Scholars Academic Journal of Biosciences
Abbreviated Key Title: Sch Acad J Biosci
ISSN 2347-9515 (Print) | ISSN 2321-6883 (Online)
Journal homepage: https://saspublishers.com

OPEN ACCESS

Biosciences

AI-Driven Risk Assessment and Mitigation Strategies for Optimizing Project Success

Nageeta Kumari[1*], Misbah Ullah Khan[2], Zahoor Ul Haq[3], Tanveer Ali[4], Muhammad Inam ul haq[5], Hina Saeed[6], Mujahid Rasool[7], Habib Ur Rehman[8]

[1]National University of computer and emerging science (FAST-NUCES)
[2]School of Management, Wuhan University of Technology
[3]School of Economics and Finance, Xi'an Jiaotong University
[4]School of Resource and Environmental Engineering, Wuhan University of Technology, China
[5]Department of Electronic Engineering, Faculty of Engineering and Technology
[6]Department of Computer science, Muhammad Ali jinnah university karachi, Pakistan
[7]Department of Artificial Intelligence, The Islamia University of Bahawalpur
[8]Centre of Data Science, Government College University Faislabad, Punjab Pakistan

DOI: https://doi.org/10.36347/sajb.2025.v13i02.007 | **Received:** 30.12.2024 | **Accepted:** 06.02.2025 | **Published:** 15.02.2025

***Corresponding author:** Nageeta Kumari
National University of Computer and Emerging Sciences (FAST-NUCES)

Abstract **Original Research Article**

In this paper, we present a predictive analytics framework for risk assessment and mitigation in software development projects, aimed at enhancing project quality and success rates. Software projects often face challenges due to their dynamic and complex nature, making early risk detection crucial. To address this, we collected data from developers, project managers, and QA engineers using targeted sampling techniques. The dataset underwent preprocessing, including extensive cleaning and transformation, before being analyzed using supervised machine learning models: K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Decision Trees, and Logistic Regression. These models were evaluated for their predictive accuracy and integrated into a real-time Stream lit application for live risk assessment and decision support. The results demonstrate the effectiveness of predictive analytics in identifying correlations between project parameters and potential quality risks, enabling proactive measures to address them. Among the models, Support Vector Machines exhibited superior performance, effectively handling high-dimensional project datasets. The developed Stream lit application allows project teams to visualize risk predictions, supporting informed and immediate decision-making. This research contributes to advancing software project management by introducing a proactive, data-driven approach to quality assurance and risk mitigation. Future research can explore integrating additional machine learning models and expanding datasets to further validate and optimize predictive performance.

Keywords: Predictive Analytics, Risk Assessment, Risk Mitigation, Machine Learning Models, Proactive Risk Management.

1. INTRODUCTION

In today's fast-paced software development landscape, delivering high-quality outcomes remains a persistent challenge due to the increasing complexity and dynamic nature of projects. Traditional risk management approaches often adopt a reactive stance, addressing issues only after they have emerged, which frequently results in project delays, budget overruns, and compromised software quality. This reactive methodology underscores the need for proactive, data-driven strategies to identify and mitigate potential risks early in the project lifecycle. Predictive analytics, a branch of advanced analytics that leverages historical data to forecast future events, has shown significant potential in addressing these challenges across various industries.

In software development, predictive analytics enables the early detection of risks and quality issues by uncovering patterns and trends that might remain unnoticed through conventional analysis methods. This research explores the integration of predictive analytics into software project management using supervised machine-learning models, including K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Decision Trees, and Logistic Regression. These models are trained on data collected from developers, project managers, and QA engineers to create robust classifiers capable of predicting potential risks and quality-related setbacks.

To ensure practical applicability, the predictive models are embedded into a real-time Stream lit-based web application, providing project stakeholders with dynamic visualization and actionable insights. This integration empowers project managers to make informed, data-driven decisions, transitioning from reactive responses to proactive strategies. Furthermore, the study identifies critical risk indicators and correlations between project parameters and quality outcomes, offering valuable insights for improved risk mitigation practices.

The significance of this research lies in its ability to bridge the gap between theoretical predictive analytics models and their practical application in software project management. By enabling early detection and resolution of potential risks, this approach facilitates better resource allocation, reduced costs, and improved software quality. This study marks a meaningful advancement in the field, demonstrating the transformative potential of predictive analytics in enhancing software project outcomes.

To further explore these contributions, this research is guided by the following questions:

1. How can predictive analytics be effectively integrated into the risk assessment and quality management processes of software projects to enhance overall product quality and project success rates?
2. What are the key data sources and critical variables necessary for implementing predictive analytics in the risk assessment and quality management of software projects?
3. How do the size and complexity of software projects affect the accuracy and effectiveness of predictive analytics models in risk assessment and quality management?

The data collection process involves gathering comprehensive information from developers, project managers, and QA engineers. Data preparation includes cleaning (handling missing values, removing duplicates), transformation (normalization, feature engineering), and splitting into training and testing sets. Model training involves feeding the models with training data and adjusting their parameters to minimize prediction errors. Evaluation metrics such as accuracy, precision, recall, and F1 score assess model performance, with cross-validation techniques ensuring generalizability.

Various studies and methodologies have been explored to enhance risk assessment and mitigation in software projects through predictive analytics. This section delves deeper into the relevant research, highlighting key approaches and findings that have informed the development of predictive models for software project management. The adoption of Industry 4.0 technologies, as observed in various studies, marks a significant shift in manufacturing and quality management practices. These technologies encompass smart materials, predictive analytics, and big data utilization, showcasing their potential to revolutionize industries [1]. Research emphasizes the impact of Industry 4.0 technologies on manufacturing, revealing how they optimize material quality, enhance smart logistics and supply chains, and solve intricate manufacturing challenges [2]. Papers indicate that these technologies enable cost reduction through optimized inventory management and minimized material wastage, with remote collaboration bridging communication gaps between on-site and off-site employees. In the context of small and medium-sized enterprises (SMEs) in Indian manufacturing, big data analytics emerges as a critical factor influencing project performance [3]. Studies highlight its mediating role, particularly in project knowledge management, green purchasing, and operational capabilities, offering insights for SME managers and policymakers to navigate challenges and improve project efficiency. Advancements in quality management, termed Quality 4.0, underscore the role of predictive analytics in proactively addressing quality issues. However, the lack of standardized methodologies poses challenges. Researchers propose leveraging established frameworks like CRISP-DM to simplify predictive analytics projects in quality management, enabling industry- specific applications without deep data science expertise [4,5]. Additionally, Predictive Software Engineering (PSE) has surfaced as a transformative framework in software development. Drawing from over 27 years of software development experience, PSE's seven principles ensure transparency, controllability, and predictability [4]. This framework optimizes custom product development processes, enhancing project transparency and budget management. Collectively, these studies shed light on how Industry 4.0 technologies, big data analytics, predictive analytics, and specialized frameworks like PSE revolutionize manufacturing, quality management, and software development practices, offering avenues for efficiency enhancement and quality improvement.

Desalegn Taye and Feleke propose a novel machine learning model aimed at predicting project management knowledge areas (PMKAs) failure in software companies. The study focuses on ten knowledge areas in project management, emphasizing the significance of understanding these areas to mitigate project failures [6]. The use of machine learning techniques like Support Vector Machines, Decision Trees, and Logistic Regression presents promising results, with Support Vector Machines exhibiting superior performance. However, the study suggests the need for more extensive datasets and comparisons to refine predictions. By leveraging machine sensor values during production, the model predicts product quality and identifies correlations between machine status and faulty product occurrence [7]. The outcomes underscore the potential of predictive analytics in optimizing quality assurance processes within a controlled industrial environment. Olaleye *et al.*'s systematic review focused on machine learning-based software defect severity prediction. They aimed to address crucial gaps in existing research, including insufficient consideration of bias-variance tradeoffs and the neglect of text analytics techniques like NLP in previous reviews [8]. By formulating eleven research questions and employing a systematic approach, they highlighted these gaps and offered valuable insights for future research directions in software quality assurance using text analytics. Their study emphasizes the importance of addressing internal and external validity concerns for robust predictive analytics in software defect severity prediction. The significance of defect prediction in ensuring the quality of software products cannot be overstated. Recent technological advancements and the rapid growth in software applications have accentuated the need for robust defect prediction techniques. Thota, Shajin, and Rajesh's study underscores the relevance of defect prediction in supporting developers and expediting time-to-market for reliable software products [9]. The paper emphasizes the necessity of defect prediction techniques in allocating resources efficiently for software validation. Soft computing-based machine learning techniques are proposed to optimize features and enhance defect prediction accuracy. Abaei and Selamat examine various prediction approaches, employing machine learning techniques such as decision trees, neural networks, and Naïve Bayes for software fault prediction [10]. The study assesses these techniques using public NASA datasets, emphasizing the performance of algorithms like random forest and Naïve Bayes based on dataset sizes and feature selection approaches. Catal's study delves into semi-supervised classification approaches for software defect prediction when labeled data points are limited. The evaluation of four methods—Low-density separation (LDS), support vector machine (SVM), expectation-maximization (EM-SEMI), and class mass normalization (CMN)—on NASA datasets demonstrates the effectiveness of SVM and LDS algorithms, particularly in scenarios with limited fault data [11]. McAdam, Miller, and McSorley present a perspective of Quality Management (QM) using Contingency Theory within Small and Medium Sized Enterprises (SMEs) [12]. The study emphasizes the interplay between Contingency Variables (strategy, culture, lifecycle, and customer focus) and QM practices, shaping strategic alignment. It underscores the uniqueness of each SME’s environment in adopting QM practices rather than a one-size-fits-all approach. Sultanow, Konopik, Ullrich, and Vladova delve into Machine Learning's applicability in Static Code Analysis for Software Quality Assurance [13]. The paper showcases how Machine Learning, beyond predictive analytics, aids in identifying potential errors in large-scale IT processes, particularly in critical environments such as the public sector. It introduces an approach utilizing Machine Learning for Static Code Analysis, uncovering hidden failure potentials that traditional analysis methods might overlook. These studies collectively highlight the shift towards adaptive approaches in managing software quality. While Contingency Theory sheds light on the contextualized application of QM practices, defect prediction techniques leverage advanced technologies like soft computing and machine learning. Additionally, the use of Machine Learning in Static Code Analysis illustrates its potential to enhance fault detection in critical software environments. However, despite these advancements, there are gaps in understanding the integration of these techniques within real-world software project management frameworks. The literature review sets the stage for further exploration into the practical implementation and synergy of predictive analytics, defect prediction, and Machine Learning in optimizing software project outcomes. The reviewed literature underscores the increasing significance of predictive analytics and machine learning in quality management for software projects. Studies by Desalegn Taye and Feleke emphasize predictive models' potential in mitigating project failures, aligning with this thesis' aim to apply predictive analytics for enhanced quality management. Additionally, gaps highlighted in Olaleye *et al.*'s systematic review and Sultanow, Konopik, Ullrich, and Vladova's study emphasize the need for practical integration of predictive analytics and machine learning within software project management frameworks, affirming the relevance and importance of this research. Real-time data analysis and visualization are crucial for effective project management. By integrating predictive models into real-time applications, project managers can continuously monitor project health, identify emerging risks, and make timely decisions. Several industry tools provide real-time risk assessment, predictive dashboards, and automated alerts, helping project teams stay informed and proactive. The practical implications of these integrations are significant, enabling immediate insights into potential risks and reducing the likelihood of project failures. The use of applications like Stream lit for real-time predictions exemplifies the practical relevance and impact of predictive analytics in software project management. This integration allows stakeholders to visualize predictions and make informed decisions promptly, transforming traditional reactive approaches into proactive strategies. Despite substantial research on predictive analytics in software project management, several gaps remain. Few studies have focused on the real-time application of predictive models, and there is limited research on the comprehensive integration of multiple predictive models for holistic risk management. Additionally, more research is needed on tailoring predictive models to the needs of different stakeholders in software projects. This thesis addresses these gaps by developing and integrating multiple predictive models into a real-time Stream lit application, providing a comprehensive approach to risk assessment and mitigation in software projects. Engaging industry professionals ensures the practical relevance and applicability of the predictive models. The literature review has highlighted the importance of predictive analytics in software development, particularly for risk assessment and quality management. Traditional risk management approaches have significant limitations, and predictive analytics offers a promising solution. Various supervised learning models, including KNN, SVM, Naive Bayes, decision trees, and logistic regression, have been applied successfully in related work, demonstrating their potential in predicting software defects and project failures. The integration of predictive models into real-time applications represents a significant advancement, providing practical tools for proactive risk management. However, gaps in the current literature indicate the need for further research, which this thesis aims to address. By developing and implementing predictive models in a real-time application, this research contributes to advancing the field of software project management through data-driven strategies.

