Context-Aware Requirements Prioritization Using Integrated Regression Learning with Ordinal Neural Modeling and Roberta

Prasis Poudel[1], Noraini Che Pa[2]*, Abdikadir Yusuf Mohamed[3]
Department of Information System and Software-Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, 43300 Malaysia[1, 2]
Faculty of Graduate Studies and Research, Somali National University, Mogadishu, Somalia[3]

***Abstract*—Effective prioritization of software requirements is essential for reducing project risks, optimizing resource allocation, and ensuring timely delivery. Conventional approaches such as Analytic Hierarchy Process (AHP) and MoSCoW often suffer from subjectivity, inefficiency, and poor scalability, making them unsuitable for large-scale projects. Although machine learning (ML) based methods improve scalability, they frequently overlook critical contextual factors such as risk, urgency, implementation effort, and inter-requirement dependencies. To address this gap, this study proposes a new machine learning based context aware software requirements prioritization system. In the proposed system, a pre-trained RoBERTa model and an ordinal neural regression model are employed to infer contextual features including technical risk, complexity, urgency, business value, implementation effort, requirement stability, stakeholder criticality, security sensitivity, and inter-requirement dependencies directly from requirement statements. These inferred features are then used as inputs to a supervised multiple regression model (XGBoost), which generates continuous priority scores for each requirement, with higher scores reflecting higher implementation priority. To ensure transparency, SHAP-based feature attribution is applied for feature importance analysis, and a feedback integration mechanism allows stakeholders to iteratively refine prioritization outcomes, thus in-turn retraining the core prioritization model. Empirical validation against three domain experts across five projects from different application domains demonstrates strong alignment, with Spearman rank correlations between 0.6 and 0.75, Mean Absolute Error (MAE) around 0.10, and Top 5 Match Rates up to 0.80. The results confirm that the proposed system provides a scalable, explainable, and context aware requirements prioritization mechanism suitable for real-world software engineering projects.**

***Keywords*—Requirements prioritization; context-aware prioritization; machine learning; natural language processing; ordinal regression; dependency analysis; Explainable AI**

I. Introduction

Contemporary software projects exhibit increasing complexity as application priorities shift during development and as requirements interact through technical and organizational dependencies [1]. In such environments, the ordering of requirements functions as a determinant of system risk, resource contention, and delivery reliability; inappropriate ordering propagates bottlenecks, extends schedules, and can precipitate project failure. Practice often adopts structured techniques such as Analytic Hierarchy Process, MoSCoW, and case-based ranking to introduce discipline into decision making; however, these techniques depend heavily on judgments elicited from experts at specific points in time and require substantial manual effort, which constrains reproducibility and scalability when the requirement set is large or volatile [1]. Related work on interactive optimization for the Next Release Problem also shows how optimization and learning can be combined for release planning [2]. A further limitation is the absence of a formal representation of project specific context. Urgency, failure risk, technical constraints, and operational dependencies are frequently handled implicitly or retrospectively, producing priority lists that appear consistent during elicitation yet diverge from the evolving state of the project [3]. Structured prioritization has also been reviewed at the use-case level, extending the evidence base beyond requirements lists [4]. Recent systematic literature reviews (SLRs) consolidate more than one hundred and forty requirements prioritization techniques from 2000 to 2021, highlighting that both traditional approaches (AHP, MoSCoW, numerical assignment) and modern AI-based methods (fuzzy logic, genetic algorithms, machine learning) still face persistent challenges in scalability, stakeholder coordination, dependency handling, and automation [3][5][6][32]. Thus, there is an evident need for development of scalable, empirically validated ML models that support can real-world prioritization workflows.

Within such contexts, machine learning models for requirements prioritization offer a scalable and objective alternative by analyzing historical data together with project specific contextual information to automate prioritization [7] [30]. This approach reduces continuous reliance on expert intervention, thereby lowering subjectivity and improving the consistency of decisions. In practice, machine learning methods have been shown to increase efficiency while decreasing manual effort in prioritization tasks [7][8][9]. Recent machine learning surveys on requirements prioritization identify scalable, data-driven approaches including case-based ranking, classification-based methods, clustering algorithms (k-means, Fuzzy C-Means), and semi-automated frameworks such as SRP Tackle, which demonstrate strong performance on large-scale requirement sets while reducing expert elicitation burden [9][10]. Additionally, several works now incorporate semi-automated and interactive mechanisms where stakeholders provide feedback to refine prioritization outcomes, partially

*Corresponding author.

addressing human-in-the-loop and iterative refinement challenges [2][9][10] [11] [31].

Although machine learning approaches improve scalability and objectivity relative to manual methods, a persistent gap is their limited treatment of context [5]. Automated models frequently omit the intent and contextual cues embedded in requirements statements. When project-specific factors such as urgency, risk, complexity, resource needs and availability, and dependencies are not represented, the resulting priority order may diverge from project goals and constraints. For example, selecting a low-risk item with no dependencies ahead of a high risk, mission critical item can increase cost and delay delivery. In such cases, the prioritization output no longer serves the project's objectives and weakens the value of the process. At the same time, several recent strands of research address parts of this gap: 1) Natural language processing (NLP) in requirements engineering uses deep language models and dependency parsing to extract requirement attributes, risk indicators, and structural properties directly from textual requirements, enabling more precise feature engineering [12] [13] [14]; 2) Dependency-aware prioritization techniques such as DRank and CDBR explicitly model and exploit requirement dependency graphs, using graph-based ranking, execute-before-after relationships, and collaborative filtering to propagate priority signals across dependent items [9][11][15][16]. 3) Recent machine learning surveys document growing adoption of supervised classification, clustering, ranking algorithms, and swarm optimization approaches that incorporate dependencies, though often in isolation and not jointly optimized [5][9] and 4) Explainability and interpretability methods are increasingly discussed as essential for understanding and validating AI-based prioritization decisions, though their application to requirements prioritization pipelines remains limited [5][9][17]. However, these approaches typically address specific factors or mechanisms (e.g., dependency graphs alone, or NLP for classification only) and often require substantial elicitation effort, manual configuration, or separate treatment of contextual signals, leaving room for more integrated, end-to-end frameworks that jointly infer multiple contextual features from requirement text, model dependencies explicitly, and provide explainability through a unified prioritization pipeline.

This study addresses these limitations through the following objectives:

- Design and implement a context aware machine learning model that prioritizes software requirements from their descriptions and contextual features, including risk, urgency, and dependencies. In contrast to prior dependency-aware approaches (such as DRank and CDBR) that focus primarily on explicit dependency matrices or pairwise comparisons, and to NLP studies that target isolated feature extraction or classification [9][11][12][15], the proposed model infers a broader integrated set of contextual variables including technical risk, complexity, urgency, implementation effort, stability, stakeholder criticality, security sensitivity, and inter-requirement dependencies directly from requirement text and combines them within a unified regression framework.
- Apply natural language processing techniques, using RoBERTa for risk estimation and dependency analysis, and an ordinal neural regression model for other contextual features such as complexity, urgency, implementation effort, stability, stakeholder criticality, security sensitivity, and business value [12][18]. This design builds on recent advances in using deep NLP models and interpretable machine learning for requirements classification, attribute extraction, and dependency identification [12][13][14][19][20], while extending them beyond isolated classification tasks to multi-feature context inference optimized specifically for downstream prioritization.
- Use a supervised multiple regression model (XGBoost) to generate continuous priority scores. Regression-based integration of contextual features aligns with recent automated prioritization frameworks and machine learning surveys that recommend ensemble methods for requirements ranking [9][21]; however, in this work the regressor operates over features that are explicitly inferred from requirement text via NLP and evaluated for importance through explainability analysis, rather than relying solely on manually annotated attributes or any ad-hoc feature engineering process.
- Establish a mechanism to collect and integrate stakeholder feedback, including revised priority scores and contextual feature values, to update the training data and improve model adaptability and accuracy over time. This feedback-driven retraining mechanism is consistent with semi-automated and human-in-the-loop prioritization techniques reported in recent literature [9][10] and aligns with the push toward interactive ML in requirements engineering; however, this work uniquely applies the feedback loop to an explicitly context-aware, explainability-informed feature space shaped by SHAP-based feature selection during design time, enabling both transparency and iterative refinement.

