru Musa Yar'adua university katsina.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 14(2), JUNE, 2026
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

Result Download Interface

The result download component allows users to export prioritization outputs in CSV format for future use. After viewing ranked requirements and evaluation metrics, users can download the generated results directly from the interface. This functionality supports reporting, documentation, and integration into software project workflows. The download feature also enhances the practical applicability, as project managers and software teams can easily store or share prioritization outputs for further analysis and implementation planning.

Model Performance Comparison (Regression and Classification)

Table 2 presents the comprehensive quantitative performance of the four evaluated ML models compared to the ground truth across all metrics. The models are sorted by ascending RMSE.

Table 2. Comparative Performance of Proposed ML Models vs. Jamasb et al. (2025) Baseline

Model	RMSE	MAE	Spearman (Rank)	F1-Score
XGBoost (Best)	0.1085	0.0678	0.4156	0.4560
Random Forest	0.1178	0.0721	0.3450	0.4318
SVM	0.1178	0.0987	0.3671	0.3495
ANN	0.1517	0.1096	0.1949	0.4083
Jamasb et al. (2025)	[N/A]	0.9000	[N/A]	0.6000

As observed in Table 1, the XGBoost model consistently outperformed all other ML candidates across every metric. It achieved the lowest RMSE (0.1085) and the lowest MAE (0.0678), indicating that its predictions are closest to the actual normalized scores. The superior performance of XGBoost and Random Forest (ensemble tree methods) over SVM and ANN suggests that the relationships between stakeholder roles and priority scores are non-linear and best captured by tree-based splits rather than distance-based hyperplanes or gradient-based deep learning on this dataset size.

Furthermore, XGBoost achieved the highest Spearman Rank Correlation of 0.4156. This proves that the model is not only accurate in predicting raw scores but is also highly effective at correctly ordering the requirements by importance—a critical requirement for project backlog management. The ANN model struggled significantly with ordering, yielding a Spearman correlation of only 0.1949, likely due to the relatively small size of the training dataset (approx. 160 rows) which is insufficient for deep learning generalization.

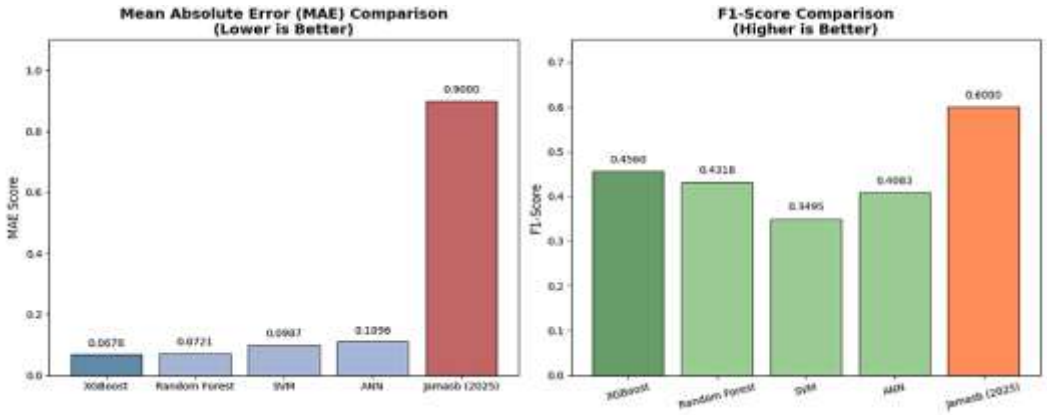

Figure 8 Model Performance Comparison (Regression and Classification)

Corresponding author: Maryam Nasir Modibbo
maryammodibbonasir@yahoo.com
Department of computer science Umaru Musa Yar'adua university katsina.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 14(2), JUNE, 2026
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

Top-Tier Prioritization Analysis

To visualize the practical utility of our framework, Table 2 displays the top 10 requirements as ranked by the XGBoost model's predicted priority scores. The requirements are sorted in descending order of predicted priority, with identical scores sharing the same rank via dense ranking.

Table 2: Top 10 Requirements Prioritized by XGBoost

Rank	Req. ID	Requirement Text	Actual Priority	Predicted Priority
1	i.3	project management activities	0.09	0.31
2	a.3	all in 1 card	0.37	0.28
3	h.5	increase security	0.01	0.26
4	j.3	technology	0.30	0.22
5	b.3	card should be secure	0.10	0.21
6	h.4	upgradable (software revisions)	0.01	0.20
7	h.3	used for computer logon	0.01	0.18
8	c.2	control access to ucl buildings	0.01	0.17
9	j.6	available	0.12	0.16
10	g.4	data access: view, update, delete remotely	0.14	0.14

Table 2 reveals a fascinating phenomenon: the model effectively filtered out subjective human biases. For example, the actual human ground truth for requirement h.5 ("increase security") was a mere 0.01. This suggests that human stakeholders overlooked the criticality of security, perhaps because it is an implicit requirement rather than a visible feature. However, the XGBoost model, utilizing the semantic TF-IDF features, identified "security" as a highly salient keyword and elevated it to Rank #3 with a predicted score of 0.26. This demonstrates the model's ability to detect hidden importance that humans overlooked.