2. METHODOLOGY

The research employs an experimental methodology. Experimental research encompasses various research designs that involve manipulation and controlled testing to gain a deeper understanding of processes that predict outcomes based on specific criteria. Accordingly, the following methods and techniques are utilized to conduct this study.

2.1 The designed proposed prediction model

The proposed prediction model for assessing and mitigating risks in software projects comprises five key phases. Initially, project data is collected from software development companies, with a focus on inputs from developers, project managers, and QA engineers. Following this, the data undergoes pre-processing, which involves cleaning, feature selection, transformation, and reduction to refine the dataset for analysis.

Once the data is prepared, selected machine learning algorithms such as Support Vector Machine (SVM), Decision Trees (DT), Naïve Bayes (NB), Logistic Regression (LR), and K-Nearest Neighbors (KNN) are implemented. To optimize the performance of these predictive models, hyperparameter tuning is performed using Grid Search with Cross-Validation (Grid Search CV). This method systematically explores multiple combinations of parameter settings, validates each combination, and identifies the optimal set of parameters. This fine-tuning process enhances the accuracy, precision, recall, and F1-score of the algorithms, ensuring robust predictive performance.

Subsequent analysis evaluates the efficiency of the proposed models using performance metrics such as accuracy, precision, F1-score, and recall. The final phase involves drawing conclusions based on graphical and aggregated experimental results, providing insights into the effectiveness of the models.

Regarding the impact of project size and complexity on predictive analytics models, larger and more complex projects introduce more variables and potential risks, increasing the challenge of accurate predictions while also underscoring their importance. To address this, the models are trained on diverse datasets representing various project sizes and complexities, ensuring accuracy and reliability across different scenarios. This approach is further reinforced through a comparative analysis of different supervised learning models. The interconnected and sequential flow of these phases ensures a thorough and systematic approach to risk assessment and mitigation in software projects, as illustrated in Fig. 1.

2.2 Data collection and dataset preparation

Data collection for this study involved administering a questionnaire to gather responses from key stakeholders in software projects, including developers, project managers, and QA engineers. The questionnaire was designed to explore perspectives on aspects crucial to software project success, categorized broadly into criteria such as functionality, reliability, efficiency, and user satisfaction. The dataset was structured to include attributes pertinent to the criteria of functionality, reliability, efficiency, and user satisfaction, chosen for their alignment with the study's emphasis on risk assessment and mitigation using predictive analytics in software projects. These attributes were carefully prepared to ensure data quality and suitability for subsequent analysis with a binary classifier. To maintain data integrity and reliability, measures such as data cleaning to handle missing values, normalization to standardize numerical data, and feature selection to optimize the dataset for predictive modeling were employed. These steps were crucial in ensuring that the dataset met the criteria of functionality, reliability, efficiency, and user satisfaction required for effective risk prediction and mitigation strategies in software project management. Key data sources include inputs from developers, project managers, and QA engineers, focusing on aspects such as functionality, reliability, efficiency, and user satisfaction. Critical variables include project parameters like development timelines, defect rates, resource allocation, and stakeholder feedback. These variables are essential in creating robust predictive models that can effectively identify potential risks and quality issues.

2.2.1 Analyzing attributes

2.2.1.1 Functionality: Each attribute should accurately represent the functional requirements of the software project. Attributes are evaluated based on their contribution to the overall functionality, with possible values being Yes (Y) or No (N). Attributes that enhance functionality were selected.

2.2.1.2 Reliability: Each attribute should contribute to the reliability of the software project, ensuring consistent performance under specified conditions. Attributes are rated as High (H), Medium (M), or Low (L). Attributes with higher reliability values were chosen.

2.2.1.3 Efficiency: Each attribute should improve the efficiency of the software project, optimizing resource usage and performance. Attributes are rated as High (H), Medium (M), or Low (L). Attributes with higher efficiency values were selected.

2.2.1.4 User Satisfaction: Each attribute should positively impact user satisfaction, ensuring that the end-user experience is favorable. Attributes are rated as High (H), Medium (M), or Low (L). Attributes that contribute significantly to user satisfaction were chosen.

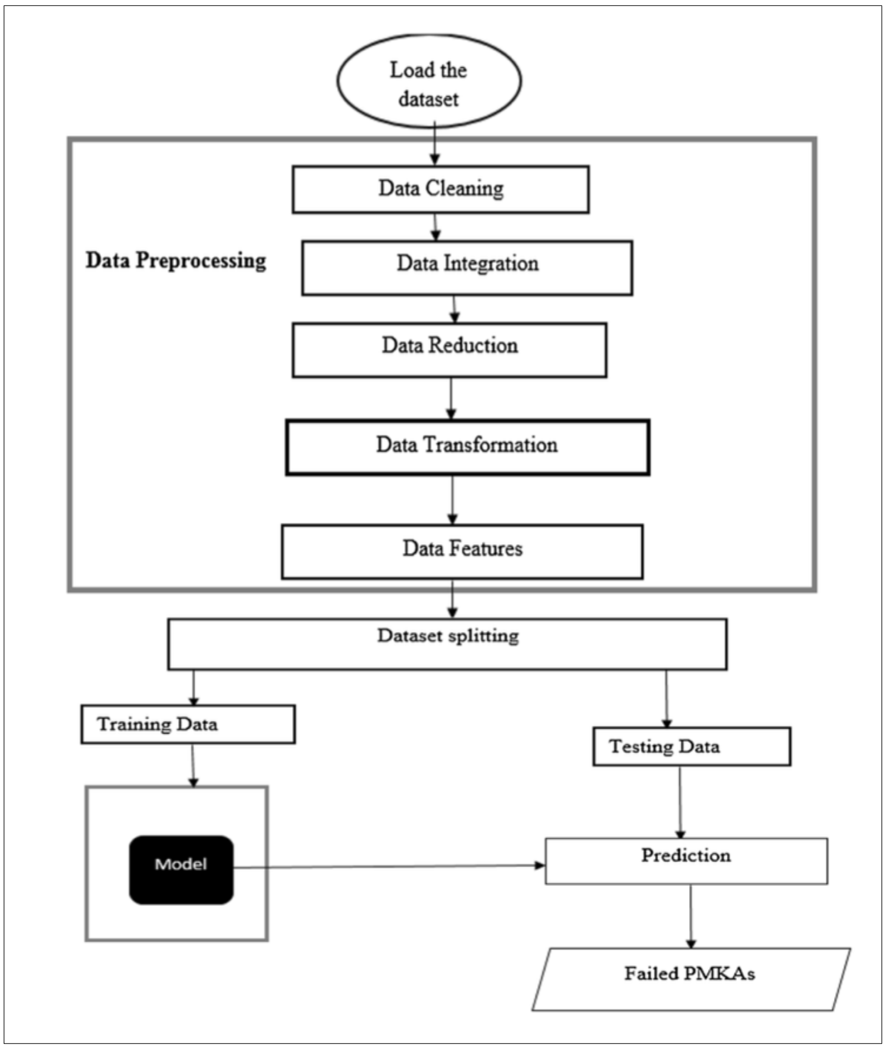

Fig. 1: The proposed model

Table 1: Risk assessment and mitigation after annotating the data

Role	**Number of responses**
Developers	146
Project Managers	91
QA Engineers	90
Total	**327**

Based on these criteria, a characteristic may be added or removed from the final list of influential attributes. Table 3 summarizes the possible values for each criterion: High (H), Medium (M), and Low (L). Considering their functionality, reliability, efficiency, and user satisfaction. The final attributes selected were chosen because they demonstrated a higher level of relevance and importance in the context of risk assessment and mitigation in software projects.

2.3 Data preprocessing

As a result, we conducted comprehensive data preprocessing, which involved data cleansing, removal of duplicate values, detection and correction of null values, and data balancing. This marks the completion of the preprocessing phase. Given the diverse sources of our data, data integration became a crucial part of the process. We aimed to create a condensed version of the dataset that is smaller in size but retains the integrity of the original data. Data preparation involved transforming the data into a format suitable for predictive analytics modeling, such as converting character values to binary values. To evaluate the performance of our machine learning models, we employed the train-test split technique. This approach involves dividing the data into two sets: a training dataset, used to fit the machine learning model, and a test dataset, used to assess the performance of the machine learning model. The purpose of splitting the dataset is to evaluate the model's performance on new data that was not used during the training phase. This methodology reflects our practical goal: to fit the model to existing data with known inputs and outputs and then use it to predict future outcomes where the target values are unknown. By doing so, we aim to enhance risk assessment and mitigation strategies in software projects using predictive analytics.

2.3.1 Model evaluation

This activity oversees describing the evaluation parameters of the designed model and its results. The comparison was made between the data categorized by the proposed model system and the manually labeled (categorized) data. Having a common performance appraisal metric for classification and classification accuracy (CA) is used as the final proof of performance.

2.3.1.1 Confusion matrix

The confusion matrix assesses the performance of a classification or classifier model on a test dataset. Our target class was binary classifier, which means classification tasks that have only two class labels. The performance of a classification model is defined by a confusion matrix. True positives (TP): cases where the classifier predicted that the true and correct class was true. True negatives (TN): cases in which the model predicted the false and correct class was false. False positives (FP) (type I error) - Classes predicted true but the correct class was false. False negatives (FN) (type II error): The classifier predicted false, but the correct class was false.

2.3.1.2 Accuracy

Accuracy means the number of all misclassified samples divided by the total number of samples in the dataset. Accuracy has the best value of one and the worst value of zero.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

2.3.1.3 Precision

Precision (P)—precision is the fraction or percentage of identified or retrieved instances that the classification algorithm considers important. High precision means that most items labeled, for example, as "positive" belong to the class "positive" and is defined as precision characterized as the number of isolated true positives times the total sum of true positives and false positives.

$$\text{Precision} = \frac{TP}{TP + FP}$$

2.3.1.4 Recall

A recall is considered a measure of completeness, which is the level of positive examples that are marked as positive. Cluster revision is characterized by the number of isolated true positives times the total number of components that have a place with the positive classes.

$$\text{Recall} = \frac{TP}{TP + FN}$$

2.3.1.5 F1 Score

F-Measure (F1 score) is defined as the harmonic means of precision and recall which is a measure that joins recall and precision into a single measure of performance. The F1-score was calculated by averaging precision and recall. The relative contribution of precision and recall to the F1-score are equal.

$$\text{F1} - \text{score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

2.3.1.6 ROC Curve and AUC

The ROC curve is a graph that shows the performance of a binary classifier by plotting the true positive rate (TPR) against the false positive rate (FPR) at different thresholds. TPR is also called recall or sensitivity, and FPR measures how many negative cases are incorrectly classified as positive. The Area Under the Curve (AUC) summarizes the model's performance across all thresholds, with values closer to 1 indicating better performance. ROC and AUC help evaluate the balance between sensitivity and specificity in a predictive model.