This study contributes the following to requirements engineering:

- Context aware framework: A scalable and automated requirements prioritization system that integrates machine learning and natural language processing to infer contextual features from requirement statements, providing a foundation for informed and efficient prioritization [21]. Semi-automated techniques reported in prior work also motivate the integration of human feedback into prioritization pipelines [10]. By jointly modeling textual features, inferred contextual signals (risk, complexity, effort, stability, security, stakeholder importance), inferred dependencies, and learned dependencies within a single end-to-end pipeline, the framework addresses research gaps identified in recent SLRs and complements existing context-aware, dependency-aware, and ML-based RP approaches, thereby clarifying the boundary between earlier techniques and this present work [5][6][9][11][15].

- Contextual feature inference: Use of RoBERTa for risk estimation and dependency analysis, together with an ordinal neural regression model for additional contextual features, and integration of these values into a unified prioritization pipeline [19]. Unlike prior NLP-based efforts in requirements engineering that focus mainly on requirement classification, non-functional requirement type detection, or structural complexity assessment [12][13][14], the feature inference layer here is explicitly optimized via supervised learning to feed a downstream prioritization regressor, enabling true end-to-end learning from requirement text to priority scores without intermediate manual steps.
- Regression-based integration: Combination of inferred and annotated contextual features within a supervised multiple regression model (XGBoost) to produce continuous priority scores and improve predictive performance [20] . This extends earlier regression and optimization-based prioritization models by incorporating explainability-informed feature selection (via SHAP) and text-derived dependency signals, which recent surveys indicate have not yet been jointly exploited in existing AI-based and machine learning-based RP studies [5][9][15][21].
- Design time SHAP feature selection: Application of Shapley Additive Explanations once during model design to select influential features for the final training dataset, ensuring consistent training and retraining while excluding SHAP from the runtime pipeline [17]. SHAP and related explainability methods are increasingly used to analyze and interpret complex ML models in various domains [17], and their application to model development (feature selection, model debugging) is well established; however, their use to deliberately shape the feature space of a requirements prioritization pipeline selecting only the most impactful features informed by explanations remains underexplored in the requirements engineering literature, particularly in conjunction with NLP-based contextual feature inference.
- Feedback driven retraining: A mechanism for stakeholders to review and adjust prioritization outputs; the collected feedback updates the training data using the same SHAP selected feature set to improve adaptability and maintain accuracy over time [12]. This human-in-the-loop mechanism complements existing interactive and semi-automated approaches such as SRP Tackle (which uses semi-automation to reduce expert burden), DRank (which resolves conflicts via dependency propagation), CDBR (which integrates stakeholder and developer preferences with dependency-based optimization), and interactive genetic algorithms discussed in recent surveys [9][10][11][15], by embedding explicit human feedback directly into the retraining cycle of a context-aware, explainability-informed model and enabling stakeholders to influence the feature importance profile over time.
- Empirical validation: The system is evaluated with multiple domain experts across diverse projects, showing reduced manual workload and improved alignment of priorities with risk mitigation and resource allocation objectives [16]. The evaluation design aligns with recommendations from recent AI-based and machine learning SLRs on requirements prioritization, which emphasize the importance of empirical assessment across diverse projects, evaluation of scalability with large requirement sets, validation of dependency handling, and measurement of stakeholder satisfaction and usability in realistic industrial settings [5][6][9].

The remainder of the paper is organized as follows. Section II presents the methodology: requirement data preparation, contextual feature inference with RoBERTa and ordinal neural regression, SHAP feature selection, and development of the core regression prioritization model. Section III describes the system design and implementation, including the architecture, integration of contextual feature estimators, dependency analysis, backend services, and the user interface. Section IV reports empirical validation with expert evaluation and performance analysis using mean absolute error, Spearman rank correlation, and Top 5 match rate. Discussion is given in Section V. Section VI concludes with a summary of contributions and directions for future work aimed at improving scalability, adaptability, and applicability in practice.

II. Methodology

This study employs a structured methodology with five sequential phases to develop and evaluate a context aware, machine learning based requirements prioritization system. The complete workflow is shown in Fig. 1.

A. Phase 1: Requirement Data Preparation

Requirement statements were collected from publicly available Software Requirements Specification (SRS) documents of multiple open-source projects. The documents were processed to construct a dataset for training and testing the prioritization models. Preprocessing included tokenization, vectorization, lemmatization, removal of special characters, and lowercasing. These steps standardize the text and prepare it for use in the natural language processing and machine learning models employed in subsequent phases of the study [22].

B. Phase 2: Contextual Feature Estimation

Each contextual feature was modeled with a dedicated machine learning pipeline appropriate to its data type and structure. The training data consisted of requirement statements from open-source SRS documents, which were annotated for contextual feature values and binary dependency [16]. The accompanying table summarizes the specification and characteristics of the estimation and inference models. Coupling preferences with dependency information has been shown to improve prioritization quality in semi-automated and dependency aware settings [11] [16] [23]. SHAP (Shapley Additive Explanations) was then applied to a subset of the labeled data to identify the features with the greatest influence on the output of the core prioritization model; a basic random forest regressor was used only for this SHAP analysis step [24]. The most influential features were selected and fixed for training the prioritization model. Finally, all trained inference models, together with configuration files, tokenizers, and vectorizers,

were saved for use by the backend API of the core prioritization system.

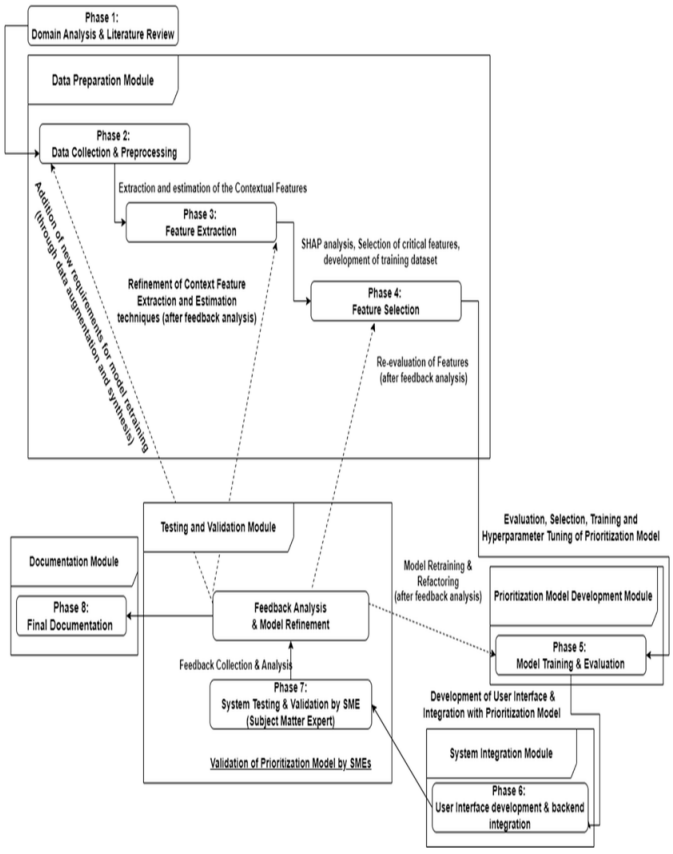

Fig. 1. Methodology workflow diagram.