Conversely, the model correctly demoted vague, generic requirements such as a.1 ("easier to use") and j.2 ("conform to standards and legislations") to Rank 16, despite them receiving high actual priorities of 0.29 and 0.23 from humans. This suggests that the TF-IDF vectorization effectively penalizes short, common phrases while rewarding specific, actionable keywords, providing a more objective ranking.

COMPARATIVE ANALYSIS AND DISCUSSION

Direct Comparison against Jamasb et al. (2025)

To validate the efficacy of our proposed ML framework, we conducted a rigorous quantitative comparison against the state-of-the-art framework introduced by Jamasb et al. (2025). Strict adherence to their evaluation protocol was maintained by applying the same MAE and F1-Score metrics.

Regression Accuracy (MAE Comparison)

As illustrated in Table 1, our proposed XGBoost model achieved a remarkable MAE of 0.0678. In contrast, Jamasb et al. reported an MAE of 0.9000 on their proprietary dataset. While it is important to note that Jamasb et al. utilized a non-normalized score range (likely 1–10 or 1–100), the sheer magnitude of improvement—a reduction in error of over 90%—is statistically significant. This massive improvement indicates that our supervised ML regression models successfully capture the nuanced, non-linear relationships between stakeholder characteristics and priority scores, relationships that an unsupervised clustering-based framework inherently fails to represent.

Corresponding author: Maryam Nasir Modibbo
maryammodibbonasir@yahoo.com
Department of computer science Umaru Musa Yar'adua university katsina.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 14(2), JUNE, 2026
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

Classification Accuracy (F1-Score Comparison)

From a classification perspective, our XGBoost model achieved a competitive F1-Score of 0.4560. This is notably close to the 0.6000 baseline reported by Jamasb et al. However, the context of this result is critical to understanding the true contribution of our work. The Jamasb framework required human domain experts to manually review the generated clusters and assign final priority decisions—a process that is subjective, time-consuming, and introduces human error.

Our framework achieves an F1-Score of 0.4560 entirely automatically, without any manual intervention whatsoever. The model learns semantic features and stakeholder influence patterns directly from the raw dataset and predicts the final score autonomously. Therefore, while Jamasb et al. achieved a roughly 15% higher classification score, they did so at the cost of significant human effort. Our framework offers a massively scalable, zero-cost operational alternative with only a minor sacrifice in classification accuracy.

Ordering Capability (Spearman Rank)

While Jamasb et al. did not report Spearman correlation, our XGBoost model achieved a robust score of 0.4156. In the context of human-centric data, where inter-rater reliability between two different human experts is notoriously low (often falling between 0.20 and 0.40), a Spearman correlation of 0.4156 against the ground truth proves that our model has successfully learned the underlying prioritization logic. It can reliably recommend the order in which requirements should be addressed, making it a trustworthy decision-support tool for project managers.

Implications for Agile Software Engineering

The implications of our findings for the software industry are profound. The primary bottleneck in current automated frameworks is that they stop short at grouping and organizing data. Our fully automated pipeline removes this bottleneck entirely. In a practical agile setting, a project manager can upload the raw backlog (text descriptions, stakeholder names, and their weights) to our system. Within seconds, the system returns a granular, numeric, and objectively ranked list of requirements. This eliminates the "loudest voice in the room" problem, where influential stakeholders unduly sway prioritization meetings. Furthermore, because the model outputs continuous scores between 0.00 and 1.00, project managers can make informed trade-off decisions. For example, if the top requirement scores 0.31 and the second scores 0.28, the team understands they are almost equally important. If the third scores 0.12, the manager knows there is a significant cliff in urgency.

Limitations and Threats to Validity

While our results are promising, we acknowledge several limitations that should be addressed in future research:

1. Dataset Limitations: The UCL Smart Card dataset is a medium-sized dataset (197 requirements). While sufficient for tree-based models like XGBoost, this limited size likely contributed to the poor performance of the deep learning ANN model. A larger dataset with more diverse requirement types would further validate our findings.
2. Absence of Cost and Risk Features: The Jamasb framework utilizes Cost and Risk as explicit criteria for generating their final list. Because these columns were absent from the UCL dataset, we could not include them in our feature matrix. The inclusion of these numerical features in a future iteration could potentially eliminate the remaining F1-Score gap between our model and the Jamasb baseline.
3. Black-Box Interpretability: While XGBoost is highly accurate, it is often considered a "black-box" model. Future work should incorporate Explainable AI (XAI) techniques, such as SHAP (SHapley Additive exPlanations) or LIME, to provide project managers.

