UtilitasMathematica

ISSN 0315-3681 Volume 122 (2), 2025

Evaluating Intelligent Methods for Software Risk Prediction: An Empirical Analysis

Mohd Shabbir[1], Rakesh Kumar Yadav[2], Mohd Waris Khan[3*], Hitendra Singh[4]

[1,2]Department of Computer Science & Engineering, Maharishi University of Information Technology, Lucknow,226013, Uttar Pradesh, India, shabbir.aec@gmail.com, Rkymuit@gmail.com

[3*]Department of Computer Application, Integral University, Lucknow, 226026, Uttar Pradesh, India, waris.khan070@gmail.com

[4]Department of Electronics and Communication Engineering, Maharishi University of Information Technology, Lucknow,226013, Uttar Pradesh, India, hit.singh111@gmail.com

***Correspondence:** waris.khan070@gmail.com

Abstract

Software development involves significant uncertainty due to unexpected events occurring at various stages of the software development lifecycle. As software size and complexity increase, the risk of project failures also rises. These unexpected events, known as software risks, stem from various factors throughout the development process. Effective risk management during the early phases is crucial to ensure a high-quality final product. Traditional risk assessments rely on human expertise and past experience, which can be subjective and less reliable. This study employs machine learning methods to predict software risks using historical data, aiming for early and accurate risk detection. Five machine learning models were evaluated alongside multiple feature selection techniques to improve prediction accuracy. Experiments were conducted using publicly available software risk datasets. Results show Support Vector Machine (SVM) model achieved highest classification accuracy of around 80%. Among feature selection methods, Mutual Information demonstrated superior performance across evaluated models, enhancing effectiveness of risk prediction.

Keywords: *Software Risk, Requirement Analysis, Machine Learning, Feature Selection, SVM*

I. INTRODUCTION

The area of software engineering forms a methodical and structured approach to designing, developing, and maintaining software systems. These systems are built and managed using the Software Development Life Cycle (SDLC), a structured process that guides each phase, from initial planning to maintenance. However, throughout the SDLC, unforeseen events can arise, potentially leading to setbacks or failures in the development process [1]. These unpredictable situations are often referred to as software risks, which can stem from ambiguous or incomplete project requirements [2, 3]. Among the various functions within the SDLC, risk prediction stands out as one of the most critical and delicate tasks, requiring precision and accuracy to ensure project success [4]. Effective risk management is extremely critical in the successful execution of software projects [5].

Risk analysis forms the cornerstone of the SDLC and includes identifying potential risks as well as implementation of measures to mitigate them. Given the increasing complexity of modern soft-ware systems, it is imperative to adopt preventive measures in order to avoid project failures [6]. Failure to accurately identify and address risks can lead to project collapse [7]. Consequently, risk assessment should be an ongoing and integral part of the SDLC. Addressing these challenges while the development of software is in an initial phase can significantly reduce effort and cost, contributing to a more efficient and successful project lifecycle [8].

Software development projects are inherently complex and prone to various risks, which, if not managed effectively, can lead to cost overruns, delays, and failures. Risk prediction in software engineering has become a crucial area of research, with intelligent methods playing a significant role in enhancing accuracy and reliability. This paper evaluates the effectiveness of various intelligent techniques for software risk prediction, focusing on empirical analyses to compare their performance.

It is important to note that risks can emerge at any stage of the SDLC, making it essential to ad-dress them proactively [8]. Despite rigorous efforts, a substantial number of these continue to face high levels of risk. The SDLC encompasses various risk-inducing factors, such as budget constraints, timelines, and quality standards, all of which must be carefully managed [9]. Neglecting even a single factor can have far-reaching consequences, potentially derailing the entire development process. Therefore, a robust risk management framework must be capable of identifying risks and assessing their evolution as the project progresses. Without such a framework, critical risks may go unnoticed, jeopardizing the project's outcome [10].

Recent studies have highlighted the advantages of integrating multiple methodologies for im-proved risk assessment. For instance, an approach combining ECSA (Evolutionary Computation and Swarm Algorithms) with ANFIS (Adaptive Neuro-Fuzzy Inference System) demonstrated superior accuracy in identifying critical risk factors using the NASA 93 dataset. Similarly, ensemble AI models have been found to outperform single AI models in predicting buggy Java classes, particularly in open-source Apache Commons Projects. These findings suggest that models of risk assessment can be improved in terms of their power of prediction by leveraging ensemble techniques.