2.4 Introduction to Stream lit Application

A Stream lit web application was developed to demonstrate the practical use of a predictive model for risk mitigation. Stream lit, a Python library, allows the creation of interactive web apps with minimal code, making it ideal for showcasing data science and machine learning projects. The app predicts the success or failure of risk mitigation strategies based on user input, such as risk factors and mitigation methods. It features a simple interface where users can enter data, click a "Predict" button, and instantly receive a prediction on the success of their mitigation efforts. The app displays the outcome along with insights or recommendations, helping to illustrate the model's real-world utility in enhancing decision-making for risk management.

3. RESULTS AND DISCUSSION

Experimentation in this study required the preparation of a dataset for training and testing, as no free, ready-to-use dataset was available online. We utilized data from Software Companies.

3.1 Experimental results and analysis

After importing the necessary Python modules and libraries, the initial step involved reading the processed data frame (df) into Python and verifying the imported rows.

The dataset includes the following attributes: Experience, Primary Role, Used Predictive Analytics, Predictive Analytics Reliance, Limitations, Predictive Analytics. Benefits, Collaboration Importance, Challenges, Bias Concerns, and Critical Data Sources. A new target variable was created by combining the attributes Predictive Analytics.

Reliance, Challenges, and Predictive Analytics Benefits. The attributes Experience Primary Role, Used Predictive Analytics, Limitations, Collaboration Importance, Bias Concerns, and Critical Data Sources were retained as features for the analysis.

3.1.1 Results of each prediction algorithm

We employed five methods to predict the Risks in Software Projects in our experiment. K-Nearest Neighbors (KNN), Decision Trees (DT), Logistic Regression (LR), Naive Bayes (NB), and Support Vector Machines (SVM) are all examples of machine learning algorithms.

Table 2: Support Vector Machine (SVM) result using the confusion matrix Benefits, Collaboration

Roles	Classes	Precision	Recall	F1-score	Support	Correctly Predicted	Incorrectly Predicted
Project Managers	0	0.79	1.00	0.88	11	21	3
	1	1.00	0.77	0.87	13		
	Accuracy			0.88	24		
	Mac avg	0.89	0.88	0.87	24		
	Weighted avg	0.90	0.88	0.87	24		
QA's	0	0.95	1.00	0.97	18	22	1
	1	1.00	0.80	0.89	5		
	Accuracy			0.96	23		
	Mac avg	0.97	0.90	0.93	23		
	Weighted avg	0.96	0.96	0.95	23		
Developers	0	0.92	0.92	0.92	26	39	4
	1	0.88	0.88	0.88	17		
	Accuracy			0.91	43		
	Mac avg	0.90	0.90	0.90	43		
	Weighted avg	0.91	0.91	0.91	43		

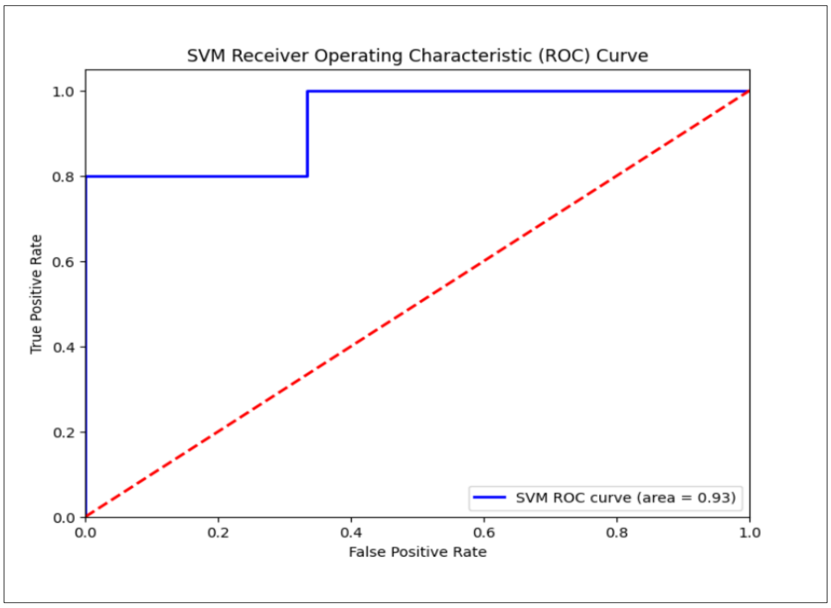

Figure 2: SVM ROC Graph (Project Managers Dataset)

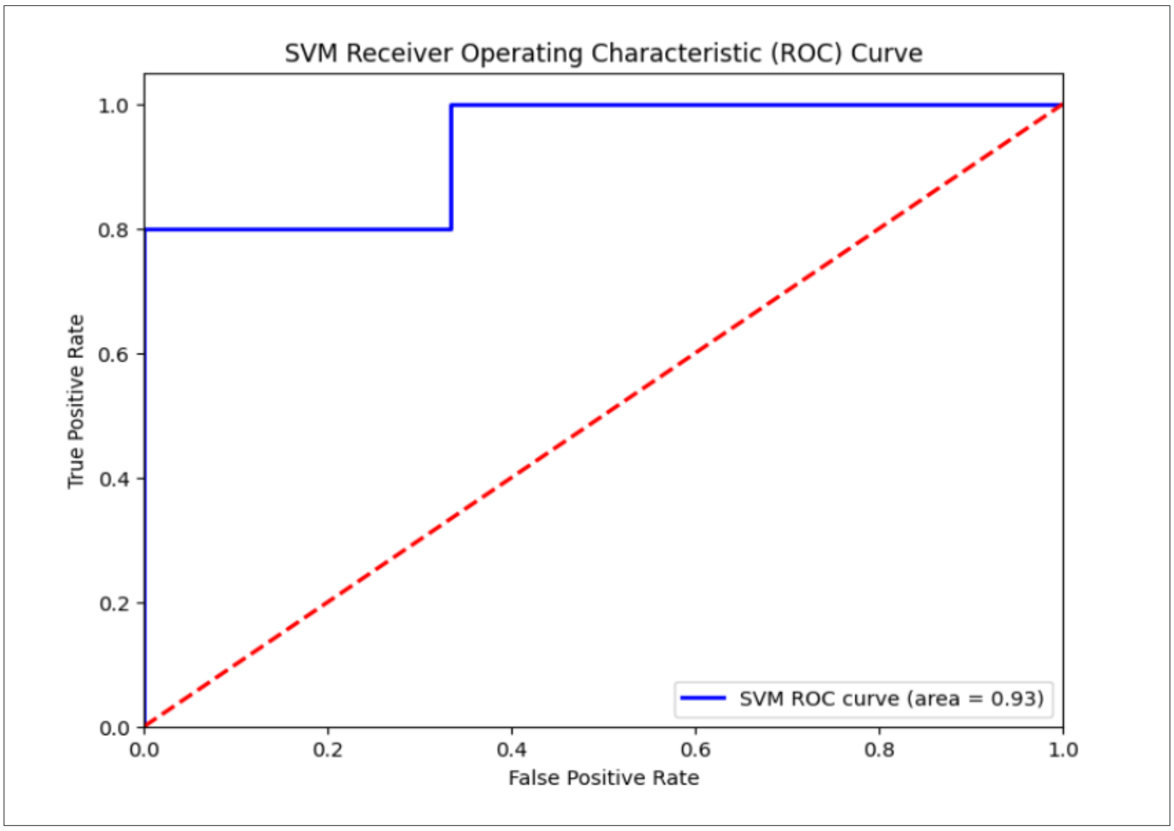

Figure 3: SVM ROC Graph (QA's Dataset)

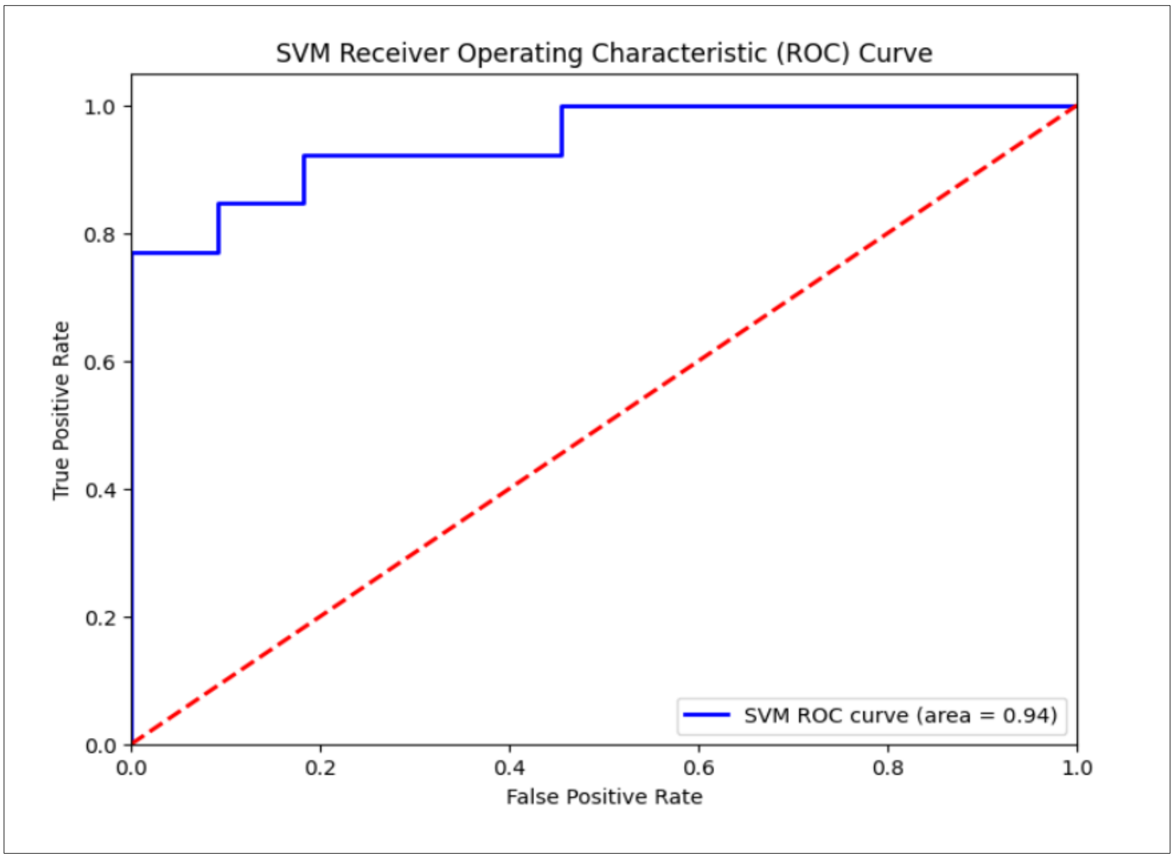

Figure 4: SVM ROC Graph (Developers Dataset)

Table 3: Naïve Bayes (NB) result using the confusion matrix

Roles	**Classes**	**Precision**	**Recall**	**F1-score**	**Support**	**Correctly Predicted**	**Incorrectly Predicted**
Project Managers	0	0.91	0.91	0.91	11	23	2
	1	0.92	0.92	0.92	13		
	Accuracy			0.88	24		
	Mac avg	0.92	0.92	0.92	24		
	Weighted avg	0.92	0.92	0.92	24		
QA’s	0	0.92	0.61	0.73	18	15	8
	1	0.36	0.80	0.50	5		
	Accuracy			0.65	23		
	Mac avg	0.64	0.71	0.62	23		
	Weighted avg	0.80	0.65	0.68	23		
Developers	0	0.83	0.96	0.89	26	37	6
	1	0.92	0.71	0.80	17		
	Accuracy			0.86	43		
	Mac avg	0.88	0.83	0.85	43		
	Weighted avg	0.87	0.86	0.86	43		