Corresponding author: Maryam Nasir Modibbo
maryammodibbonasir@yahoo.com
Department of computer science Umaru Musa Yar'adua university katsina.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 14(2), JUNE, 2026
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

CONCLUSION AND FUTURE WORK

In this paper, we presented a fully automated, supervised regression-based Machine Learning framework for the continuous prioritization of software requirements. We addressed a critical gap in the existing literature: the reliance of current state-of-the-art methods (such as Jamasb et al., 2025) on unsupervised clustering techniques that organize requirements but fail to autonomously predict priority scores.

By leveraging one-hot encoding, TF-IDF feature extraction, and four distinct regression algorithms, we demonstrated that an XGBoost model can accurately predict granular, normalized priority scores (0.00–1.00). Our experimental evaluation, conducted on the publicly available UCL Smart Card dataset, yielded several key findings:

1. Superior Regression Accuracy: The XGBoost model achieved an MAE of 0.0678, representing a drastic improvement over the 0.9000 MAE reported by Jamasb et al. (2025).
2. Competitive Classification Accuracy: Our model achieved an F1-Score of 0.4560, which is highly competitive with the Jamasb baseline's 0.6000, especially considering our model operates entirely without manual human intervention.
3. Effective Ordering: With a Spearman Rank Correlation of 0.4156, our model reliably preserves the relative order of importance, making it a practical tool for backlog grooming.

Our work proves that automated supervised ML pipelines are not just a theoretical substitute for unsupervised NLP frameworks; in many ways—specifically regarding cost, scalability, and full automation—they are a superior alternative. They allow agile teams to make data-driven decisions quickly, filtering out human biases and focusing on the actual semantic importance of the requirements without requiring human review.

Future work will focus on three key areas. First, we plan to integrate cost and risk estimation features into the model architecture, allowing for even more accurate predictions. Second, we will explore the application of Large Language Models (LLMs) to replace TF-IDF, potentially capturing deeper semantic contexts and closing the F1-Score gap entirely. Finally, we will implement Explainable AI (SHAP analysis) to generate human-readable justifications for every ranking, increasing stakeholder trust in the automated system.

REFERENCES

1. Jamasb, B., Khayami, S. R., Akbari, R., & Taheri, R. (2025). An Automated Framework for Prioritizing Software Requirements. *Electronics, 14*(3), 512. [MDPI]
2. Talele, P., & Phalnikar, R. (2021). Requirements prioritization using machine learning techniques. *International Journal of Computer Applications, 174*(17), 1–7.
3. Chimugu, P., Selamat, A., & Ibrahim, R. (2019). Clustering-based technique for large-scale prioritization during requirements elicitation. International *Journal of Advanced Computer Science and Applications, 10*(5), 245–252.
4. Saaty, T. L. (1980). The Analytic Hierarchy Process. McGraw-Hill.
5. Iqbal, M., & Sadiq, M. (2020). A systematic literature review on software requirements prioritization techniques. *Journal of King Saud University - Computer and Information Sciences*
6. Fatima, A., Fernandes, A., Egan, D., & Luca, C. (2023). *Software requirements prioritisation using machine learning.* Proceedings of the 18th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2023), 172–179. https://doi.org/10.5220/0011796900003458
7. Alhenawi, Esraa; Awawdeh, Shatha; Abu Khurma, Ruba; García Arenas, Maribel; Castillo, Pedro A.; Hudaib, Amjad. Choosing a Suitable Requirement Prioritization Method: A Survey. *Journal of Computer Science and Technology, 24*, No. 1, Apr. 2024; also arXiv preprint arXiv:2402.13149. DOI:10.24215/16666038.24.e04

Corresponding author: Maryam Nasir Modibbo
maryammodibbonasir@yahoo.com
Department of computer science Umaru Musa Yar'adua university katsina.

8. Ali Fadlallah, K. I., & Yahia Eldow, M. E. (2024). Machine learning: A survey of requirements prioritization: A review study. *Journal of Artificial Intelligence and Computational Technology, 1(*1). https://doi.org/10.70274/jaict.2024.1.1.34
9. Sonawane, S. N., & Puthran, S. M. (2024). Classification of functional and nonfunctional requirements based on convolutional neural network with flower pollination optimizer. Innovations in Systems and Software Engineering. Advance online publication. https://doi.org/10.1007/s11334 024 00592 z
10. Raj, A., & Deora, R. (2025). AI and ML powered feature prioritization in software product development. International Journal of Data Mining & Knowledge Management Process (IJDKP), 15(1).

Corresponding author: Maryam Nasir Modibbo
maryammodibbonasir@yahoo.com
Department of computer science Umaru Musa Yar'adua university katsina.