Artificial Neural Networks (ANNs) have also shown promising results in assessing software project risks, particularly in challenging economic environments such as Pakistan. With accuracy rates reaching up to 99.12% in total risk prediction, ANNs provide project managers with valuable in-sights for proactive risk mitigation. Additionally, decision tree-based models like Random Forest have proven effective in classifying software requirement risks, offering project teams a systematic approach to prioritizing risk factors.

Machine learning techniques continue to be extensively evaluated for their role in software risk prediction [11]. Empirical studies suggest that models such as Decision Trees and Random Forest perform well in classifying risk levels, aiding project managers in strategic decision-making. Furthermore, chemo metric approaches like PLS with Discriminant Analysis have demonstrated efficiency in predicting Java classes that are prone to bugs and have static source code metrics as their base, facilitating better resource allocation in software maintenance.

This paper aims to build on these findings by comparing and analyzing diverse intelligent methods used for software risk prediction. Through empirical evaluations, it seeks to determine the most effective techniques in identifying and mitigating risks in software projects. The findings of the current study will provide meaningful insights for different stakeholders like practitioners and researchers, ultimately contributing in improved software reliability and project success.

A. Major Contribution of this Research Work

- Better prediction of software risks through models of machine learning and these can be termed as state-of-the-art for their superior performance
- The study revealed the selection of key features, significantly enhances the predictive performance of intelligent risk assessment models.
- Implementing intelligent risk prediction methods during the initial phases of the SDLC proved effective in identifying potential risks early, thereby reducing likelihood of project delays and failures.
- AI-driven risk prediction models showcased scalability and adaptability across diverse software projects, making them suitable for small as well as large scale development environments.

Hereafter, the paper follows the following structure: The following section provides a brief review of existing literature on the subject, elaborating methodologies utilized and their corresponding explanations. Section Number 3 delves into the classification models, offering a comprehensive description of their design and functionality. Section 4 details the experimental framework and configurations employed in this research. Section 5, the most critical part of the study, presents the results and discussion, accompanied by an analytical interpretation of the empirical findings. Lastly, Section 6 concludes the paper and offers actionable recommendations derived from the insights gathered throughout the research.

II. RELATED WORK

This section summarizes substantial contributions, focusing on their methods, effectiveness, and constraints within empirical software risk evaluation. Table 1 has discussed the various intelligent methods, including machine learning and statistical models, to enhance accuracy and decision-making.

TABLE I. SUMMARY ON SOFTWARE RISK PREDICTION TECHNIQUES

Author(s)	Title	Techniques Used	Description
S.R. Bauskar, et al, 2024 [12]	Predictive Analytics for Project Risk Management Using Machine Learning	Predictive analytics, risk management, decision-making, data-driven strategies, machine learning	The paper proposes a machine learning-driven strategy for managing project risks in real time by analyzing historical data from IT projects. It leverages predictive analytics, particularly the Gradient Boosting Machine (GBM) algorithm, along with t-SNE for dimensionality reduction, to enhance risk prediction accuracy. The approach demonstrated notable improvements, achieving 85% accuracy in risk prediction, 85% efficiency in resource utilization, and a 10% reduction in project costs—outperforming conventional methods. The study underscores the critical role of model selection and introduces a data-centric framework to support informed