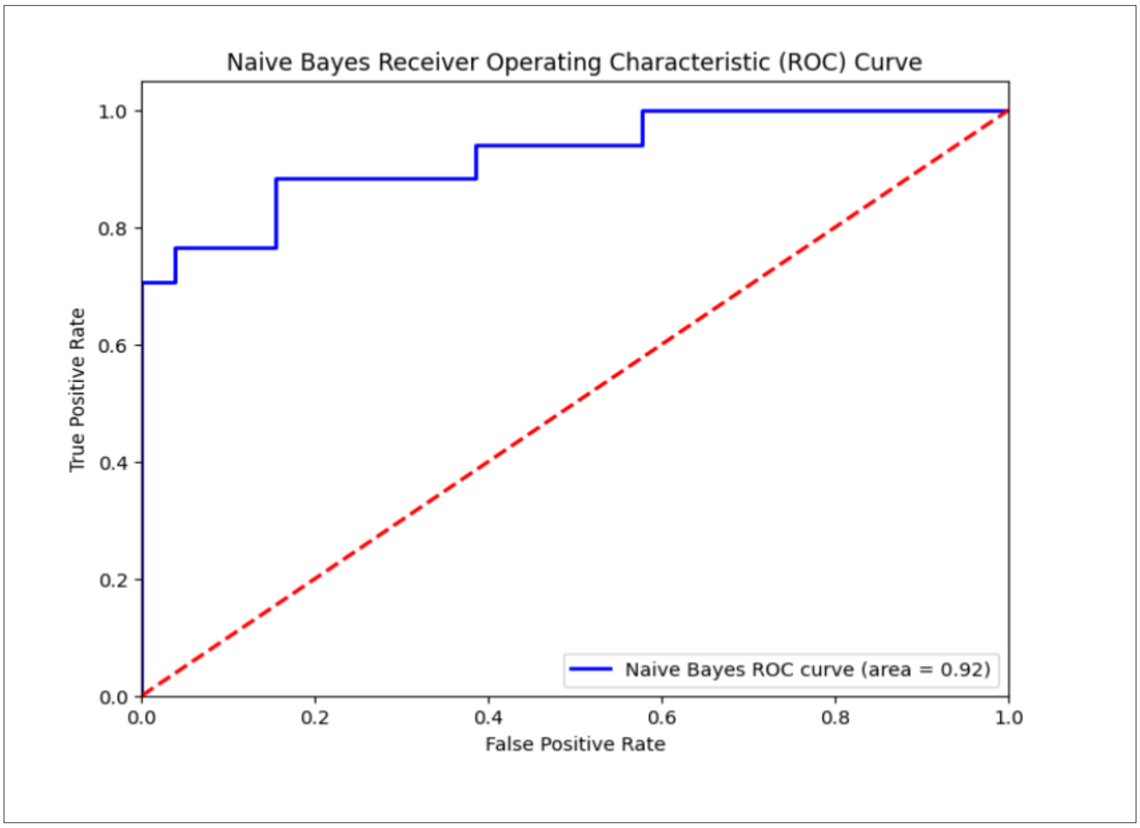

Figure 5: Naïve Bayes ROC Graph (Project Managers Dataset)

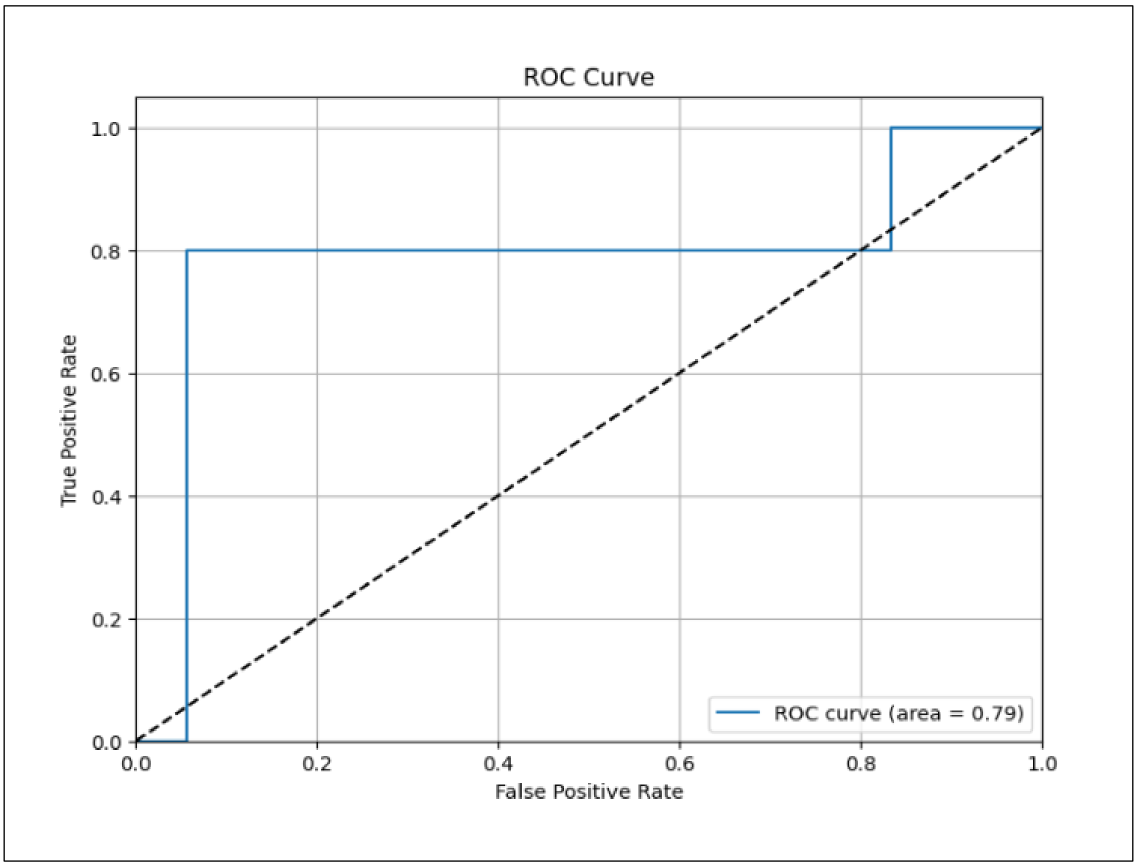

Figure 6: Naïve Bayes ROC Graph (QA's Dataset)

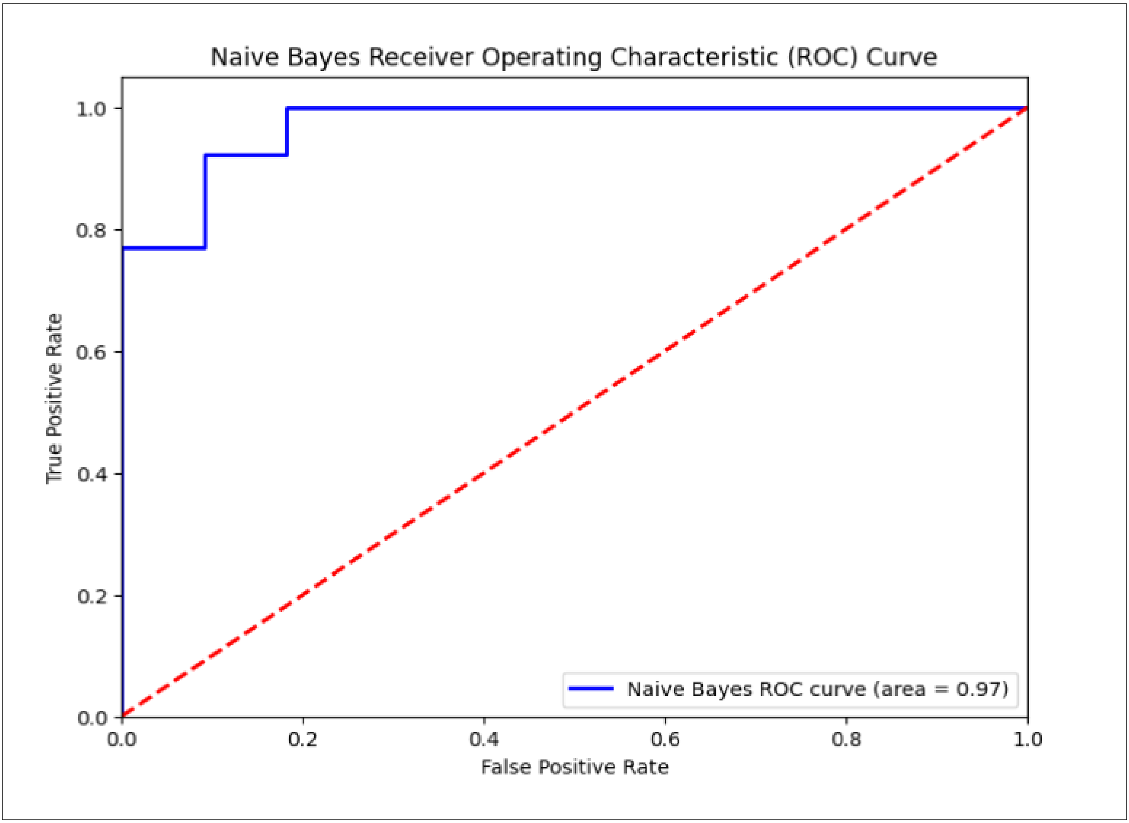

Figure 7: Naïve Bayes ROC Graph (Developers Dataset)

Table 4: K-Nearest Neighbors (KNN) result using the confusion matrix

Roles	Classes	Precision	Recall	F1-score	Support	Correctly Predicted	Incorrectly Predicted
Project Managers	0	1.00	0.91	0.95	11	23	1
	1	0.93	1.00	0.96	13		
	Accuracy			0.96	24		
	Mac avg	0.96	0.95	0.96	24		
	Weighted avg	0.96	0.96	0.96	24		
QA's	0	0.89	0.94	0.92	18	20	3
	1	0.75	0.60	0.67	5		
	Accuracy			0.87	23		
	Mac avg	0.82	0.77	0.79	23		
	Weighted avg	0.86	0.87	0.86	23		
Developers	0	0.95	0.96	0.89	26	34	9
	1	0.67	0.71	0.80	17		
	Accuracy			0.86	43		
	Mac avg	0.81	0.83	0.85	43		
	Weighted avg	0.84	0.86	0.86	43		

