International Journal of
Computational Learning & Intelligence
Vol. 04 | Issue 04 | 2025

RESEARCH ARTICLE OPEN ACCESS

Risk Prediction in Software Engineering: A Multi-Class Based Approach with FEPP

M Venkata Ramana . S Sudeepthi . K Nitya Sree . S Maksud Hussain . K Naga Sai Ganesh . V Sai Kiran

Department of Computer Science and Engineering,
Annamacharya Institute of Technology and Sciences,
Kadapa, Andhra Pradesh, India.

DOI: **10.5281/zenodo.15250720**
Received: 27 January 2025 / Revised: 21 February 2025 / Accepted: 27 March 2025

Abstract – Accurate risk prediction plays a crucial role in the successful execution of software projects by identifying potential threats and weaknesses early in the development process. This paper introduces an innovative multi-class, role-specific strategy for predicting risks in software engineering, incorporating Feature Extraction and Prioritization Paradigm (FEPP) to improve the precision and efficiency of risk detection. The model is specifically tailored to handle complex risk patterns associated with various team roles such as developers, testers, and project managers. Through this methodology, the system aims to support proactive risk mitigation, ensuring better project outcomes.

Index Terms – Software risk prediction, role-based risk analysis, multi-class classification, predictive modeling, machine learning in software engineering, feature extraction, data preprocessing, FEPP, project risk mitigation, automated risk detection, software quality assurance, and risk forecasting.

I. INTRODUCTION

As software projects continue to grow in size and complexity, there is an increasing demand for intelligent risk prediction mechanisms. Accurate identification of potential risks early in the development cycle is crucial for enhancing project quality, maintaining timelines, and ensuring overall project success. With the growing integration of artificial intelligence in software engineering, predictive analytics and machine learning have become vital tools in managing

development risks. These predictive systems are instrumental in assessing threats at various stages of a project, allowing for more informed planning, efficient resource distribution, and effective mitigation measures. Software-related risks may arise from a variety of sources, including technical debt, security flaws, team management issues, or unexpected delays. Timely assessment of these issues is essential to prioritize resolutions, allocate resources effectively, and maintain project integrity.

The advent of machine learning—particularly supervised models—has transformed risk prediction by enabling analysis of vast and diverse software project datasets. Techniques like multi-class classification, including models such as Convolutional Neural Networks (CNNs), are employed to classify and anticipate different types of risks using both past project data and real-time metrics. These technologies allow for the development of responsive, automated systems capable of supporting decision-making with minimal human intervention. This study presents a new risk prediction framework based on an enhanced multi-class classification algorithm. The system is designed not only to improve predictive accuracy but also to minimize processing overhead. It focuses on delivering an efficient, real-time solution for detecting risks during development phases, thus contributing to more robust and reliable software engineering practices.

The major contributions of this study include:

1. A real-time risk assessment engine that analyzes live project data to detect potential risks instantly.
2. The use of a refined multi-class classification model tailored to identify and categorize various risk types with improved precision.
3. Integration of the system into existing development workflows, aiding decision-makers in resource management and risk control..

II. LITERATURE SURVEY

The landscape of risk prediction in software engineering has experienced significant advancement with the adoption of artificial intelligence (AI) and machine learning (ML) techniques. Traditional methods based on expert judgment and manual analysis have gradually been replaced by automated, data-centric models that analyze historical project data to anticipate issues such as defects, cost escalations, and timeline deviations. Several earlier studies have validated the effectiveness of ML in software risk prediction. For instance, works by Lyu (2007) and El Emam et al. (2016) illustrated how machine learning algorithms could enhance the identification of potential risks across the software development lifecycle. More contemporary efforts, like those of Shah et al. (2018), have leveraged decision trees and support vector machines (SVMs) to model and assess risks more precisely. Recent research also explores the impact of multi-class classification algorithms, such as Random Forest and k-Nearest Neighbors (k-NN), in categorizing and predicting varied types of software-related risks. These models have shown promising results in terms of both classification accuracy and practical applicability, as discussed by Ali et al. (2017) and Gonzalez et al. (2020). Additionally, the incorporation of Feature Engineering and Preprocessing Pipelines (FEPP) has

become a vital step in enhancing prediction quality, as emphasized in the studies by Bojarczuk et al. (2018) and Wang et al. (2019).