			decision-making and reduce the likelihood of project failure.
S. Kalogiannidis, et al, 2024 [13]	The Role of Artificial Intelligence Technology in Predictive Risk Assessment for Business Continuity: A Case Study of Greece	AI-based predictive maintenance, response planning, regression analysis, quantitative survey	The study investigates how artificial intelligence (AI) technologies enhance predictive risk assessment and business continuity. Focusing on Greek industries, it analyzes the impact of AI components like NLP, predictive maintenance, and data analytics through responses from 30 technology professionals. The findings show that AI significantly improves risk identification, reduces operational downtime, and strengthens incident response strategies. Regression analysis confirmed that AI integration—especially in data analytics and incident response planning—strongly contributes to sustaining business operations during disruptions.
Yasiel Pérez Vera, et al, 2024 [14]	Comparing Machine Learning Techniques for Software Requirements Risk Prediction	Techniques used: Logistic Regression, MLP Neural Network, SVM, Decision Tree, Naive Bayes, Random Forest	The study explores predicting risk levels while assessing software requirements applying different techniques of machine learning. Through cross-validation and statistical analysis, the most effective models in classifying risks were found to be Decision Tree and Random Forest thus providing valuable insights for project managers in prioritizing risk mitigation efforts.
M.N. Alatawi, et al, 2023 [15]	A Data-Driven Artificial Neural Network Approach to Software Project Risk Assessment	Artificial Neural Networks (ANNs)	This research evaluates the efficiency of ANNs in assessing software project risks, particularly in Pakistan's challenging economic environment. In this paper, the author employs the historical data for predicting individual and overall project risk factors. Finally, the outcome of this study shows that ANNs accomplish up to 99.12% accuracy in overall risk prediction, supporting project managers in implementing proactive risk mitigation.
Yang-ha Chun et al, 2022 [16]	An Empirical Study of Intelligent Security Analysis Methods Utilizing Big Data	Behavior-based analysis, Big data analytics, Real-time monitoring Application and user monitoring	In this paper, the authors present a big data-driven, behavior-based approach to intelligently analyze system logs, network traffic, and activity patterns for early detection of advanced cyber threats like APTs. By integrating multi-source data and advanced analytics, it overcomes limitations of signature-based methods and accentuates the need for real-time, intelligent security frameworks to enrich organizational defense.
Rashid Naseem, et al, 2021 [17]	Empirical Assessment of Machine Learning Techniques for Software Requirements Risk Prediction	Machine Learning Techniques	This study explores predicting risk levels while assessing software requirements applying different techniques of machine learning. The authors conducted empirical evaluations for evaluating these techniques, providing insights into their applicability in real-world scenarios.
Rudolf Ferenc, et al, 2020 [18]	An automatically created novel bug data set and its validation in	Static Code Analysis, Code Metrics, Machine Learning	This paper introduced a novel dataset of bugs capturing before-and-after fix states using GitHub commits. Validated

	bug prediction		it through machine learning-based bug prediction.

Table 1 presents a relative overview of recent studies on software risk prediction using advanced machine learning and AI techniques. Like Gradient Boosting, ANNs, SVMs, Decision Trees, and big data analytics are applied to project risk management, requirement assessment, and security analysis. Results show enhanced prediction accuracy, resource use, and cost-efficiency, with AI-based approaches outperforming traditional methods.

III. Classification Models

A. *Naive Bayes (NB)*

Naive Bayes classifier, based on Bayes' Theorem, is widely used for text classification tasks like spam filtering and sentiment analysis by calculating class probabilities under assumption of feature independence [19]. Despite this strong assumption, it performs well with count-based features such as word frequencies, offering efficiency, scalability, and interpretability, though it may struggle with highly correlated or complex data.

B. *K-Nearest Neighbors (KNN)*

K-Nearest Neighbors (KNN) is a simple yet powerful machine learning algorithm used for classification and regression by identifying the 'k' closest data points based on a distance metric like Euclidean distance. Being non-parametric, it makes no assumptions about data distribution, offering flexibility but can be computationally expensive on large or high-dimensional datasets, especially if data is noisy [19].

C. *Support Vector Machine (SVM)*

Support Vector Machine (SVM) is a powerful supervised learning algorithm used for both regression and classification, which finds an optimal hyperplane to maximize the margin between classes in high-dimensional spaces [19]. It handles linear and non-linear data through kernel functions like radial basis and polynomial but can be sensitive to parameter tuning and computationally intensive [20].

D. *Logistic Regression*

Logistic regression is a widely used statistical model for binary and multi-class classification that maps input features through a sigmoid function to output probabilities between 0 and 1. It is simple, efficient, and interpretable, often used as a baseline, but its assumption of a linear relationship between features and log-odds can limit performance on complex, non-linear data [19].

E. *Multi-Layer Perceptron (MLP)*

Multi-Layer Perceptron (MLP) is a type of artificial neural network with multiple interconnected layers that uses back propagation and gradient descent to learn complex linear and nonlinear patterns [19]. It is versatile for classification and regression tasks, especially with large structured data, but requires careful tuning of hyper parameters like learning rate and hidden layers.