C. Phase 3: Requirements Prioritization Model

The requirement prioritization pipeline begins by estimating contextual features from requirement statements using the inference models developed in Phase 2. Next, the expert labeled training data are augmented to create the initial training set for the core prioritization model. Augmentation injects small Gaussian noise into sentence embeddings derived from the requirement text, as well as into priority scores and contextual feature values scaled from 0 to 1, while preserving the original dependency labels to maintain structural consistency. A subset of the augmented data is then used to train multiple supervised regression models, including Random Forest, Gradient Boosting Regressor, Support Vector Regressor, and XGBoost Regressor. Model performance is evaluated with fivefold cross validation using Root Mean Squared Error, Mean Bias Error, Mean Absolute Error, Mean Bias Ratio, and the coefficient of determination R2 as metrics. Across these comparisons, the XGBoost Regressor shows the strongest accuracy and generalization [25].

After selecting XGBoost as the core prioritization model, hyperparameter tuning was conducted using two strategies: randomized search cross validation and Bayesian optimization [26]. The search covered learning rate, maximum tree depth, number of estimators, L1 and L2 regularization strengths, and subsampling ratio [29]. In comparison, Bayesian optimization produced lower root mean squared error and higher overall performance while exploring the search space more efficiently [26]. The final XGBoost model, trained on the selected feature set with the best parameters from Bayesian optimization, was exported together with the TF IDF vectorizer. These components were integrated into the backend application programming interface of the prioritization system, which prioritizes new software requirements. The main stakeholder interactions are illustrated within the use case diagram shown within Fig. 2.

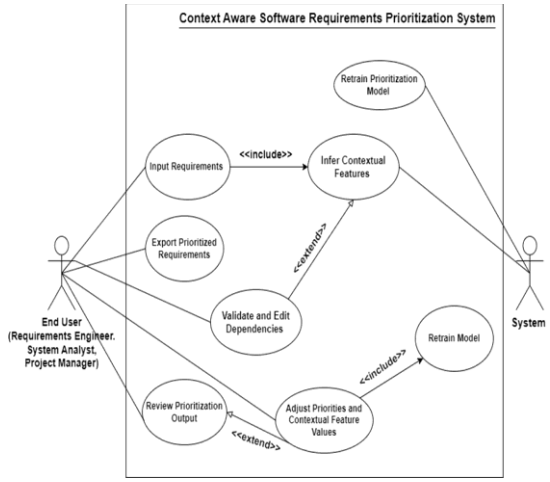

Fig. 2. Use case diagram (core priorization).

D. Phase 4: Model Validation and Interpretability

After tuning and finalizing the core prioritization model, an empirical study is conducted to assess its applicability in real settings. The model's predicted priority scores and ordering are compared with the judgments of three domain experts on a new set of requirement statements and project contexts. Performance is measured using Spearman rank correlation, Top 5 match rate, and mean absolute error. Two evaluation modes are used. In the end-to-end automated mode, the inference models estimate contextual feature values and binary dependencies. In the manual mode, domain experts provide contextual feature and dependency values after reviewing the project contexts. The results provide quantitative evidence of the applicability and effectiveness of the machine learning based prioritization system and address the study objective of reducing subjectivity and improving scalability in requirements prioritization [5][9].

E. Phase 5: System Deployment

The system is deployed with Fast API to provide backend services and programmatic access to the core prioritization models through HTTP endpoints. A user interface enables nontechnical users to submit requirement statements and view the resulting priority scores and ordering.

III. Implementation

A. Data Acquisition and Preparation

The first step is the search and collection of software requirement statements from Software Requirements Specification (SRS) documents associated with open source or declassified projects. To support contextual diversity and generalizability of the machine learning models, SRS documents were drawn from a range of project domains,

including telecommunication, healthcare, finance, and education.

After extracting the requirement statements, a domain expert manually annotated contextual feature values for each statement in the context of its source project. The features included technical implementation risk, complexity, urgency, stability across the project lifecycle, implementation effort, security sensitivity, and stakeholder criticality. Prior work has used NLP to assess structural properties and complexity of requirements, supporting the inclusion of such signals in our feature set [14]. Each feature was rated on a five-point ordinal scale from 1 (lowest) to 5 (highest) with respect to the corresponding SRS. In addition, inter requirement dependencies were annotated within each project using a binary label (dependent or independent) based on structural relationships observed in the SRS. Graph-based representations capture inter-sentence dependencies relevant to inferring such links [27]. These annotations constitute the ground truth for training the contextual feature inference models and the core requirements prioritization model used in this study.

Before training the machine learning models, the requirement text was preprocessed to standardize format by lowercasing, removing special characters, and normalizing whitespace. The semantic content of the statements was preserved without paraphrasing to remain consistent with the expert annotations. Contextual feature values were normalized to the interval [0,1] (for example, 2 maps to 0.4). Requirement statements were then vectorized with the all-mpnet-base-v2 Sentence Transformer, yielding 768 dimensional embeddings that capture the meaning and nuances of each statement.

To address the sparsity of expert labeled training data and to improve the robustness of the contextual feature estimators, the dependency detector, and the core prioritization model, a controlled data augmentation procedure is applied. The procedure injects Gaussian noise into the sentence embeddings and the normalized contextual feature values while leaving the dependency labels unchanged to preserve logical correctness. This emulates minor contextual and semantic variations that arise in practice [28]. Gaussian noise is applied to each input element, including normalized feature values and the 768-dimensional embedding that represents each requirement statement, using the formula given below.

$$x^j = x + \varepsilon,\ \varepsilon \sim N(\mu, \sigma^2) \quad (1)$$

x: Original data point (scalar or vector).

x^j: jth augmented sample generated from x.

ε: Additive Gaussian noise.

μ: Mean of the Gaussian noise (often 0).

σ: Standard deviation of the Gaussian noise; the variance is σ^2.

Here, gaussian noise is sampled from the probability density function given below:

The Gaussian probability density function specifies the distribution of the noise introduced during data augmentation and enables controlled simulation of stochastic fluctuations around a chosen mean. This supports more robust estimation of contextual features and dependencies and improves the generalizability of the core prioritization model. The augmented samples were then combined with the original expert labeled data to form a more comprehensive and balanced training set. Finally, the target variable Priority Score was perturbed with Gaussian noise using a separate standard deviation σ priority and then clipped to the interval [0.5,1.0] to keep augmented labels within realistic scoring ranges.

B. SHAP Analysis for Contextual Feature Evaluation

To examine the effect of contextual features on the output of the core prioritization model, Shapley Additive Explanations (SHAP) are used to decompose each prediction into additive contributions from individual features. SHAP supports global and local interpretation, which allows quantification of feature importance, assessment of average directional effects, and inspection of instance level behavior for each contextual feature [24]. The following subsections present visualizations of the SHAP analysis performed on the initial augmented training data with a basic random forest regressor.