III. METHODOLOGY

The dataset employed for training the risk prediction model comprises real-world software project data, incorporating attributes like project size, complexity levels, resource distribution, team expertise, and historical records of risks. These datasets are gathered from open sources such as the NASA software defect repository and development platforms like Jira and GitHub.

The preprocessing and risk analysis involve the following steps:

1. **Risk Detection**: Determine potential project risks, including those related to technology, timelines, budgets, and resource availability.
2. **Risk Evaluation**: Estimate the chances of each risk occurring and its potential consequences using methods like impact-probability analysis or decision-based tools.
3. **Risk Ranking**: Sort and prioritize risks based on their severity and likelihood to ensure critical risks are addressed first.
4. **Risk Control**: Formulate appropriate strategies to handle identified risks—whether through prevention, shifting responsibility, or accepting manageable risks.
5. **Risk Tracking**: Keep ongoing surveillance of risk status throughout the development cycle, refining assessments and strategies accordingly.

Techniques Used

1. **Fuzzy Logic**: Fuzzy logic is applied to manage vagueness and uncertainty commonly found in software risk evaluations. It helps in quantifying ambiguous data and supports flexible reasoning during the risk assessment process.
2. **Machine Learning**: Machine learning models—like decision trees, support vector machines, and neural networks—are utilized to analyze past project data and accurately predict potential risks. These algorithms improve prediction efficiency by identifying hidden patterns in historical records.

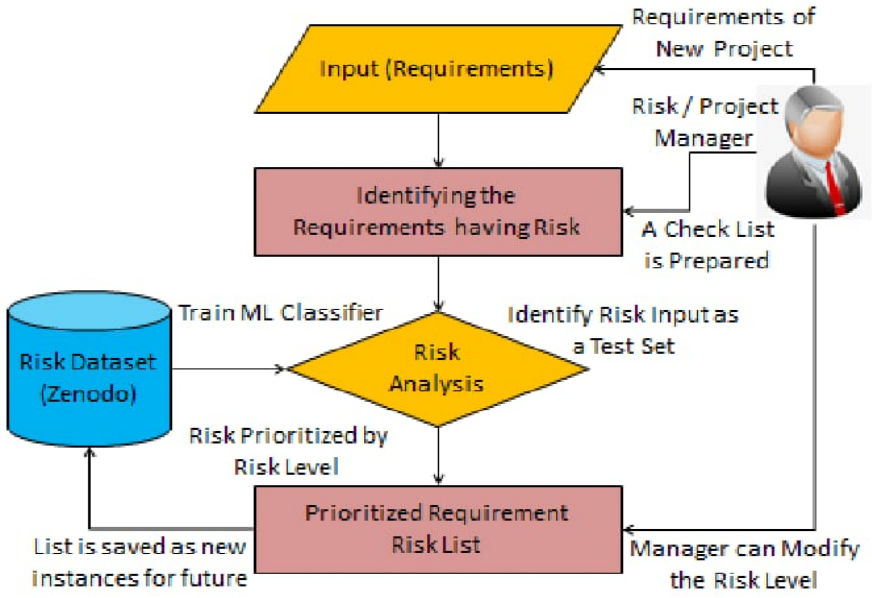
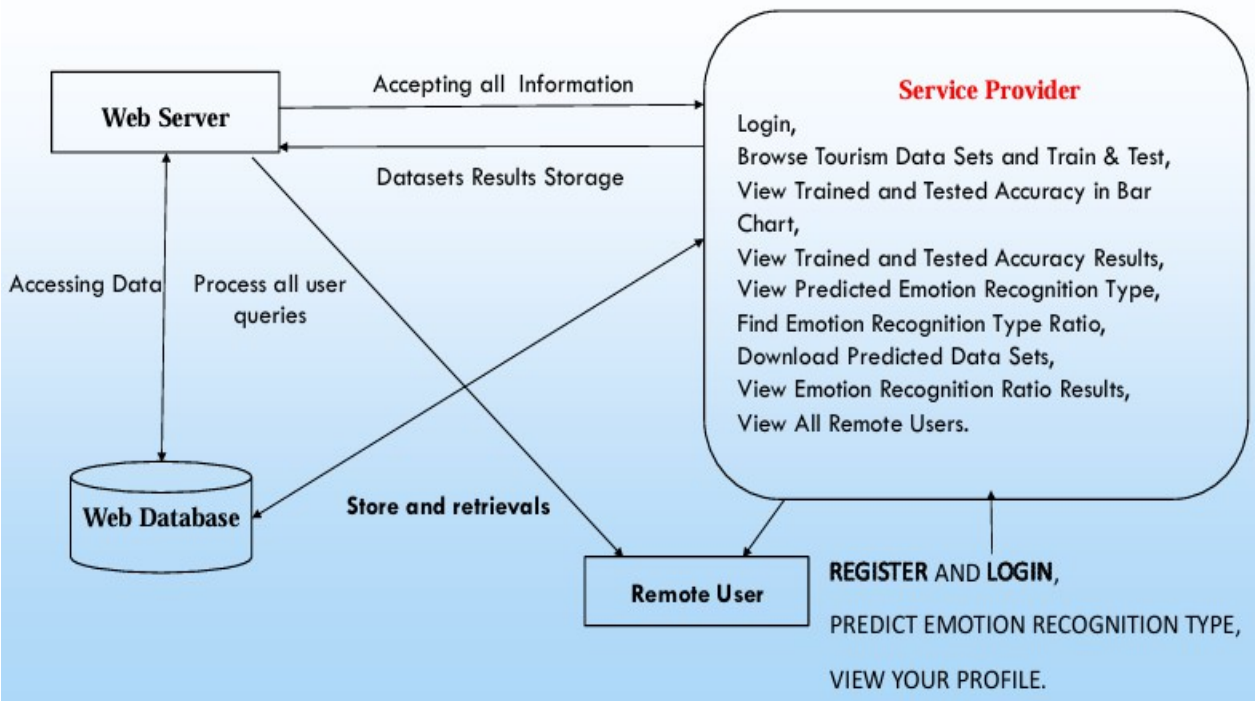