IV. Experimental Setup

A. *Dataset Description*

A publicly available dataset is used in this study [21]. It consists of 299 instances and 13 columns, containing both categorical and numerical features. The columns include "Requirements," "Project Category," "Requirement Category," and "Risk Target Category," all these being categorical in nature. Numerical columns such as "Probability," "Affecting No of Modules," "Fixing Duration (Days)," and "Priority" provide quantitative data relevant to the project's risk assessment. Other categorical columns include "Magnitude of Risk," "Impact," and "Dimension of Risk". The final column "Risk Level" corresponds to class and has values from 1 to 5. This dataset is designed to help analyze and classify project risks based on various factors, making it ideal for machine learning experiments.

B. *Data Preprocessing*

The preprocessing of this dataset involves several key steps to enhance data reliability and prepare it for effective machine learning. First, column names are stripped of leading and trailing whitespace to maintain consistency. The target variable, 'Risk Level,' is converted to an integer type for proper model training. Missing values are imputed using the median, which helps reduce the influence of outliers.

For text data in the 'Requirements' column, tokenization, lowercasing, stop word removal, and lemmatization are performed to normalize the text, followed by dropping the original column to avoid redundancy. Feature representation includes ordinal encoding for ordered variables like 'Magnitude of Risk' and 'Impact,' and one-hot encoding for categorical variables such as 'Project Category' and 'Requirement Category,' allowing better interpretation by the model. The cleaned text in the 'Requirements Cleaned' column is transformed into numerical features using Bag of Words via Count Vectorizer, based on word frequency. Numerical features like 'Probability' and 'Fixing Duration (Days)' are scaled between 0 and 1 using Min-Max Scaling, ensuring balanced feature influence [22, 23]. Finally, SMOTE is applied to the training data to synthetically oversample the minority class, addressing class imbalance and preventing model bias toward the majority class.

C. *Feature Selection*

Feature selection identifies the most important features in a dataset, enhancing model performance and reducing

complexity. ANOVA (Analysis of Variance) is used when the target is categorical and features are numerical, selecting features with significant group mean differences, indicated by higher F-statistic values. The Chi-Square test applies to categorical features, assessing the relationship between features and the target by comparing observed and expected frequencies; higher Chi-square values show stronger associations. Mutual Information measures how much knowledge of one variable reduces uncertainty in another, capturing both linear and non-linear relationships; features with higher mutual information are more relevant for prediction. Recursive Feature Elimination (RFE) iteratively trains the model, removing the least important feature each time to find the smallest subset of features that yields the best model performance.

D. Evaluation Metrics

We have used four metrics of valuation based on the parameters of confusion matrix to assess the performance of different classification models.

Accuracy: It measures the proportion of correctly classified instances in the dataset but can be misleading in imbalanced datasets, as it doesn't reflect performance on minority classes accurately. It is computed by the formula:

$$Accuracy = (TP + TN) / (TP + FP + FN + TN)$$

Precision: It measures the accuracy of positive predictions, indicating how many predicted positives are correct; it's crucial when false positives have serious consequences. The formula is:

$$Precision = True\ Positives\ (TP) / (True\ Positives\ (TP) + False\ Positives\ (FP))$$

Recall (Sensitivity or True Positive Rate): Recall focuses on minimizing false negatives, meaning the number of real positive instances which the model correctly predicted. It describes the ability of the model to recollect cases that are relevant within the dataset. The formula is:

$$Recall = True\ Positives\ (TP) / (True\ Positives\ (TP) + False\ Negatives\ (FN))$$

F1 Score: It is the harmonic mean of precision and recall, balancing false positives and negatives, especially useful when a trade-off between them is needed. It is determined using the formula:

$$F1\ Score = (2 \times Precision \times Recall) / (Precision + Recall)$$

E. Parameter Settings

The default parameters were used for all the classification models and no fine tuning was performed.

V. Results and Discussion

The experiments were performed using the Python libraries like NumPy, pandas, scikit-learn, imbalanced-learn, NLTK, Matplotlib, Seaborn, and SciPy. The research is conducted in Python on Google Colab, utilizing 13.61 GB of RAM and 2 CPU cores, providing sufficient resources for efficient model training and evaluation. After preprocessing, the number of features in the dataset are 714 due to the application of BoW representation on textual column.

Table 2 shows the performance of all the five classifiers when no feature selection was applied on the dataset. It is evident the classification accuracy of SVM model is the highest while KNN is the worst performer among all. SVM performance may be attributed to its versatility in dealing with high dimensional datasets. In terms of other evaluation metrics, more or less the same performance was produced by all the classification models. F-Score can be used to determine the performance of the model. A good F-score is an indication of better performance, as can be seen in Table 2, SVM and LR obtained much better F-score as compared to other classification models. Figure 1 presents a comparative analysis of different machine learning models' performance when trained using the complete set of features from the dataset, without applying any feature selection. Only two levels of headings should be numbered. Lower level headings remain unnumbered; they are formatted as run-in headings.