Fig. 2 ranks the features by their mean absolute SHAP value, which reflects overall influence on the model output irrespective of sign. For each feature, the mean of the absolute Shapley values is computed across all test instances, providing an aggregate measure of the extent to which the feature moves predictions away from the base value. Requirements stability shows the highest mean absolute SHAP value (0.035), indicating the strongest contribution to the predicted priority scores. Complexity (0.028) and urgency (0.020) follow, suggesting that anticipated implementation difficulty and time sensitivity also have substantial impact. Implementation effort (0.018) and stakeholder criticality (0.016) form a middle group with moderate effects. Business value, security sensitivity, and technical risk have lower values (0.010 to 0.012), indicating less consistent contributions across the dataset.

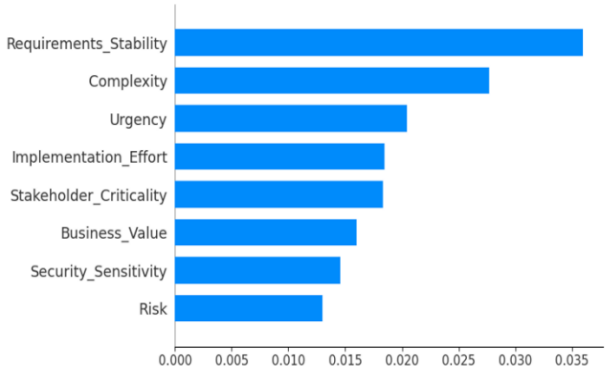

Fig. 3. Outcome of SHAP analysis.

This observation raises the possibility of under representation or collinearity with other features. Whereas the previous figure reports the absolute magnitude of feature importance, the current Fig. 3 presents signed mean SHAP values, which indicate whether a feature increases or decreases the predicted priority on average. The signed mean is obtained by averaging the raw Shapley values, including their signs, across all samples in the training data. In this analysis,

requirements stability shows a small negative mean (approximately −0.0010), indicating that higher stability is associated with lower predicted priority. These average directional effects of each contextual feature on the predicted priority scores are illustrated in Fig. 4.

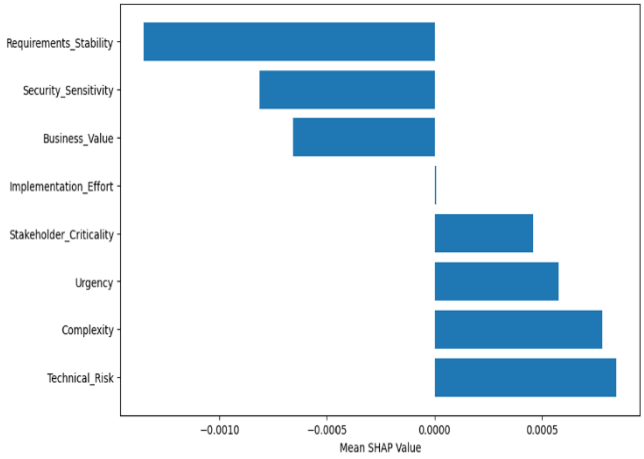

Fig. 4. Average Directional Effects (SHAP analysis).

This pattern is consistent with the local rationale that stable requirements are less urgent and can be deferred without jeopardizing schedules or increasing the risk of implementation failure. Security sensitivity (approximately −0.0007) and business value (approximately −0.0005) also show negative means, suggesting that highly valuable or sensitive items may be deprioritized relative to volatile tasks, possibly reflecting a trade off with complexity or risk. In contrast, technical risk exhibits a positive mean effect (approximately +0.0005), indicating that riskier requirements tend to receive higher priority, which aligns with a mitigation first strategy that addresses components prone to failure early in the lifecycle. Urgency, stakeholder criticality, and complexity show positive average contributions (approximately +0.0004 to +0.0008), reinforcing their influence in increasing the priority score. Taken together, the directional analysis indicates an implicit policy to defer stable, even high value, requirements while advancing urgent, complex, or risky items. Based on the SHAP analysis, requirements stability, complexity, and urgency are retained for their strong mean absolute contributions, and technical risk, implementation effort, and stakeholder criticality are included for their consistent directional effects, which capture risk mitigation, resource demand and allocation, and stakeholder influence in prioritization. This feature subset is expected to preserve predictive accuracy, improve interpretability, and reduce the risk of overfitting.

C. Model Evaluation and Metrics

The dataset refined through augmentation was used to train four machine learning models: Random Forest, Gradient Boosting, XGBoost, and a multilayer perceptron. Model performance was assessed with K fold cross validation using root mean square error (RMSE), mean bias error (MBE), mean absolute error (MAE), mean bias ratio (MBR), and the coefficient of determination R^2. Brief descriptions of these metrics are provided in Table I. The corresponding formulas are presented below.

TABLE I. MODEL EVALUATION METRICS

Model Evaluation Metrics	
Evaluation Metrics	***Description***
Root Mean Square Error	The Root Mean Square Error metric measures the average magnitude of error (net difference) between the actual values (within the training subset of data) and predicted values (actual values predicted by the regression learning models). This approach typically penalizes larger errors (difference in prediction) more than smaller errors due to the squaring operation before averaging. So, lower root means square error signifies better model accuracy as it quantifies comparatively lower prediction error.
Mean Bias Error	The Mean Bias Error actually measures the average difference between the actual and predicted values, which takes a bit different approach than RMSE where more focus is given only on measuring magnitude 78 of prediction error, whereas MBE also takes signs of the absolute difference into consideration. Considering the sign of the difference in predicted and actual values, MBE signifies positive and negative biases in prediction. This approach aids in identifying systemic bias, where positive MBE indicates model is under predicting (model output lower than actual values) and negative MBE indicates the model is over predicting (model output higher than actual values) than the actual values
Mean Absolute Error	The mean absolute error (MAE) is a popular metric for evaluation of regression models, which measures the average magnitude of the prediction error without considering the direction. This approach of evaluating only the net difference in predictive error, makes MAE a more direct and explainable error metric, which neither allows errors to cancel out like MBE.
Mean Bias Ratio	This evaluation metric like the MBE is also focused on evaluation of the bias (over and under prediction) within a model's prediction. However, MBR focuses on determining whether a model systematically over predicts or under predicts outcomes. Similarly, a positive mean bias ratio signifies that the model underestimates values and a negative mean bias ratio signifies that the model overestimates values. So, mean bias ratio closer to zero signifies overall lower or minimal bias in predictions made by model.
R2 (R-squared coefficient of determination)	The R2 is a metric that quantifies how well the regression learning model explains the variability of the actual data (.i.e. How well does the regression model's prediction match the actual data). If the model predicts perfectly, the numerator (sum of squared errors) becomes zero, then . But if the model can't explain variance well, the numerator will be larger making it closer to zero or even negative in some cases

R-squared coefficient of determination

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \quad (2)$$

Mean Bias Ratio

$$MBR = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)}{\sum_{i=1}^{n} y_i} \quad (3)$$

Mean Absolute Error

$$MAE = \left(\frac{1}{n}\right)\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (4)$$

The evaluation results of the model after K - cross fold validation using the above-mentioned metrics are illustrated within the graphs provided within Fig. 4, 5 and 6 where Fig. 4, Fig. 5 and Fig. 6 refer to average directional effects (SHAP Analysis), Test RMSE, and Test MBE respectively.

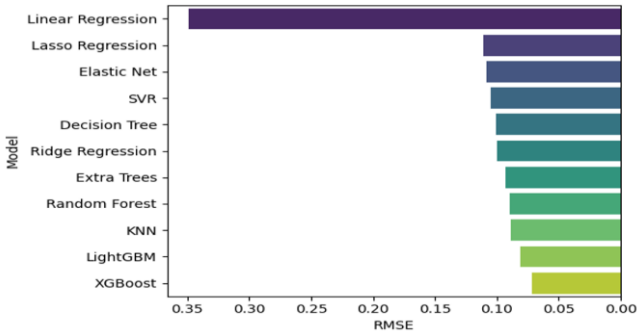

Fig. 5. Test RMSE.