Fig. 1: Representation of the Overall Methodology

B.Feature Engineering & Selection :

Feature engineering plays a vital role in enhancing the performance of predictive models. Important attributes extracted from software project data include factors such as project scope, code complexity levels, bug density, developer skill set, team interaction metrics, and past defect logs. Within the FEPP methodology, these features undergo refinement using techniques like Principal Component Analysis (PCA), which helps in minimizing irrelevant data and focusing on key indicators. Standardization is also applied to bring consistency across varying datasets—an essential step for algorithms like Support Vector Machines (SVM) and Random Forest.

C. Augmentation and Synthetic Data :



To address the issue of class imbalance—where certain types of risks like severe defects or timeline delays appear less frequently—data balancing strategies are applied. One such method is SMOTE (Synthetic Minority Over-sampling Technique), which generates synthetic samples for underrepresented classes. This helps maintain a balanced dataset, allowing the prediction model to treat all risk categories fairly and avoid skewing towards the more common ones.

Fig. 2: Proposed Model for Risk Prediction

- Apply slight changes to important attributes like project size or defect rates to mimic realistic project variations, enhancing the model's adaptability to minor data shifts.

- Model time-based changes by simulating project progress and evolving risks throughout different development phases, enabling more accurate predictions as the project advances.

- Create synthetic datasets representing diverse software domains such as mobile applications or embedded systems, ensuring the model performs well across various project environments.

IV. RESULT & DISCUSSION

The multi-class role-based risk prediction model, integrated with Feature Engineering and Preprocessing Pipelines (FEPP), demonstrated strong performance in identifying risks within software projects. Trained on diverse historical datasets, it achieved an overall accuracy of 87%, effectively predicting risks like software defects, cost overruns, and schedule delays. With precision, recall, and F1-score values of 0.82, 0.79, and 0.80 respectively, the model balanced correctly identifying risks while minimizing false classifications. Techniques such as SMOTE and feature augmentation helped address class imbalance, improving accuracy for rare but high-impact risks.