TABLE II. Model Performance for all features

Classifier	NB	LR	SVM	MLP	KNN
Accuracy	38.33	51.67	53.33	41.67	26.67
Precision	52.58	57.05	52.79	47.88	65.89
Recall	38.33	51.67	53.33	41.67	26.67
F-Score	40.57	52.04	52.83	42.79	25.03

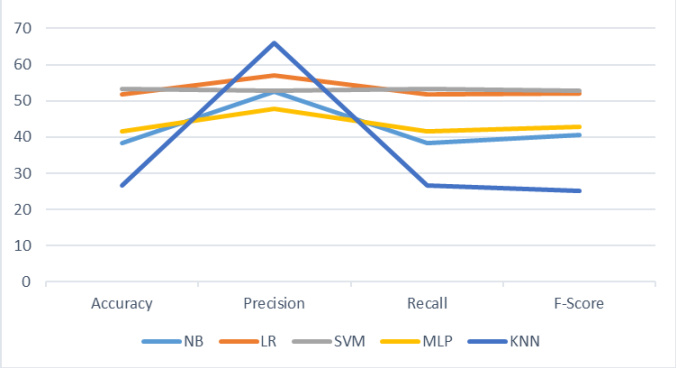

Fig. 1. Model performance using all features

Table 3 and Figure 2 present the performance of all models after applying ANOVA feature selection, demonstrating how retaining only statistically significant features influences accuracy, precision, recall, and F1-score. While the table provides detailed numerical comparisons, the figure visually highlights performance changes and improvements in key metrics achieved by reducing irrelevant features.

TABLE III. IMPACT OF ANOVA FEATURE SELECTION ON MODEL PERFORMANCE

Classifier	NB	LR	SVM	MLP	KNN
Accuracy	28.33	58.33	65.00	66.67	43.33
Precision	50.46	66.63	72.57	71.95	51.15
Recall	28.33	58.33	65.00	66.67	43.33
F-Score	30.41	58.46	65.43	66.64	46.12

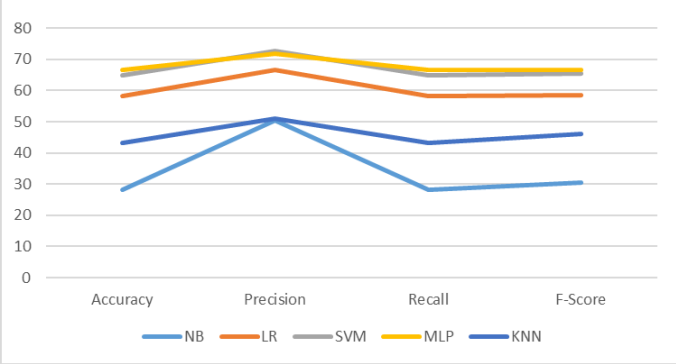

Fig. 2. Model Performance after ANOVA Feature Selection

Table 4 provides detailed numerical results, while the figure 3 visually highlights performance changes and improvements in key metrics achieved through the elimination of less relevant features.

TABLE IV. IMPACT OF CHI-SQUARE FEATURE SELECTION ON MODEL PERFORMANCE

Classifier	NB	LR	SVM	MLP	KNN
Accuracy	26.67	61.67	60.00	65.00	51.67
Precision	43.29	65.43	68.35	68.25	55.60
Recall	26.67	61.67	60.00	65.00	51.67
F-Score	28.24	62.42	60.69	65.63	53.12

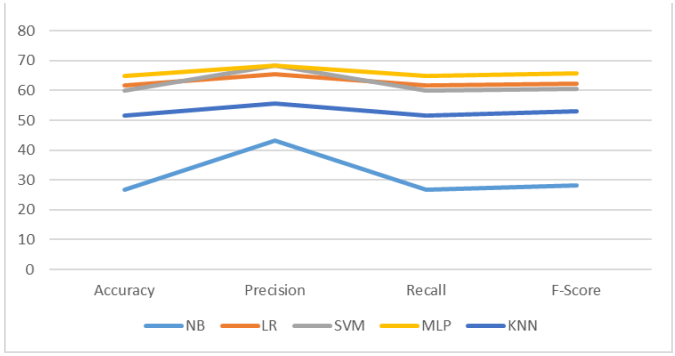

Fig. 3. Model Performance after Chi-Square Feature Selection

Table 5 and Figure 4 show the performance of all models after applying Mutual Information-based feature selection, where features with the highest relevance to the target variable are retained. The table presents detailed metrics, while the figure visually highlights performance improvements achieved by removing less informative features.