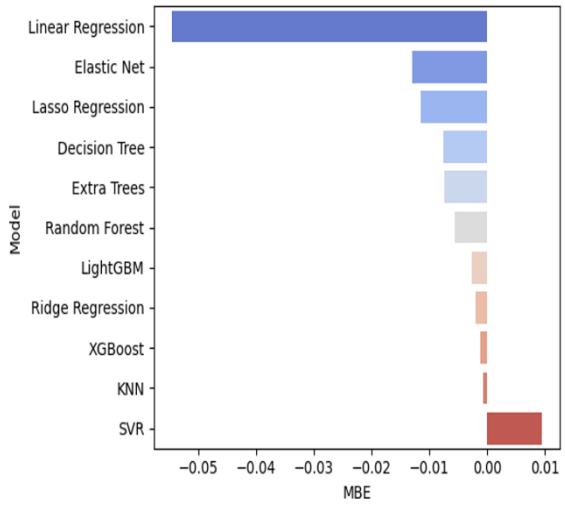

Fig. 6. Test MBE.

Mean Bias Error

$$MBE = \left(\frac{1}{n}\right)\sum_{i=1}^{n}(\hat{y}_i - y_i) \quad (5)$$

Root Mean Square Error

$$RMSE = \sqrt{\left(\frac{1}{n}\right)\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (6)$$

The results show that XGBoost outperformed the other models. It achieved the lowest root mean square error (0.0718), the highest coefficient of determination R^2 (0.7991), and near zero bias (MBE = −0.0011, MBR = −0.0013). Light GBM and k nearest neighbors were the next best models but with slightly lower accuracy and consistency. Linear regression performed poorly, with higher error and a negative R^2, indicating that it did not capture the structure of the data. Plots comparing the models across all metrics corroborate these findings. Test RMSE, cross validation RMSE, R^2, MBE, and MBR were all lowest or closest to ideal for XGBoost, supporting its better generalization, stability, and low bias. The cross-validation RMSE values for the candidate models are reported in Fig. 7 and distribution of test mean bias ratio in Fig. 8.

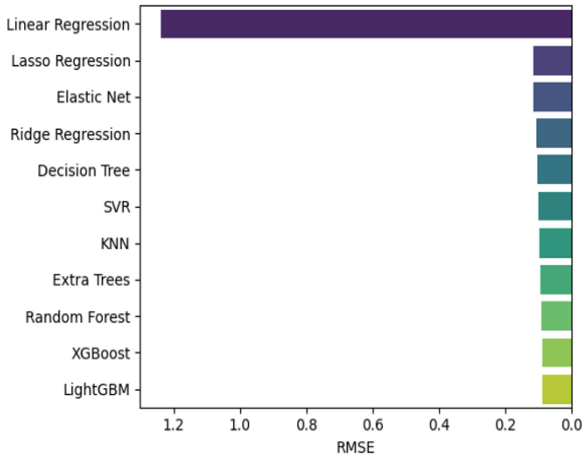

Fig. 7. Cross-Validation RMSE (Train).

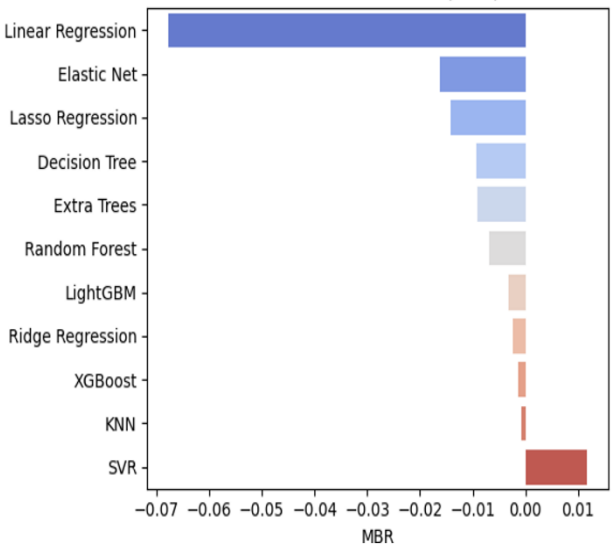

Fig. 8. Test Mean Bias Ratio (MBR).

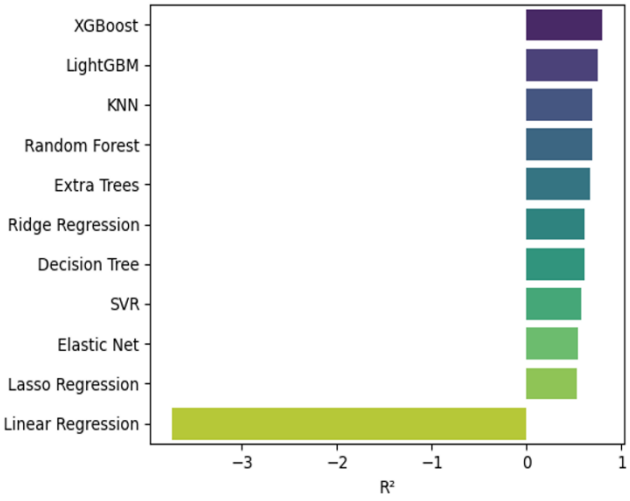

Fig. 9. Coefficient of determination (R2).

Furthermore, the corresponding coefficients of determination for the evaluated models are presented in Fig. 9.

D. Hyperparameter Tuning Process

After selecting XGBoost as the best performing model, we refined it through hyperparameter tuning using two techniques: randomized search cross-validation and Bayesian optimization. Both procedures were implemented and evaluated using the study's core metrics, and the tuned configurations were compared on the same validation protocol. The results of the two hyperparameter tuning approaches for XGBoost are as follows:

TABLE II. HYPERPARAMETER PROCESS

Outcome of Hyperparameter Tunning			
Method	***Best RMSE***	***Best R2***	***Best MBE***
Bayesian Optimization	0.074881	0.781547	0.000841
Randomized Search	0.078633	0.759108	–0.002580

The model tuned with Bayesian optimization produced a lower root mean square error (0.07488 versus 0.07863), a higher coefficient of determination R^2 (0.78155 versus 0.75911), and a mean bias error closer to zero and positive (0.00084 versus −0.00258), indicating lower and more balanced bias. The tuning results are summarized in Table II. These results support Bayesian optimization as the preferred hyperparameter tuning strategy for the XGBoost model used for contextual requirements prioritization. The tuned model was then retrained on the full training set, excluding the portion reserved for evaluation, and both the final model and the TF IDF vectorizer were saved for deployment within the prioritization system.

E. Implementation and Validation of Core Prioritization Model

The system architecture consists of a ReactJS frontend and a Fast API backend. Fast API is used to implement RESTful endpoints with asynchronous request handling and data validation. The endpoints support requirement submission, prioritization, editing of contextual features, dependency analysis, and retrieval of prioritization results. The frontend provides modular components for user interaction and communicates with the backend through HTTP POST and GET requests. The High-Level overview of the prioritization model's architecture is shown in Fig. 10 and details of API endpoints are summarized in Table III.