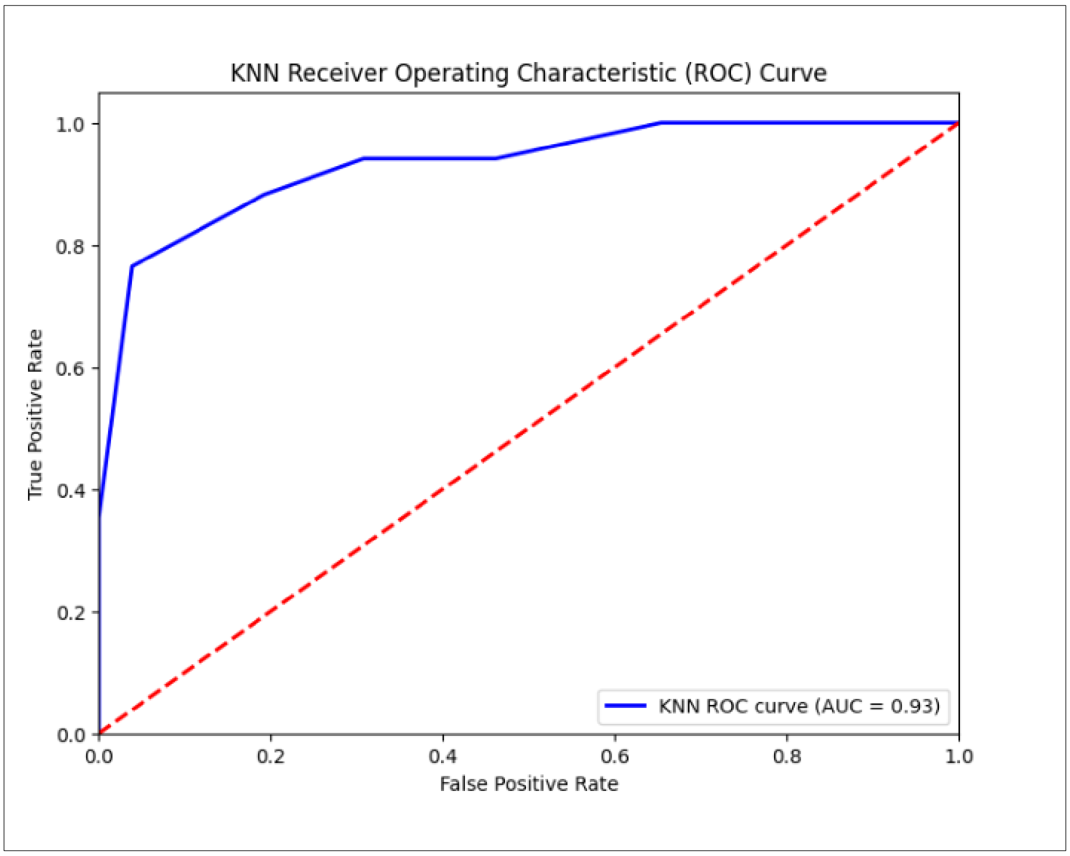

Figure 8: KNN ROC Graph (Project Managers Dataset)

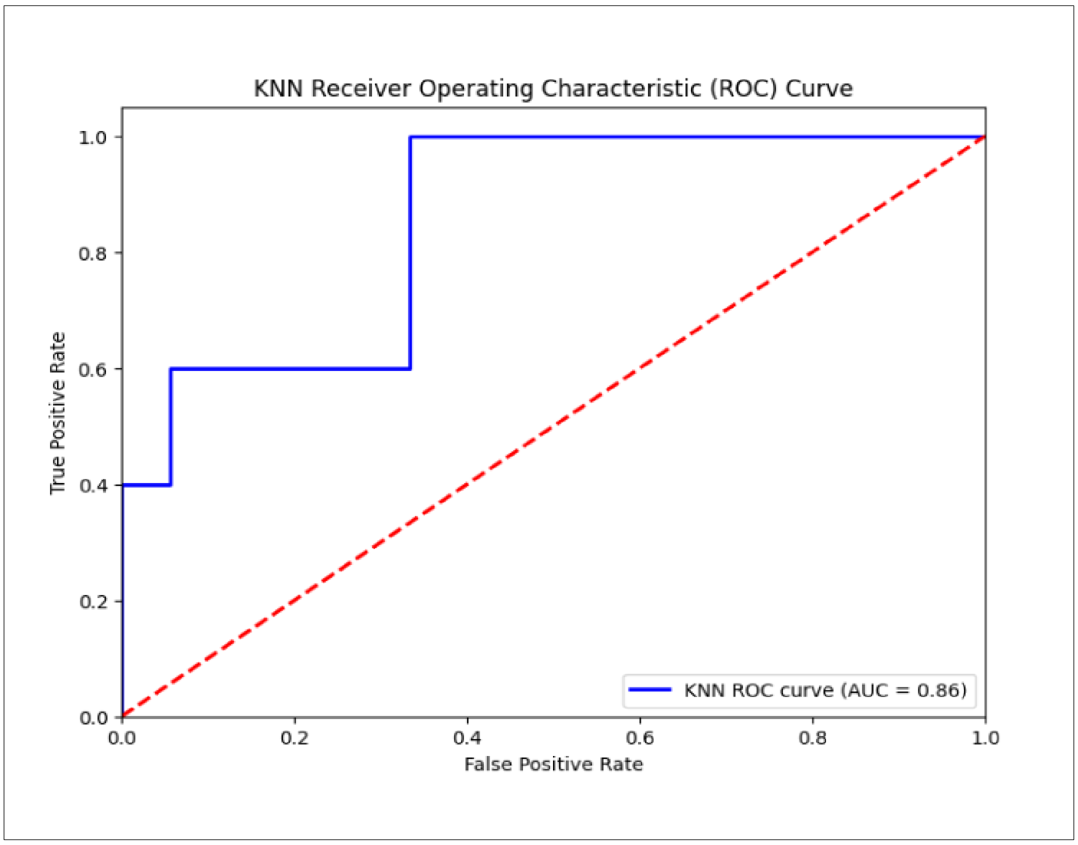

Figure 9: KNN ROC Graph (QA's Dataset)

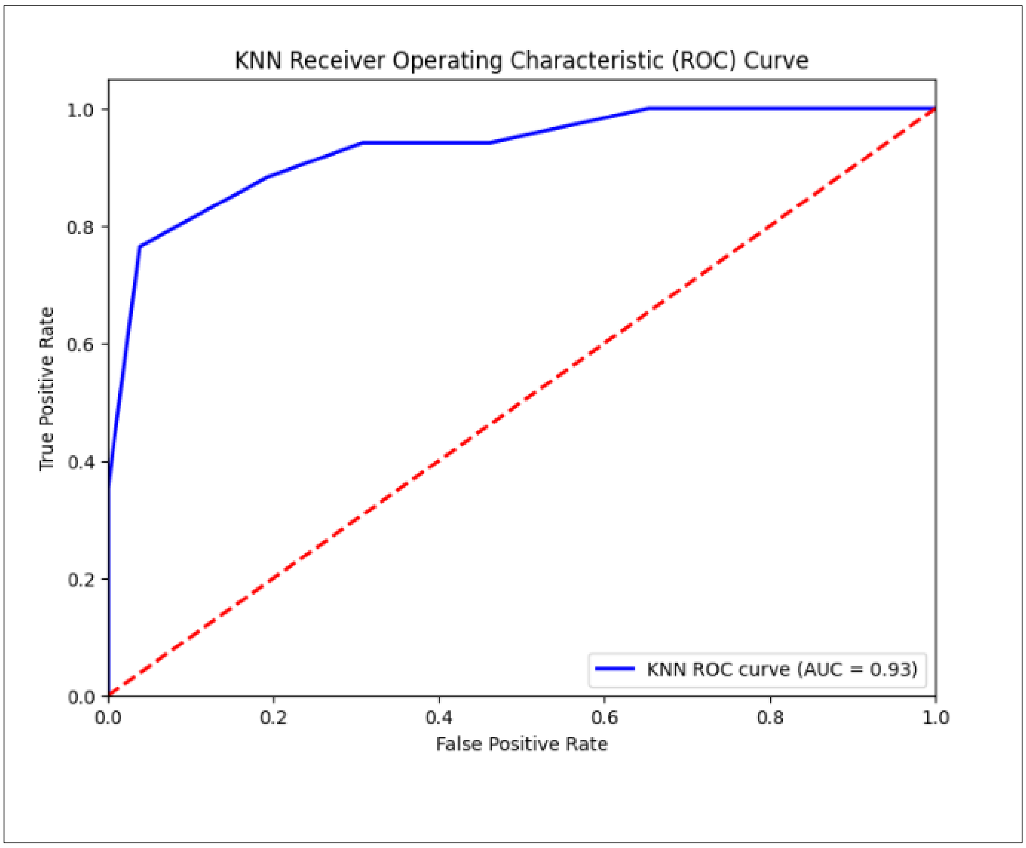

Figure 10: KNN ROC Graph (Developers Dataset)

Table 5: Decision Tree (DT) result using the confusion matrix

Roles	Classes	Precision	Recall	F1-score	Support	Correctly Predicted	Incorrectly Predicted
Project Managers	0	1.00	0.91	0.95	11	23	1
	1	0.93	1.00	0.96	13		
	Accuracy			0.96	24		
	Mac avg	0.96	0.95	0.96	24		
	Weighted avg	0.96	0.96	0.96	24		
QA's	0	1.00	0.61	0.76	18	16	7
	1	0.42	1.00	0.59	5		
	Accuracy			0.70	23		
	Mac avg	0.71	0.81	0.67	23		
	Weighted avg	0.87	0.70	0.72	23		
Developers	0	1.00	0.73	0.84	26	36	7
	1	0.71	1.00	0.83	17		
	Accuracy			0.84	43		
	Mac avg	0.90	0.90	0.84	43		
	Weighted avg	0.91	0.91	0.84	43		

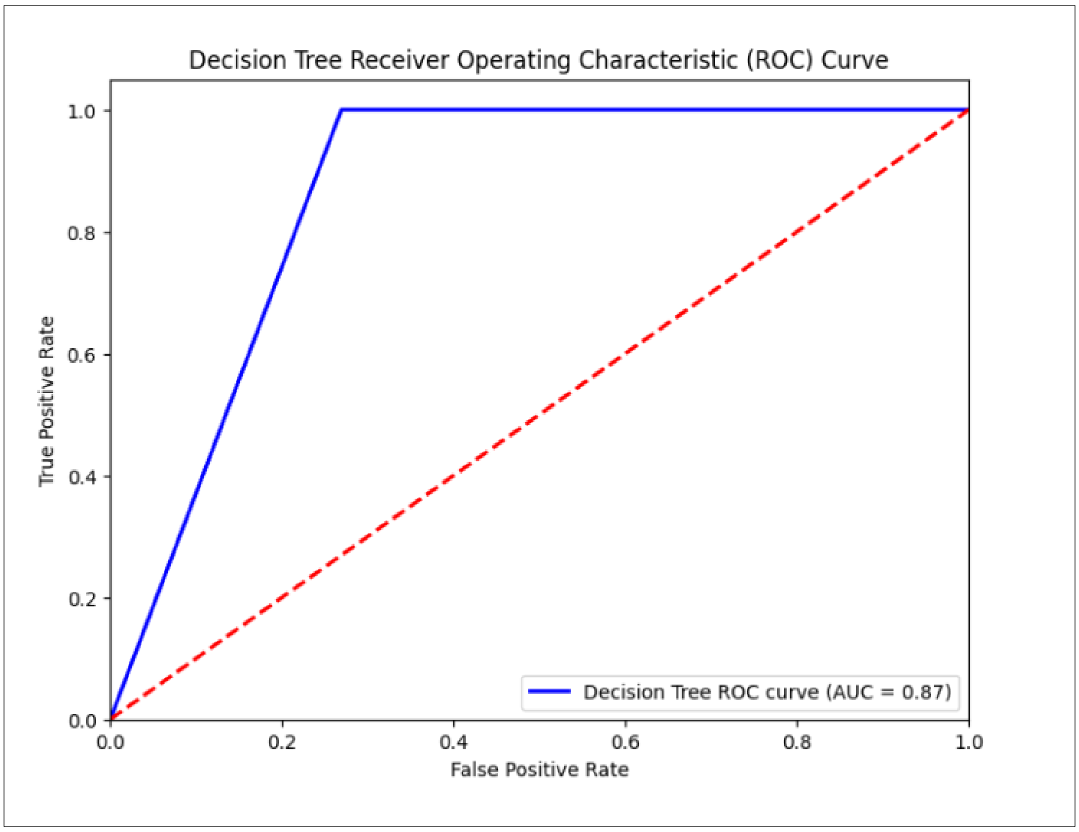

Figure 11: Decision Tree ROC Graph (Project Managers Dataset)