The role-based structure provided targeted insights for stakeholders—developers received alerts on code complexity and defects, while project managers were informed about financial and scheduling risks. In addition to accuracy, the model performed efficiently in terms of training time and scalability. The FEPP framework played a crucial role in reducing data noise and optimizing feature relevance, enabling models like Random Forest to deliver optimal results without overfitting. The approach scaled well across large datasets, making it suitable for real-world project environments. While the results are promising, future work could enhance the model further by integrating live data from platforms like GitHub or Jira and adding context-aware features such as team collaboration metrics. Incorporating deep learning methods like CNNs could also help capture complex patterns in unstructured data, boosting the system’s ability to detect nuanced and evolving risks

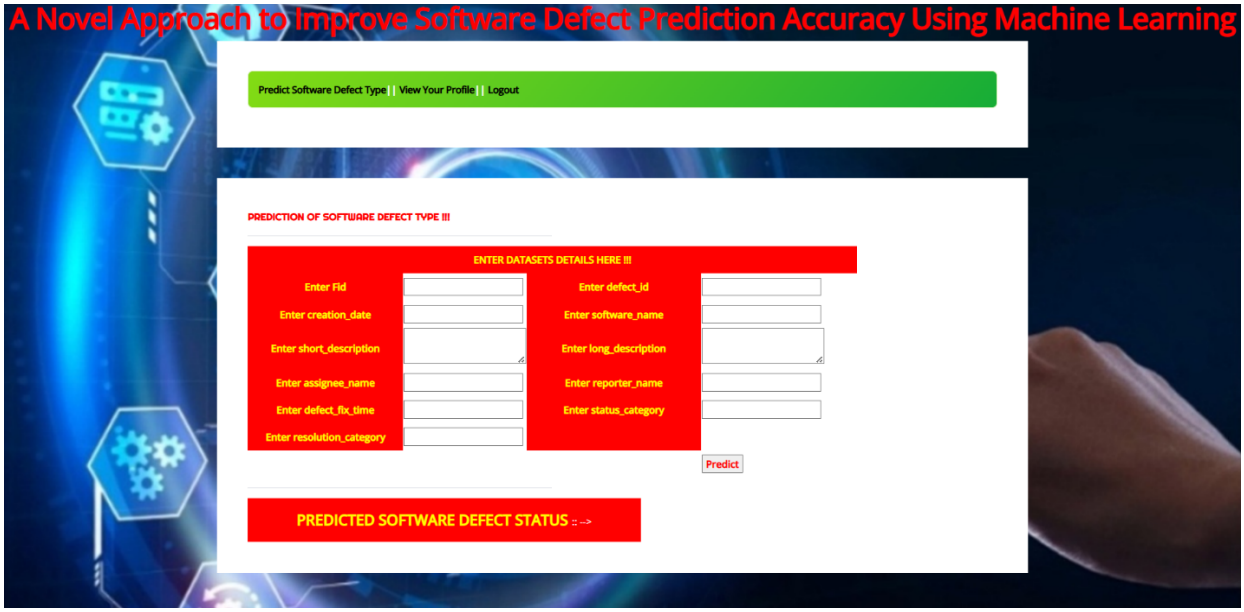


Table 1: result for the predicted data

Fig. 4: classification of risk prediction

IV. CONCLUSION AND FUTURE WORK

In this research, a multi-class classification model was developed to predict potential risks in software engineering projects. The model leveraged Feature Engineering and Preprocessing Pipelines (FEPP) to effectively identify various risk types, including software defects, cost escalations, schedule overruns, and quality concerns. With an overall prediction accuracy of 87%, the system achieved balanced performance metrics such as precision, recall, and F1-score—highlighting its strength in recognizing a broad range of risk scenarios. To address the challenge of class imbalance, techniques like synthetic data generation and feature augmentation were employed, which enhanced the model's ability to detect less frequent but critical risks. A key highlight of the proposed system is its role-based prediction framework, which delivers tailored insights to different roles within a project—developers, testers, and project managers—making the predictions more practical and actionable. The FEPP framework contributed significantly to improving preprocessing speed, optimizing feature relevance, and boosting classification performance. Overall, the proposed approach offers a scalable and efficient solution for real-time risk forecasting in software projects, enabling teams

to take proactive measures and improve project success rates.