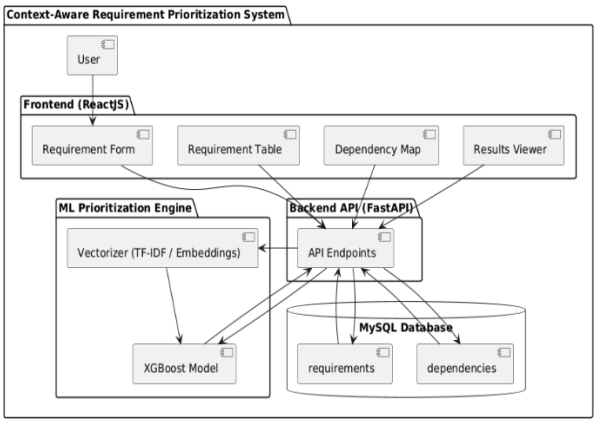

Fig. 10. Prioritization model architecture.

TABLE III. API ENDPOINTS DESCRIPTION

API endpoints description				
Endpoint	***Purpose***	***Method***	***Input***	***Output***
/estimate_risk	Estimate technical risk	Post	Requirement text	Risk level (1–5)
/estimate_dependency	Estimate binary	Post	Requirement pair	0 or 1
/estimate_complexity	dependency	Post	Requirement text	Score (1–5)
/estimate_urgency	Estimate implementation complexity	Post	Requirement text	Score (1–5)
/estimate_stability	Estimate urgency	Post	Requirement text	Score (1–5)
/estimate_effort	Estimate requirement stability	Post	Requirement text	Score (1–5)
/estimate_security	Estimate implementation effort	Post	Requirement text	Score (1–5)
/estimate_criticality	Estimate security sensitivity	Post	Requirement text	Score (1–5)
/prioritize	Predict final requirement priority	Post	Requirement text embeddings, Contextual features and dependency values within JSON format.	Priority score (0–1)

IV. RESULTS

This section reports the results of expert-based validation of the core prioritization model, an XGBoost model optimized with Bayesian optimization. Performance is summarized for five projects from different domains and compared with the judgments of three domain experts. The analysis focuses on alignment between the model and the experts using three metrics: Spearman rank correlation, Top 5 match rate, and mean absolute error.

Fig. 11 summarizes mean absolute error results across projects and experts. Excluding three specified instances, namely Expert 1 in Project 5 with MAE 0.07, Expert 2 in Project 4 with MAE 0.06, and Expert 3 in Project 3 with MAE 0.07, the model maintains low error across the remaining evaluations, with MAE values between 0.06 and 0.11 and an average near 0.10. The agreement is strongest in logistics, where Project 2 shows MAE 0.06 for all experts, and in telecommunications, where Project 1 reports MAE 0.08 for Expert 1, 0.07 for Expert 2, and 0.11 for Expert 3. Minor variations, such as the higher MAE of 0.11 for Expert 3 in Project 1, indicate deviations in some project contexts.

Fig. 12 reports Spearman rank correlation outcomes. Across projects, the model shows consistent ranking agreement, with average Spearman rank correlation between 0.6 and 0.7. The strongest agreement is observed with Expert 2, particularly in

healthcare (Project 5) and logistics (Project 2), where Spearman rank correlation reaches 0.75 and 0.70 respectively.

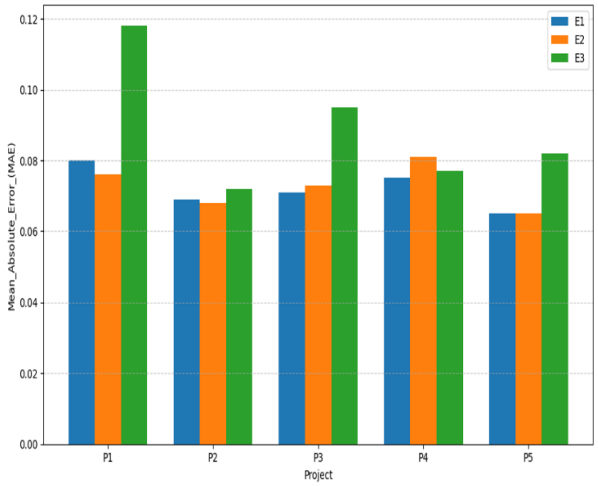

Fig. 11. Mean Absolute Error (MAE) values.

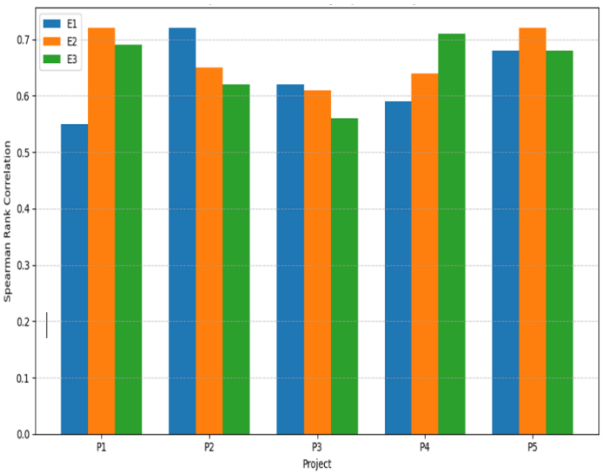

Fig. 12. Spearman rank correlation outcomes.

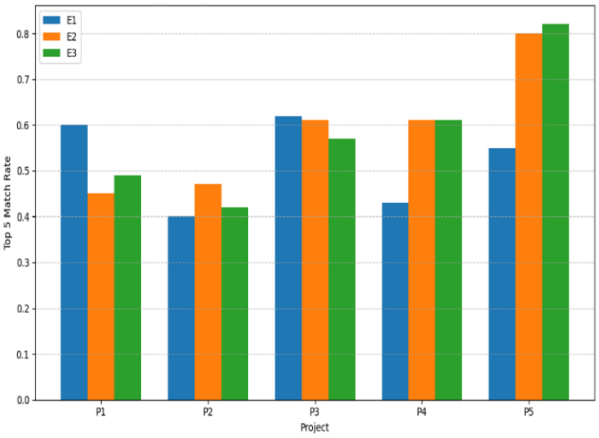

Fig. 13. Top 5 match rate values.

Fig. 13 presents Top 5 match rate values, with average Top 5 match rate between 0.5 and 0.55. In Project 5, the Top 5 match rate attains 0.80 for Expert 2, indicating close agreement on the highest-priority items. In Project 3, Spearman rank correlation is 0.60 for Experts 1 and 2 and the Top 5 match rate is 0.60 for both.

Additional results include Project 4 with Spearman rank correlation 0.60 for Expert 1 and 0.40 for Expert 3, and Top 5 match rates of 0.40 and 0.60 respectively. In Project 4, mean absolute error is 0.07 for Expert 1 and 0.09 for Expert 3. In Project 1, MAE is 0.08 for Expert 1, 0.07 for Expert 2, and 0.11 for Expert 3. In Project 5, MAE is 0.08 for Experts 2 and 3. The absence of measurements for Expert 3 in Project 3 limits a complete assessment for that project.

V. Discussion

The results indicate that the model can align closely with expert judgments in domains where requirements are well defined and evaluation criteria are relatively consistent. The strongest agreement with Expert 2 in healthcare and logistics suggests that the inferred contextual representation supports reliable prioritization in such settings, including for critical non-functional requirements.

Performance varies across domains, particularly in e-commerce (Project 4) and smart city (Project 3), where agreement is weaker for some experts. This variability is consistent with domains that exhibit stronger context dependence and greater divergence in expert preferences, which can reduce rank agreement even when absolute score differences remain small. The missing Expert 3 measurements in Project 3 further limits the interpretation of results for that domain.