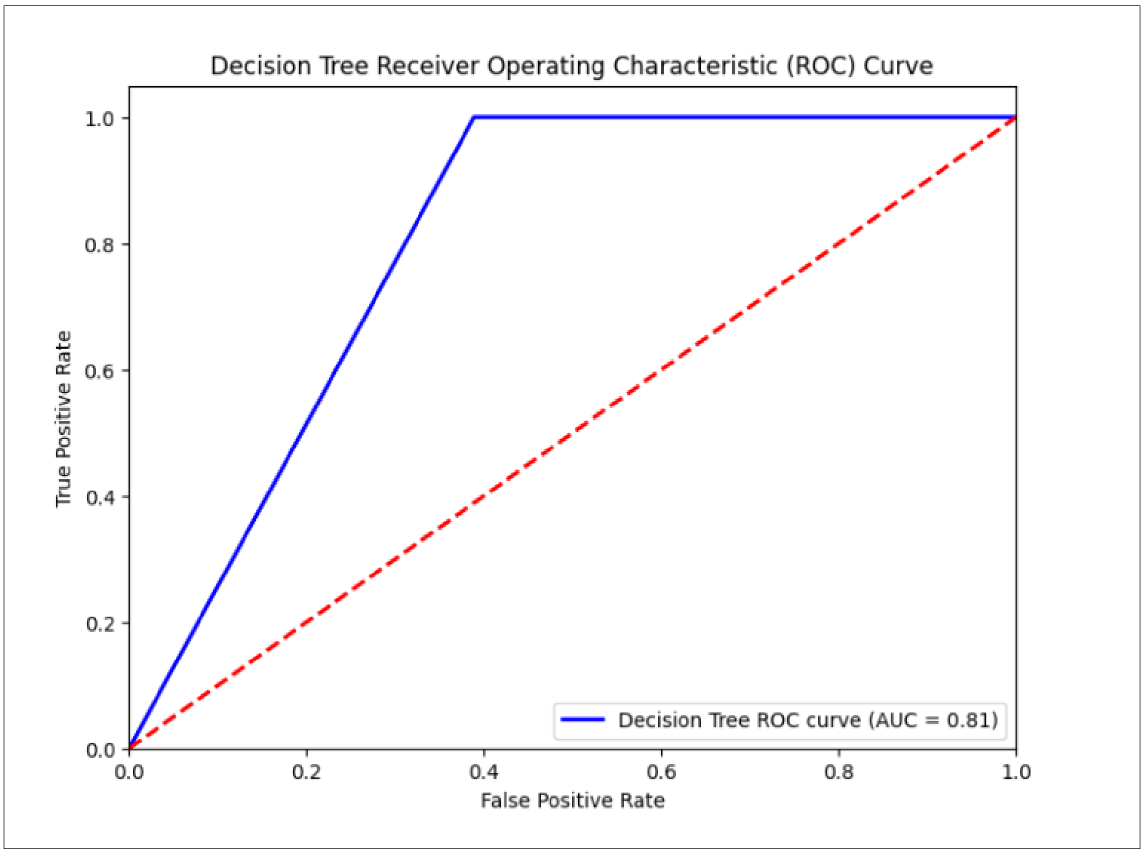

Figure 12: Decision Tree ROC Graph (QA's Dataset)

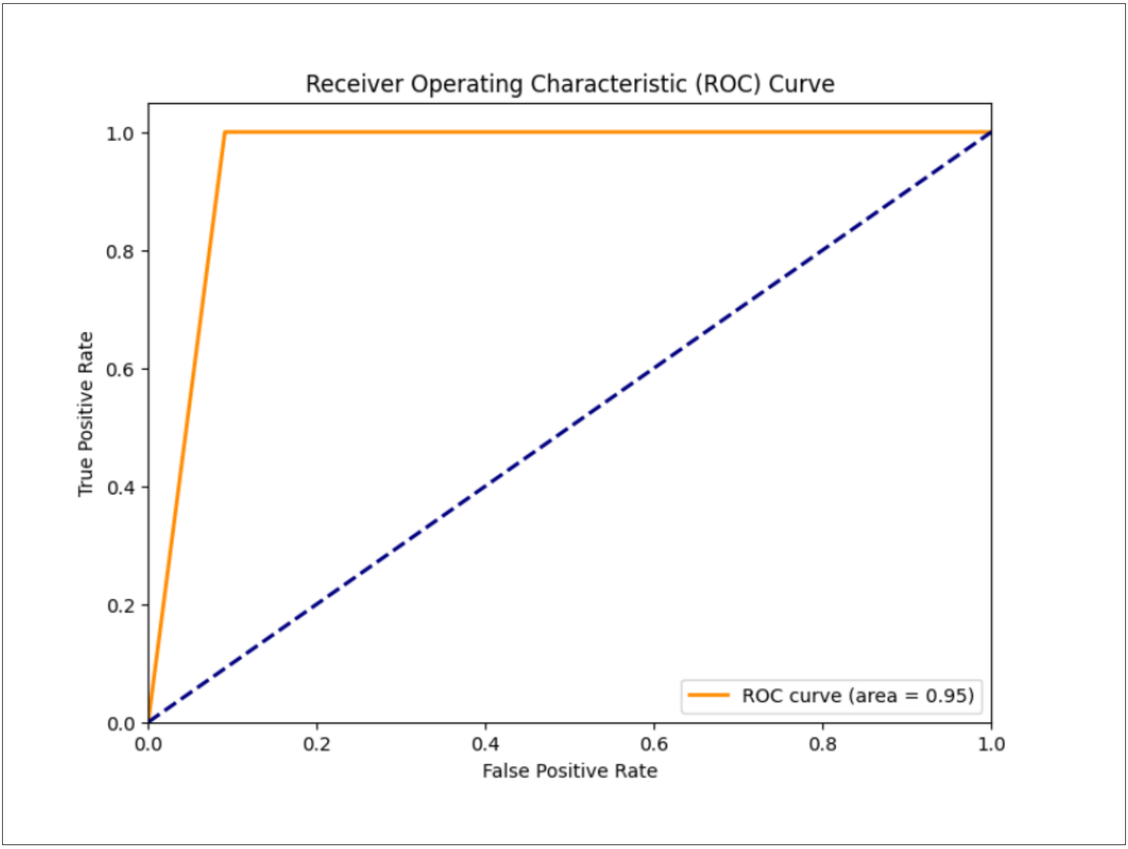

Figure 13: Decision Tree ROC Graph (Developer's Dataset)

Table 6: Logistic Regression (LR) result using the confusion matrix

Roles	Classes	Precision	Recall	F1-score	Support	Correctly Predicted	Incorrectly Predicted
Project Managers	0	0.85	1.00	0.92	11	23	1
	1	1.00	0.85	0.92	13		
	Accuracy			0.92	24		
	Mac avg	0.92	0.92	0.92	24		
	Weighted avg	0.93	0.92	0.92	24		
QA's	0	0.94	0.94	0.94	18	15	8
	1	0.80	0.80	0.80	5		
	Accuracy			0.91	23		
	Mac avg	0.87	0.87	0.87	23		
	Weighted avg	0.91	0.91	0.91	23		
Developers	0	0.81	0.65	0.84	26	34	9
	1	0.51	0.76	0.83	17		
	Accuracy			0.84	43		
	Mac avg	0.70	0.71	0.84	43		
	Weighted avg	0.72	0.70	0.84	43		

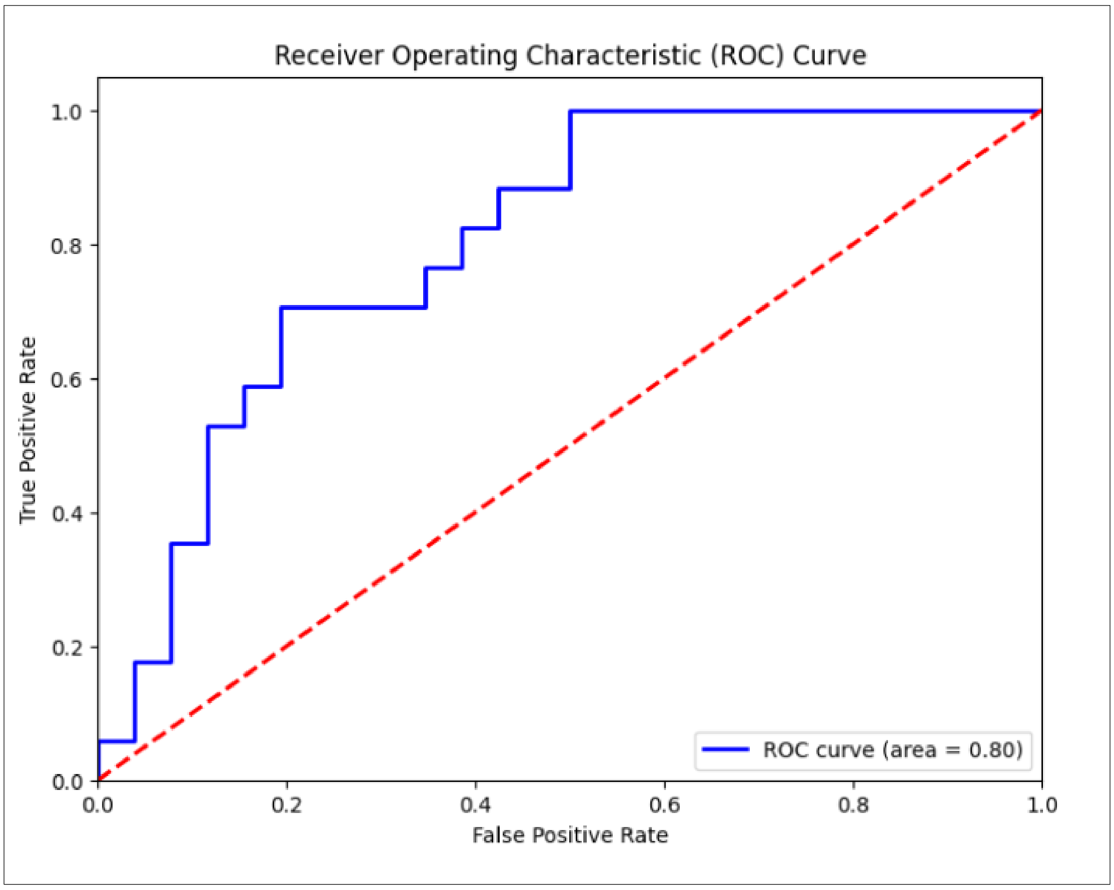

Figure 14: Logistic Regression ROC Graph (Project Managers Dataset)