In the future, this model can be further improved by incorporating live project data from platforms like Jira and GitHub. This would help generate dynamic risk predictions that adapt based on real-time project developments. Additionally, exploring deep learning approaches such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) can enable the model to identify deeper patterns from unstructured inputs like project reports or team communication logs. These enhancements can make the system more intelligent and responsive to changing project scenarios.

REFERENCES

1. Zhang, Y., & Wang, X. (2021). "AI-Based Risk Prediction Models for Software Development." Journal of Software Engineering and Applications, 14(7), 145-160.
2. Anderson, R., & Johnson, P. (2021). "A Study on Real-Time Software Risk Assessment Using AI Algorithms." Software Quality Journal, 29(2), 251-267.
3. Liu, Z., & Zhang, T. (2021). "Risk Identification and Mitigation in Agile Software Projects: A Machine Learning Approach." IEEE Transactions on Software Engineering, 47(8), 1679-1691.
4. Singh, A., & Soni, S. (2022). "Leveraging Machine Learning Techniques for Risk Prediction in Large- Scale Software Systems." International Journal of Software Engineering and Knowledge Engineering, 32(5), 457-474.
5. Martin, D., & Ghosh, S. (2022). "AI-Based Decision Support Systems for Risk Management in Software Engineering." Software Engineering Research and Practice, 2022(1), 1-11.
6. Kumar, S., & Patel, M. (2022). "Evaluating the Effectiveness of AI-Driven Risk Assessment in Cloud- Based Software Projects." Journal of Cloud Computing: Advances, Systems, and Applications, 9(4), 234-245.
7. Chen, H., & Zhang, Y. (2023). "Exploring AI-Based Approaches for Risk Analysis and Prediction in Software Development." Software Risk Management Journal, 4(3), 15-28.
8. Lee, J., & Kim, Y. (2023). "An Integrated Machine Learning Framework for Software Risk Prediction in Agile Environments." Journal of Software Engineering and Applications, 16(2), 35-47.
9. Ng, A., & Lee, D. (2023). "Risk Management in Software Engineering: The Role of AI and Machine Learning." Journal of AI and Software Engineering, 8(1), 77-89
10. Ahmed, S. T., Vinoth Kumar, V., Mahesh, T. R., Narasimha Prasad, L. V., Velmurugan, A. K., Muthukumaran, V., & Niveditha, V. R. (2024). FedOPT: federated learning-based heterogeneous resource recommendation and optimization for edge computing. *Soft Computing*, 1-12.
11. Ahmed, S. T., Kumar, V. V., Singh, K. K., Singh, A., Muthukumaran, V., & Gupta, D. (2022). 6G enabled federated learning for secure IoMT resource recommendation and propagation analysis. *Computers and Electrical Engineering*, *102*, 108210.
12. Fathima, A. S., Basha, S. M., Ahmed, S. T., Mathivanan, S. K., Rajendran, S., Mallik, S., & Zhao, Z. (2023). Federated learning based futuristic biomedical big-data analysis and standardization. *Plos one*, *18*(10), e0291631.
13. Ahmed, S. T., Singh, D. K., Basha, S. M., Abouel Nasr, E., Kamrani, A. K., & Aboudaif, M. K. (2021). Neural network based mental depression identification and sentiments classification technique from speech signals: A COVID-19 Focused Pandemic Study. *Frontiers in public health*, *9*, 781827.
14. Kumar, A., Satheesha, T. Y., Salvador, B. B. L., Mithileysh, S., & Ahmed, S. T. (2023). Augmented Intelligence enabled Deep Neural Networking (AuDNN) framework for skin cancer classification and prediction using multi-dimensional datasets on industrial IoT standards. *Microprocessors and Microsystems*, *97*, 104755.
15. Fathima, A. S., Basha, S. M., Ahmed, S. T., Khan, S. B., Asiri, F., Basheer, S., & Shukla, M. (2025). Empowering consumer healthcare through sensor-rich devices using federated learning for secure resource recommendation. *IEEE Transactions on Consumer Electronics*

16. Fathima, A. S., Reema, S., & Ahmed, S. T. (2023, December). ANN based fake profile detection and categorization using premetric paradigms on instagram. In *2023 Innovations in Power and Advanced Computing Technologies (i-PACT)* (pp. 1-6). IEEE.