This study has several limitations that should be considered when interpreting the findings. First, the datasets were derived primarily from open-source Software Requirements Specification documents, which may not fully represent industrial requirements practices, particularly in regulated or safety critical environments. Second, contextual feature values and dependency labels were annotated by a limited number of domain experts, and expert judgments can vary across projects and domains. This variability may influence both the learned mapping between contextual features and priority scores and the observed agreement metrics. Third, the approach infers contextual features from requirement statements, and the inference quality depends on the clarity and completeness of the textual descriptions. Requirements that are underspecified, ambiguous, or strongly dependent on external project context can reduce inference accuracy and, in turn, affect prioritization performance. Fourth, although cross validation and held out expert evaluation were used, the training data remains limited in scale. Data augmentation improves robustness, but it cannot substitute for broader real-world coverage, and generalization to unseen domains may require additional calibration. Finally, the system evaluates agreement against expert prioritizations rather than measuring downstream project outcomes. Therefore, the reported improvements reflect alignment with expert judgment and do not directly quantify impacts on delivery time, defect reduction, or operational risk.

Overall, the findings support robustness in critical domains while indicating the need for targeted improvements to address inconsistencies in subjective or complex project contexts. In practice, the model can reduce manual effort and provide decision support in resource constrained settings, but it should

be used to assist expert decision making rather than replace expert judgment, particularly in domains with high subjectivity.

A. Comparison with Existing Approaches

Existing machine learning based requirements prioritization techniques report improved scalability and reduced elicitation effort, but often without rich contextual features or explicit dependency handling [5] [9]. Semi-automated models such as SRP Tackle achieve high prioritization accuracy on large requirement sets but focus mainly on multi criteria scoring and clustering, with limited attention to fine grained dependency and risk signals [10]. Dependency aware approaches like DRank and CDBR show that incorporating requirement graphs and execution order improves prioritization quality, yet they typically rely on manually specified dependencies and do not infer contextual attributes directly from requirement text [11][16][23].

Within this landscape, the proposed model attains Spearman rank correlations between 0.6 and 0.75, Top 5 match rates up to 0.80, and mean absolute errors around 0.10 across diverse domains, which is consistent with or better than performance ranges reported for recent ML based RP methods when evaluated against expert judgments [7] [9]. The main added value lies in jointly modeling contextual factors such as stability, urgency, risk, and stakeholder criticality, inferring dependencies from SRS text, and providing SHAP based explanations, which extend prior dependency awareness and semi-automated frameworks while maintaining competitive predictive accuracy [5] [16] [24].

B. Limitations and Threats to Validity

The empirical evaluation is based on requirement statements drawn from publicly available or declassified SRS documents across a limited set of domains, including telecommunications, healthcare, finance, and education. This dataset may not fully represent industrial projects with different documentation styles, regulatory constraints, or safety critical requirements, which restricts the generalizability of the findings to other contexts. In addition, the number of projects and requirements per project is modest compared with large scale industrial portfolios, so scalability to very large backlogs remain to be confirmed empirically.

The expert-based validation involves three domain experts whose judgments reflect individual preferences, experience, and familiarity with specific domains. While multiple experts and projects were used to mitigate individual bias, differences in their rankings, particularly in e-commerce and smart city domains, indicate that part of the observed disagreement arises from variability in human judgments rather than model error alone. The study also focuses on Spearman rank correlation, mean absolute error, and Top 5 match rate as evaluation metrics, which capture ordering and score alignment but do not fully reflect all project level outcomes such as business value realization, downstream defect reduction, or stakeholder satisfaction.

From a modeling perspective, there is a risk of overfitting or tuning bias because the XGBoost model and its hyperparameters were selected based on cross validation performance on an augmented dataset. Although K fold cross validation and data augmentation were used to improve robustness, the same family of metrics guides both model selection and evaluation, which may overestimate performance relative to unseen industrial data. Furthermore, the contextual feature estimators and dependency detectors are trained on annotations produced within this study, so systematic biases or inconsistencies in the labeling process could propagate into the final prioritization outputs.

VI. Conclusion and Future Work

This research developed a context-aware requirements prioritization framework using machine learning, centered on an XGBoost regression model optimized with Bayesian optimization. The system automates and improves prioritization by integrating contextual factors such as technical risk, urgency, complexity, requirement stability, implementation effort, security sensitivity, stakeholder criticality, business value, and inter-requirement dependencies, which are inferred from requirement statements in SRS documents or provided through user input. In doing so, the framework addresses limitations of traditional approaches such as Analytic Hierarchy Process and MoSCoW that are sensitive to subjectivity, scale poorly, and depend heavily on expert availability.

The methodology combines fine-tuned RoBERTa models for risk and dependency estimation with sentence transformer embeddings and Coral Ordinal models for complexity and related features. At design time, a feature selection step is performed using the Shapley Additive Explanations (SHAP) process to retain the most influential contextual variables for consistent training and retraining, thereby enhancing interpretability and transparency. Empirical evaluation across five projects with three domain experts demonstrates consistent alignment between model outputs and expert prioritizations, as reflected by Spearman rank correlation, Top 5 match rate, and low mean absolute error across projects.

In practice, the system supports prioritization in dynamic development settings by enabling iterative refinement through stakeholder feedback and by providing structured prioritization outputs that can assist decision-making. The approach is intended to support expert judgment by producing an initial ordering and exposing the contextual basis of priority scores rather than replacing expert decision-making.

Despite the contributions of this study, the evaluation remains limited to open-source and declassified SRS documents, a modest number of projects, and expert validation in selected domains. Future work will expand the annotated training data with larger industrial case studies to increase stakeholder and domain diversity, refine contextual inference using more advanced natural language processing techniques, and strengthen dependency modeling for complex requirement networks. Additional evaluation criteria will be incorporated to improve external validity and assess downstream project outcomes such as delivery efficiency and risk reduction. Continued attention to interpretability, transparency, and ethical scaling is essential for deployment in sensitive domains such as healthcare and other mission-critical systems. Overall, this research contributes to machine learning-assisted software engineering by providing a practical and scalable foundation for context-aware requirements prioritization and management.

ACKNOWLEDGMENT

The authors acknowledge the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM), for supporting this work.

REFERENCES

[1] M. Sufian, Z. Khan, S. Rehman, and W. Haider Butt, "A Systematic Literature Review: Software Requirements Prioritization Techniques," in 2018 International Conference on Frontiers of Information Technology (FIT), Dec. 2018, pp. 35–40. doi: 10.1109/FIT.2018.00014.

[2] A. A. Araújo, M. Paixao, I. Yeltsin, A. Dantas, and J. Souza, "An Architecture based on interactive optimization and machine learning applied to the next release problem," Autom. Software. Eng., vol. 24, no. 3, pp. 623–671, Sep. 2017, doi: 10.1007/s10515-016-0200-3.

[3] F. A. Bukhsh, Z. A. Bukhsh, and M. Daneva, "A systematic literature review on requirement prioritization techniques and their empirical evaluation," Comput. Stand. Interfaces, vol. 69, p. 103389, Mar. 2020, doi: 10.1016/j.csi.2019.103389.

[4] Y. Odeh and N. Al-Saiyd, "Prioritizing Use Cases: A Systematic Literature Review," Computers, vol. 12, no. 7, p. 136, Jul. 2023, doi: 10.3390/computers12070136.

[5] R. Anwar and M. B. Bashir, "A Systematic Literature Review of AI-Based Software Requirements Prioritization Techniques," IEEE Access, vol. 11, pp. 143815–143860, 2023, doi: 10.1109/ACCESS.2023.3343252.