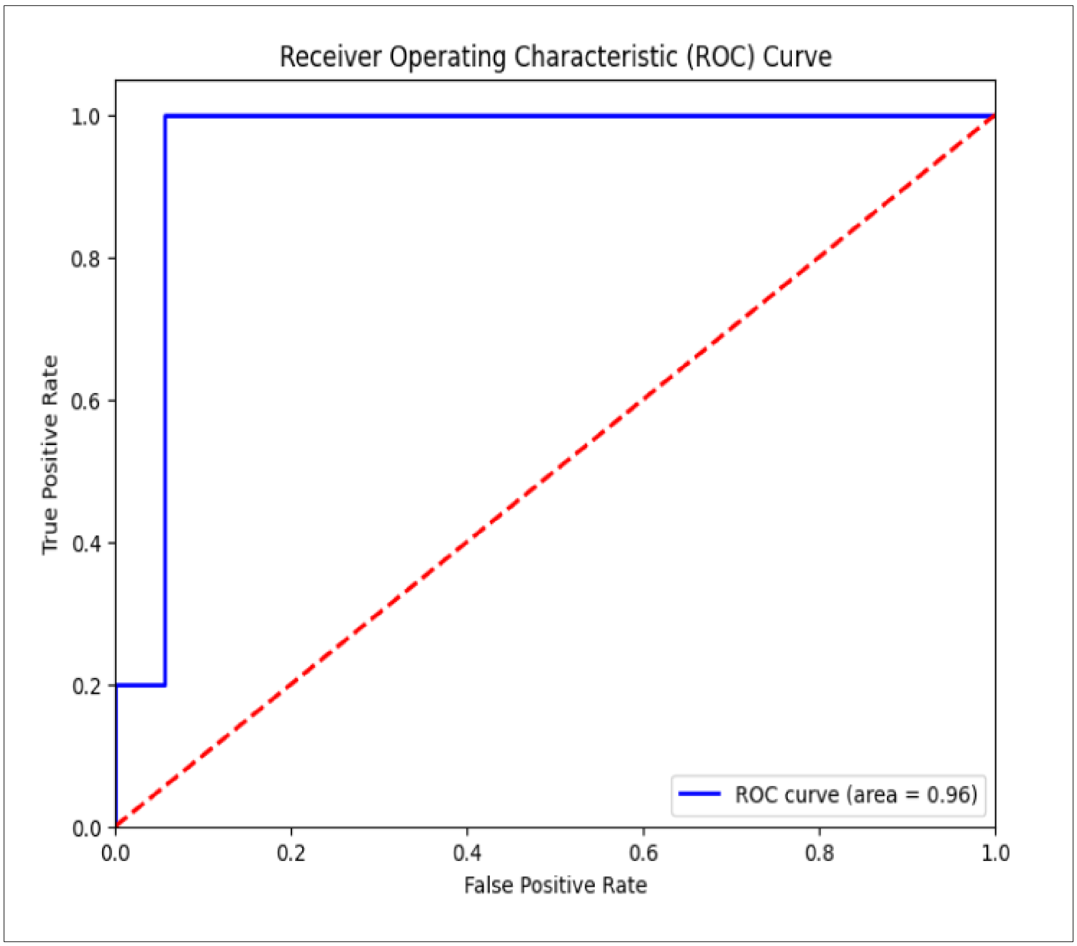

Figure 15: Logistic Regression ROC Graph (QA's Dataset)

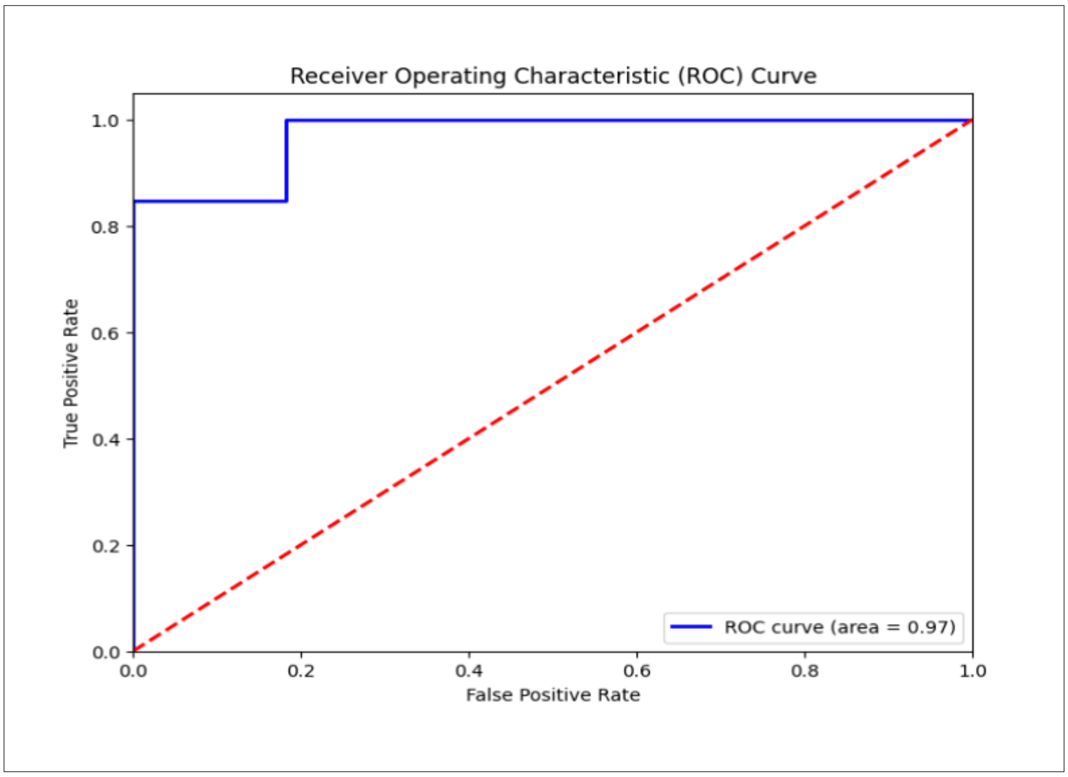

Figure 16: Logistic Regression ROC Graph (Developers Dataset)

3.2 Insights with reasons for algorithm performance

Support Vector Machine (SVM): For Project Managers, SVM achieves precision and recall of 0.79 and 1.00, respectively, due to its ability to handle complex decision boundaries and robustness to outliers. For QA's, SVM delivers precision of 0.95 and recall of 1.00, benefiting from its margin-based classification, particularly for smaller datasets (90 instances). For Developers, SVM balances precision and recall at 0.92, effectively generalizing on moderate-sized datasets (140 instances). Naïve Bayes (NB): For Project Managers, NB shows consistent precision and recall of 0.91 each, leveraging its simplicity and independence assumptions suited for smaller datasets (91 instances). For QA's, NB achieves precision of 0.92 but struggles with recall at 0.61 due to feature dependencies affecting performance. For Developers, NB achieves precision of 0.83 and recall of 0.96, benefiting from larger sample sizes (140 instances). K-Nearest Neighbors (KNN): For Project Managers, KNN achieves high precision (1.00) and recall (0.91), capturing local patterns effectively in smaller datasets (91 instances). For QA's, KNN shows lower precision (0.89), affected by the curse of dimensionality in small datasets (90 instances). For Developers, KNN achieves high precision (0.95) but lower recall (0.69), with larger datasets aiding classification but recall impacted by instance balance. Decision Tree (DT): For Project Managers, DT delivers precision of 1.00 and recall of 0.91 by creating intuitive feature splits for smaller datasets (91 instances). For QA's, DT achieves precision of 1.00 but lower recall (0.61), struggling with overfitting in small datasets (90 instances). For Developers, DT achieves precision of 1.00 and recall of 0.73, effectively generalizing on larger datasets (140 instances). Logistic Regression (LR): For Project Managers, LR shows precision of 0.85 and recall of 1.00, leveraging linear relationships in small datasets (91 instances). For QA's, LR achieves consistent precision and recall of 0.94, finding stable decision boundaries in small datasets (90 instances). For Developers, LR shows precision of 0.81 and recall of 0.65, struggling with class imbalance and non-linear relationships in larger datasets (140 instances). Dataset Size Impact: Algorithms like SVM and Decision Trees excel across roles, handling varying dataset sizes and capturing complex relationships effectively. Algorithm Suitability: Naïve Bayes and KNN show variability, influenced by dataset size and feature dependencies.

3.3 Performance Considerations:

Logistic Regression shows stable performance across roles but may be influenced by dataset specific characteristics like class imbalance and non-linear relationships.

Fig 17 compares the performance of five machine learning models (SVM, Naive Bayes, Decision Tree, KNN, and Logistic Regression) using two metrics: accuracy and AUC (Area Under the Curve). Both bar charts show high performance across all models, with Decision Tree and KNN achieving the highest accuracy, and Logistic Regression, Naive Bayes, and Decision Tree demonstrating the highest AUC. This indicates that while Decision Tree and KNN are slightly better at correctly classifying instances, Logistic Regression, Naive Bayes, and Decision Tree are superior in distinguishing between classes.

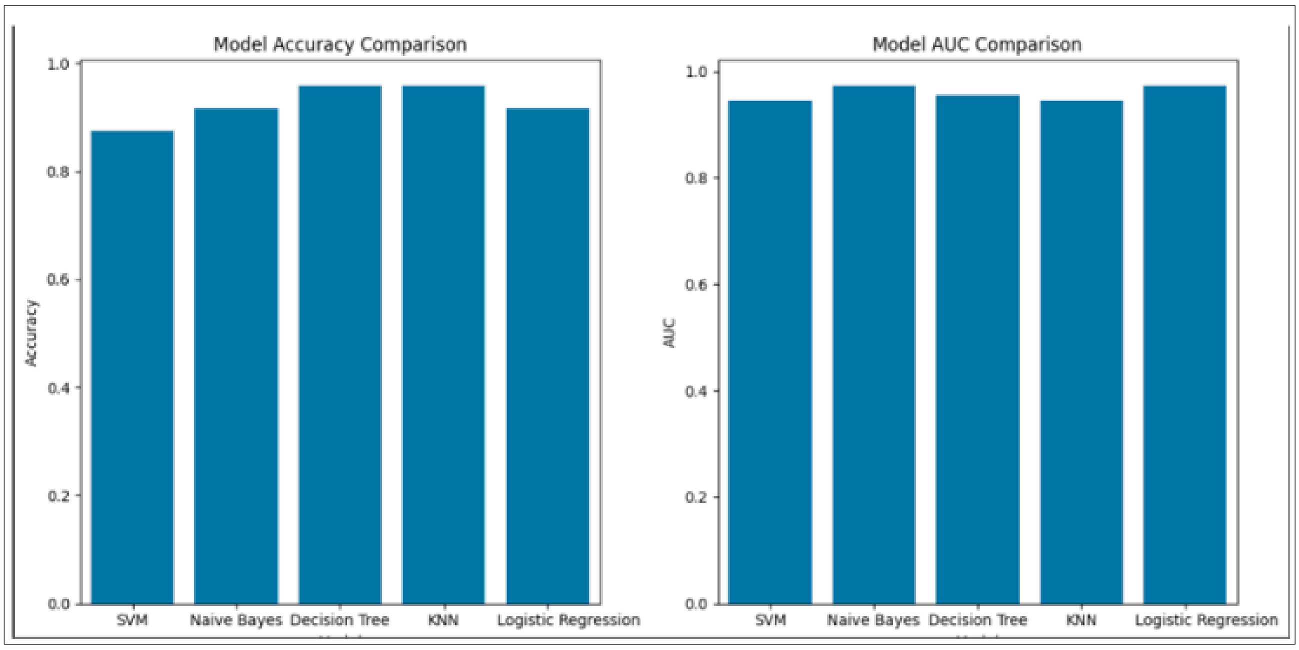

Figure 17: Models Performance Comparison (Project Managers Dataset)

Fig 18 compares the performance of five machine learning models (SVM, Naive Bayes, Decision Tree, KNN, and Logistic Regression) using two metrics: accuracy and AUC (Area Under the Curve). The left bar chart shows that SVM and Logistic Regression achieve the highest accuracy, followed by KNN, with Decision Tree and Naive Bayes having lower accuracy. The right bar chart indicates that Logistic Regression has the highest AUC, followed closely by SVM and KNN, while Decision Tree and Naive Bayes have comparatively lower AUC values. This indicates that SVM and Logistic Regression are superior in both correctly classifying instances and distinguishing between classes, whereas Naive Bayes performs the worst on both metrics.