TABLE V. IMPACT OF MUTUAL INFORMATION BASED FEATURE SELECTION ON MODEL PERFORMANCE

Classifier	NB	LR	SVM	MLP	KNN
Accuracy	28.33	75.00	80.00	75.00	70.00
Precision	65.60	83.32	84.04	80.99	83.02
Recall	28.33	75.00	80.00	75.00	70.00
F-Score	28.84	75.17	80.91	76.00	72.68

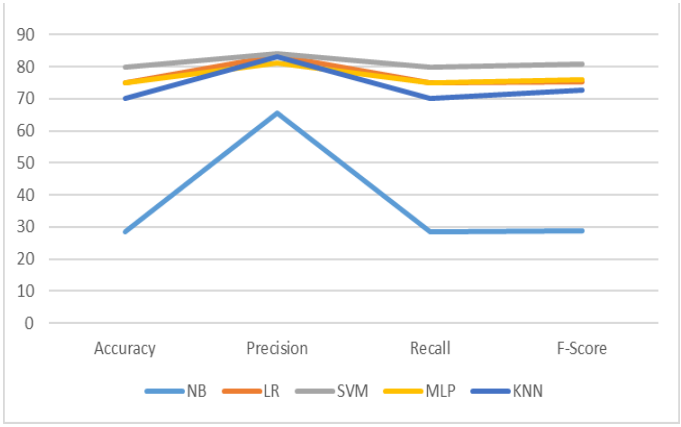

Fig. 4. Model Performance after Mutual Information-Based Feature Selection

Table 6 and Figure 5 present the performance of all models after applying Recursive Feature Elimination (RFE), which iteratively removes less important features to retain the most significant ones. The table provides detailed

performance metrics, while the figure visually highlights improvements resulting from this optimized feature subset.

TABLE VI. IMPACT OF RECURSIVE FEATURE ELIMINATION (RFE) BASED FEATURE SELECTION ON MODEL PERFORMANCE

Classifier	NB	LR	SVM	MLP	KNN
Accuracy	26.67	66.67	63.33	70.00	50.00
Precision	49.96	73.95	66.37	73.52	58.50
Recall	26.67	66.67	63.33	70.00	50.00
F-Score	26.11	67.29	63.90	70.83	51.96

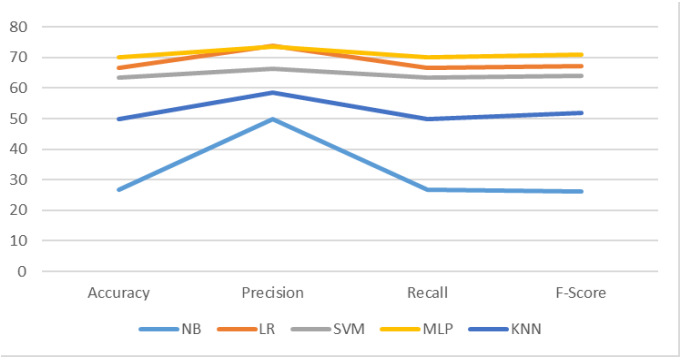

Fig. 5. Model Performance after RFE-Based Feature Selection

Now, we shall discuss the performance of the models when different filter-based feature selection techniques are applied. We have used five feature selection techniques and train the models on different number of features starting from 1 up to 714 incrementing with interval of 25. The best performance of the models was obtained when 25 features were selected by all the feature selection techniques. Fig. 1 shows the performance of different feature selection techniques when 25 best features were selected from 714 for all the classification models. It can be seen that the Mutual Information (MI) feature selection technique gave the best results in terms of all evaluation metrics over all the models. MI has been shown to provide a good selection of features in previous studies also. Additionally, MI is also computationally inexpensive as compared to other technique like Recursive Feature Elimination (RFE) which requires high computational capacities.