[6] A. M. Radwan, M. A. Abdel-Fattah, and W. Mohamed, "AI-Driven Prioritization Techniques of Requirements in Agile Methodologies: A Systematic Literature Review," Int. J. Adv. Comput. Sci. Appl., vol. 15, no. 9, 2024, doi: 10.14569/IJACSA.2024.0150983.

[7] A. Fatima, A. Fernandes, D. Egan, and C. Luca, "Software Requirements Prioritisation Using Machine Learning:," in Proceedings of the 15th International Conference on Agents and Artificial Intelligence, Lisbon, Portugal: SCITEPRESS - Science and Technology Publications, 2023, pp. 893–900. doi: 10.5220/0011796900003393.

[8] N. R. Bollumpally, A. C. Evans, S. W. Gleave, A. R. Gromadzki, and G. Learmonth, "A Machine Learning Approach to Workflow Prioritization," in 2019 Systems and Information Engineering Design Symposium (SIEDS), Charlottesville, VA, USA: IEEE, Apr. 2019, pp. 1–5. doi: 10.1109/SIEDS.2019.8735589.

[9] K. I. A. Fadlallah and M. E. Y. Eldow, "Machine Learning: A survey of requirements prioritization: A Review Study," J. Artif. Intell. Comput. Technol., vol. 1, no. 1, Nov. 2024, doi: 10.70274/jaict.2024.1.1.34.

[10] F. Hujainah, R. Binti Abu Bakar, A. B. Nasser, B. Al-haimi, and K. Z. Zamli, "SRPTackle: A semi-automated requirements prioritisation technique for scalable requirements of software system projects," Inf. Softw. Technol., vol. 131, p. 106501, Mar. 2021, doi: 10.1016/j.infsof.2020.106501.

[11] F. Shao, R. Peng, H. Lai, and B. Wang, "DRank: A semi-automated requirements prioritization method based on preferences and dependencies," J. Syst. Softw., vol. 126, pp. 141–156, Apr. 2017, doi: 10.1016/j.jss.2016.09.043.

[12] S.-C. Necula, F. Dumitriu, and V. Greavu-Șerban, "A Systematic Literature Review on Using Natural Language Processing in Software Requirements Engineering," Electronics, vol. 13, no. 11, p. 2055, Jan. 2024, doi: 10.3390/electronics13112055.

[13] F. Dalpiaz, D. Dell'Anna, F. Aydemir, and S. Cevikol, "Requirements Classification with Interpretable Machine Learning and Dependency Parsing," Sep. 2019, pp. 142–152. doi: 10.1109/RE.2019.00025.

[14] A. M. Madni and M. Sievers, "Natural Language Processing To Assess Structure and Complexity of System Requirements," Syst. Eng., vol. 21, no. 3, pp. 172–190, May 2018, doi: 10.1002/sys.21438.

[15] A. Gupta and C. Gupta, "CDBR: A semi-automated collaborative execute-before-after dependency-based requirement prioritization approach," J. King Saud Univ. - Comput. Inf. Sci., vol. 34, no. 2, pp. 421–432, Feb. 2022, doi: 10.1016/j.jksuci.2018.10.004.

[16] F. Noviyanto, R. Razali, and M. Z. A. Nazree, "Understanding requirements dependency in requirements prioritization: a systematic literature review," Int. J. Adv. Intell. Inform., vol. 9, no. 2, pp. 249–272, Jul. 2023, doi: 10.26555/ijain.v9i2.1082.

[17] W. Zhou, Z. Yan, and L. Zhang, "A comparative study of 11 non-linear regression models highlighting autoencoder, DBN, and SVR, enhanced by SHAP importance analysis in soybean branching prediction," Sci. Rep., vol. 14, no. 1, p. 5905, Mar. 2024, doi: 10.1038/s41598-024-55243-x.

[18] G. Y. Quba, H. Al Qaisi, A. Althunibat, and S. AlZu'bi, "Software Requirements Classification using Machine Learning algorithm's," in 2021 International Conference on Information Technology (ICIT), Jul. 2021, pp. 685–690. doi: 10.1109/ICIT52682.2021.9491688.

[19] M. Atas, R. Samer, and A. Felfernig, "Automated Identification of Type-Specific Dependencies between Requirements," in 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI), Santiago: IEEE, Dec. 2018, pp. 688–695. doi: 10.1109/WI.2018.00-10.

[20] H. Guan, H. Xu, and L. Cai, "Requirement Dependency Extraction Based on Improved Stacking Ensemble Machine Learning," Mathematics, vol. 12, no. 9, p. 1272, Jan. 2024, doi: 10.3390/math12091272.

[21] B. Jamasb, S. R. Khayami, R. Akbari, and R. Taheri, "An Automated Framework for Prioritizing Software Requirements," Electronics, vol. 14, no. 6, p. 1220, Jan. 2025, doi: 10.3390/electronics14061220.

[22] A.-G. Núñez, M. Granda, V. Saquicela, and O. Parra, "Machine Learning-Enhanced Requirements Engineering: A Systematic Literature Review:," in Proceedings of the 19th International Conference on Evaluation of Novel Approaches to Software Engineering, Angers, France: SCITEPRESS - Science and Technology Publications, 2024, pp. 521–528. doi: 10.5220/0012688100003687.

[23] T. Amelia and R. Mohamed, "Reprocolla: Requirements Prioritisation Model with Collaboration Perspectives Based on Cost-Value Approach," J. Inf. Commun. Technol., vol. 23, no. 2, pp. 211–252, Apr. 2024, doi: 10.32890/jict2024.23.2.3.

[24] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in Advances in Neural Information Processing Systems, Curran Associates, Inc., 2017. Accessed: Dec. 22, 2025. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html

[25] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in KDD '16. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 785–794. doi: 10.1145/2939672.2939785.

[26] X. Wang, Y. Jin, S. Schmitt, and M. Olhofer, "Recent Advances in Bayesian Optimization," Nov. 11, 2022, arXiv: arXiv:2206.03301. doi: 10.48550/arXiv.2206.03301.

[27] L. Xu et al., "Zero-Shot Cross-Lingual Machine Reading Comprehension via Inter-sentence Dependency Graph," Mar. 15, 2022, arXiv: arXiv:2112.00503. doi: 10.48550/arXiv.2112.00503.

[28] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," J. Big Data, vol. 6, no. 1, p. 60, Jul. 2019, doi: 10.1186/s40537-019-0197-0.

[29] P. Probst, A.-L. Boulesteix, and B. Bischl, "Tunability: Importance of Hyperparameters of Machine Learning Algorithms," J. Mach. Learn. Res. 20 2019 1-32, 2019.

[30] T. Li, X. Zhang, Y. Wang, Q. Zhou, Y. Wang, and F. Dong, "Machine learning for requirements engineering (ML4RE): A systematic literature review complemented by practitioners' voices from Stack Overflow," Inf. Softw. Technol., vol. 172, p. 107477, Aug. 2024, doi: 10.1016/j.infsof.2024.107477.

[31] S. S. Tanveer and Z. A. Rana, "Prioritizing Software Requirements by Combining the Usage Monitoring and User Feedback Data," IEEE Access, vol. 12, pp. 82825–82841, Jan. 2024, doi: 10.1109/access.2024.3409847.

[32] F. Hujainah, R. Bakar, M. Abdulhak, and K. Zamli, "Software Requirements Prioritisation: A Systematic Literature Review on Significance, Stakeholders, Techniques and Challenges," IEEE Access, vol. PP, pp. 1–1, Nov. 2018, doi: 10.1109/ACCESS.2018.2881755.