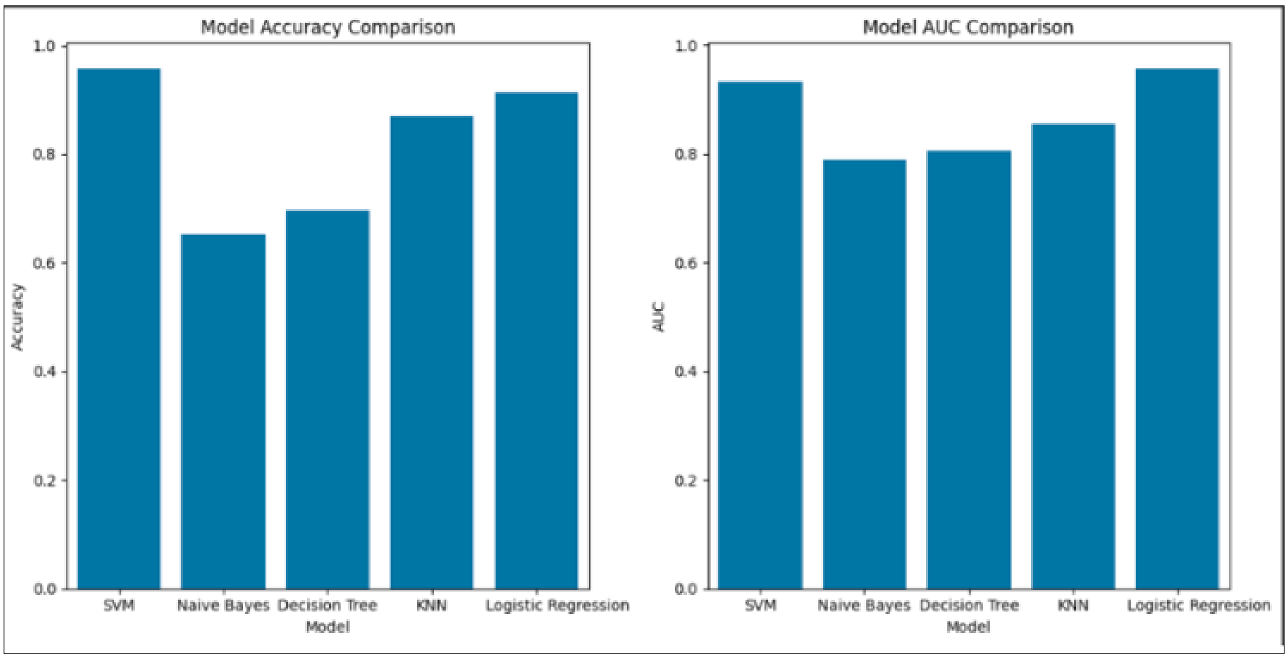

Figure 18: Models Performance Comparison (QA's Dataset)

Fig 19 presents a comparison of different machine learning models based on two performance metrics: accuracy and AUC (Area Under the Curve). The left bar chart shows the accuracy of five models: SVM, Naive Bayes, KNN, Decision Tree, and Logistic Regression. The SVM model has the highest accuracy, followed closely by Naive Bayes, while Logistic Regression has the lowest accuracy. The right bar chart displays the AUC for the same models, indicating a similar trend. SVM, Naive Bayes, and KNN have high AUC values, with SVM leading slightly, whereas Logistic Regression again shows the lowest performance. This comparison highlights that SVM consistently performs the best across both metrics, while Logistic Regression lags.

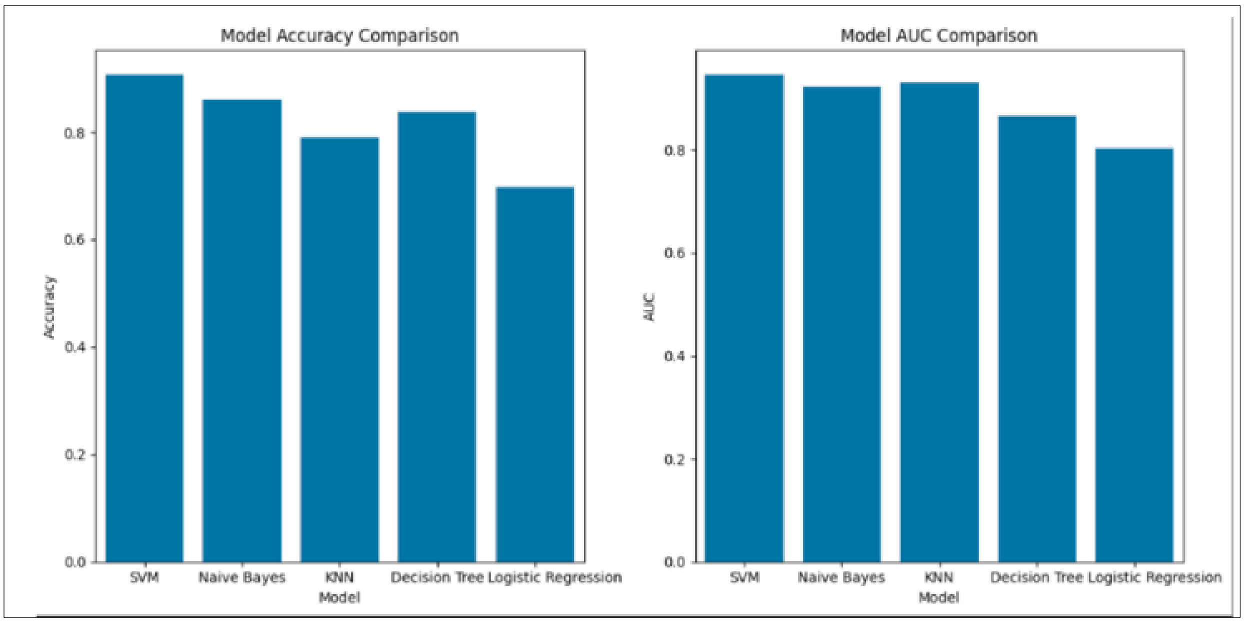

Figure 19: Models Performance Comparison (Developer's Dataset)

4. CONCLUSIONS

The thesis makes a substantial contribution to software project management by addressing the critical issue of proactive risk management. It showcases the potential of predictive analytics to shift from a traditionally reactive approach to a proactive strategy, thereby improving the quality and success rates of software projects. The research begins by clearly identifying the challenges of managing risks in software development projects. A comprehensive literature review establishes a strong foundation by exploring various predictive analytics techniques and their applications. The methodology is rigorous, involving data collection, preprocessing, model implementation, and evaluation, with active participation from developers, project managers, and QA engineers to ensure practical relevance.

The study employs multiple machines learning models, including K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Decision Trees, and Logistic Regression. These models are evaluated using metrics such as accuracy, precision, recall, and F1 score. While the analysis is thorough, a more detailed comparative evaluation could have provided deeper insights into the models' relative effectiveness.

A significant highlight of the research is the development of a Stream lit-based web application that integrates the predictive models for real-time risk prediction and analysis. This practical tool underscores the research's impact and utility, offering project managers a valuable resource for proactive risk management.

5. FUTURE WORKS

In terms of future research, we recommend the following:

- Provide more detailed information on the dataset, including size, diversity, and how representative it is of typical software projects.
- Include a comparative analysis of the evaluation metrics for each model. Discuss their strengths and weaknesses in more detail.
- Explore the interpretability of the models, identifying key features that influence predictions.
- Conduct a detailed evaluation of the real-time application in practical settings. Include user feedback and case studies.

REFERENCES

1. Ammar, M., Haleem, A., Javaid, M., Walia, R., & Bahl, S. (2021). Improving material quality management and manufacturing organizations system through Industry 4.0 technologies. *Materials Today: Proceedings*, *45*, 5089-5096. https://doi.org/10.1016/j.matpr.2021.01.585.
2. Mangla, S. K., Raut, R., Narwane, V. S., Zhang, Z., & Priyadarshinee, P. (2021). Mediating effect of big data analytics on project performance of small and medium enterprises. *Journal of Enterprise Information Management*, *34*(1), 168-198.
3. Jalloh, R. (2023). *Synthesizing a Predictive Analytics Methodology for Quality Management* (Master's thesis).
4. Kontsevoi, B., Kizyan, S., & Dubovik, I. (2023, February). How Predictive Software Engineering Addresses Issues in Custom Software Development and Boosts Efficiency and Productivity. In *International Congress on Information and Communication Technology* (pp. 23-31). Singapore: Springer Nature Singapore.
5. Kontsevoi, B., & Kizyan, S. (2022). Predictive software engineering: transform custom software development into effective business solutions. *J. Econ., Finan. Manage. Stud*, *5*(01), 73-77. https://doi.org/10.47191/jefms/v5-i1-09.
6. Taye, G. D., & Feleke, Y. A. (2022). Prediction of failures in the project management knowledge areas using a machine learning approach for software companies. *SN Applied Sciences*, *4*(6), 165. https://doi.org/10.1007/s42452-022-05051-7.
7. Burggraef, P., Wagner, J., Heinbach, B., Steinberg, F., Schmallenbach, L., Garcke, J., ... & Wolter, M. (2021). Predictive analytics in quality assurance for assembly processes: Lessons learned from a case study at an industry 4.0 demonstration cell. *Procedia CIRP*, *104*, 641-646. https://doi.org/10.1016/j.procir.2021.11.108.
8. Olaleye, T. O., Arogundade, O. T., Misra, S., Abayomi-Alli, A., & Kose, U. (2023). Predictive analytics and software defect severity: A systematic review and future directions. *Scientific Programming*, *2023*(1), 6221388.
9. Thota, M. K., Shajin, F. H., & Rajesh, P. (2020). Survey on software defect prediction techniques. *International Journal of Applied Science and Engineering*, *17*(4), 331-344.
10. Abaei, G., & Selamat, A. (2014). A survey on software fault detection based on different prediction approaches. *Vietnam Journal of Computer Science*, *1*, 79-95. https://doi.org/10.1007/s40595-013-0008-z.
11. Catal, C. (2014). A comparison of semi-supervised classification approaches for software defect prediction. *Journal of Intelligent Systems*, *23*(1), 75-82. https://doi.org/10.1515/jisys-2013-0030.
12. McAdam, R., Miller, K., & McSorley, C. (2019). Towards a contingency theory perspective of quality management in enabling strategic alignment. *International Journal of Production Economics*, *207*, 195-209. https://doi.org/10.1016/j.ijpe.2016.07.003.
13. Sultanow, E., Ullrich, A., Konopik, S., & Vladova, G. (2018, September). Machine learning based static code analysis for software quality assurance. In *2018 Thirteenth International Conference on*

Digital Information Management (ICDIM) (pp. 156-161). IEEE. https://doi.org/10.1109/ICDIM.2018.8847079.

14. Javed, S. A., & Liu, S. (2017, August). Evaluation of project management knowledge areas using grey incidence model and AHP. In *2017 international conference on grey systems and intelligent services (GSIS)* (Vol. 120). IEEE. https://doi.org/10.1109/GSIS.2017.8077684.
15. Oun, T. A., Blackburn, T. D., Olson, B. A., & Blessner, P. (2016). An enterprise-wide knowledge management approach to project management. *Engineering Management Journal*, *28*(3), 179-192. https://doi.org/10.1080/10429247.2016.1203715