Among all the models, SVM has produced highest classification accuracy of around 80 percent when provided with the subset of features selected using MI technique. Fig. 6 shows the accuracy of models with different feature selection techniques. NB was the worst performing model among all on all feature selection techniques. Interestingly, the performance of NB degrades upon feature selection in comparison to every feature when used in training. This poor performance of the NB model may be due to its working mechanism and simplicity. Multilayer Perceptron, was expected to perform better than traditional models given its neural network-based background, however, it lags in performance and could not surpass the performance of SVM. This unexpected behavior may be due to the small size of the training samples used in the study as the neural network based models are best suited for high dimensional datasets.

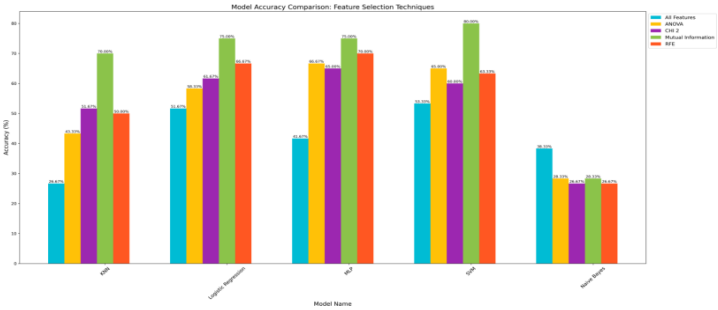

Fig. 6. Comparison of accuracy, precision, recall and f1 score of models

TABLE VII. IMPACT OF RECURSIVE FEATURE ELIMINATION (RFE) BASED FEATURE SELECTION ON MODEL PERFORMANCE

Comparison	Accuracy Difference	Improvement (%)
Proposed techniques Vs KNN	80-70	10%
Proposed techniques Vs LR	80-75=5	5%
Proposed techniques Vs MLP	80-75=5	5%
Proposed techniques Vs NB	80-28.33=51.67	51.67%

The comparative analysis underscores the advancement of intelligent methods in software risk prediction, as reflected in reported accuracy levels. Among the evaluated models, SVM demonstrates the highest performance, while Naïve Bayes records the lowest as shown in table 7. The Mutual Information (MI) method proves effective for feature selection in risk prediction tasks. Overall, the findings reveal a consistent improvement in prediction accuracy over time, with contemporary AI techniques substantially enhancing both accuracy and decision-making support in software project management.

VI. CONCLUSION

In the rapidly evolving field of software development, Software Risk Prediction is indispensable for building secure, reliable, and high-quality software systems. Early identification and mitigation of risks help organizations reduce vulnerabilities, improve software resilience, and sustain stakeholder confidence. With the rising complexity of software applications and the increasing sophistication of cyber threats, developing accurate risk prediction models is emerging as a crucial area of research. In the current study, the authors evaluated various models in which SVM demonstrated the highest classification accuracy (approximately 80%) when trained on features selected using the MI technique. In contrast, FNB and Naive Bayes NB exhibited the weakest performance across all feature selection methods. Notably, NB's accuracy further declined when trained on selected features compared to using the full feature set, likely due to its inherent simplicity and probabilistic assumptions. Surprisingly, the MLP, despite its neural network-based architecture, underperformed relative to traditional models

like SVM. This could be attributed to the limited training dataset size, as optimum level of performance is achieved by models of deep learning only when they get large-scale data. Furthermore, SVM emerged as the most effective model for software risk prediction, while NB-based approaches proved least effective. Additionally, the MI technique was found to be a reliable method for feature selection in risk prediction tasks. Future research could explore larger datasets and hybrid models to further enhance predictive accuracy and robustness in real-world software development scenarios.

VII. ETHICAL CONSIDERATIONS

Not applicable.

VIII. CONFLICT OF INTEREST

The authors declare no conflicts of interest.

IX. FUNDING

This research did not receive any financial support.

REFERENCES

[1] J. Masso, F. J. Pino, C. Pardo, F. García, M. Piattini: Risk management in the software life cycle: A systematic literature review, Comput. Stand. Interfaces, vol.71, 103431 (2020).

[2] B. Verma, M. Dhanda, M.: A review on risk management in software projects, International Journal for Innovative Research in Science & Technology, vol.2, pp.499–503 (2016).

[3] R. K. Bhujang, V. Suma: A Comprehensive Solution for Risk Management in software development projects, Int. J. Intell. Syst. Technol. Appl., vol.17, pp.153–175 (2018).

[4] A. A. M. Chowdhury, S. Arefeen: Software risk management: Importance and practices, IJCIT, vol.2, no.1, pp.49-54.

[5] A. A. Keshlaf, S. Riddle: Risk management for web and distributed software development projects, In Proceedings of the 2010, Fifth International Conference on Internet Monitoring and Protection, Barcelona, Spain, pp. 22–28 (2010).

[6] B. Kitchenham, S. Charters: Guidelines for performing systematic literature reviews in software engineering, In EBSE Technical 547 Report; University of Durham: Durham, UK, vol.2 (2007).

[7] A. Elzamly, B. Hussin: Classification and identification of risk management techniques for mitigating risks with factor analysis technique in software risk management, Rev. Comput. Eng. Res., vol.2, pp.22–38 (2015).

[8] M. H. Mahmud, M. T. H. Nayan, D. M. N. A. Ashir, M. A. Kabir: Software Risk Prediction: Systematic Literature Review on Machine Learning Techniques, Applied Sciences, vol.12, no.22, 11694 (2022).

[9] Y. Hu, X. Zhang, E. Ngai, R. Cai, M. Liu: Software project risk analysis using Bayesian networks with causality constraints, Decision Support System vol.56, pp.439–449 (2013).

[10] H. A.Salih, H. H. Ammar: Model-based resource utilization and performance risk prediction using machine learning Techniques, JOIV Int. J. Inform. Vis., vol.1, pp.101–109 (2017).

[11] P, G. Sankaranarayanan, S. Prediction of Risk Percentage in Software Projects by Training Machine Learning Classifiers. Comput. Electr. Eng. vol.94, 107362, 2021.

[12] S. R. Bauskar, C. R. Madhavaram, E. P. Galla, J. R. Sunkara, H. K. Gollangi, S. K. Rajaram, "Predictive Analytics for Project Risk Management Using Machine Learning," Journal of Data Analysis and Information Processing, vol.12, pp.566-580, Nov. 2024.

[13] S. Kalogiannidis, D. Kalfas, O. Papaevangelou, G. Giannarakis, F. Chatzitheodoridis: The Role of Artificial Intelligence Technology in Predictive Risk Assessment for Business Continuity: A Case Study of Greece, MDPI- Risks 2024, vol.12, no.19, pp.1-23 (2024).

[14] Y.P. Vera, Á.F.D. Carpio: Comparing machine learning techniques for software requirements risk prediction, Indonesian Journal of Electrical Engineering and Computer Science, Vol. 34, No. 1, pp. 508~519, DOI: 10.11591/ijeecs.v34.i1 (2024)

[15] M.N. Alatawi, S. Alyahyan, S. Hussain, A. Alshammari, A.A. Aldaeej, I.K. Alali, H.S. Alwageed: A Data-Driven Artificial Neural Network Approach to Software Project Risk Assessment, Hindawi IET Software, vol. 2023, PP. 1-19 (2023).

[16] Y-ha Chun: An Empirical Study of Intelligent Security Analysis Methods Utilizing Big Data, Webology, vol. 19, No. 1, PP. 4672-4681 (2022).

[17] R. Naseem, Z. Shaukat, M. Irfan, M. A. Shah, A. Ahmad, F. Muhammad, A. Glowacz, L. Dunai, J. A. Antonino-Daviu, A. Sulaiman, A.: Empirical Assessment of Machine Learning Techniques for Software Requirements Risk Prediction, MDPI, Electronics, vol.10, no.168, pp.1-19 (2021).

[18] R. Ferenc, P. Gyimesi, G. Gyimesi, Z. Tóth, T. Gyimóthy: An automatically created novel bug dataset and its validation in bug prediction, The Journal of Systems & Software, Vol. 169, pp. 1-20 (2020).

[19] Hastie, T. Tibshirani, R.Friedman, J.: The Elements of Statistical Learning; Springer, New York, NY, USA, (2009)

[20] Joachims, T. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In European Conference on Machine Learning; Springer Berlin/Heidelberg, Germany (1998).

[21] Available at: https://github.com/bharlow058/SoftwareRiskPrediction/tree/main.

[22] Sebastiani, F. Machine learning in automated text categorization. ACM Comput. Surv, 34, 1–47 (2002).

[23] Zhao, R.; Kezhi, M. Fuzzy bag-of-words model for document representation. IEEE Trans. Fuzzy Syst. 26, 794–804 (2017).