Empirical Analysis of Computationally Intelligent Technique for Software Risk Prediction

Mohd Shabbir[1,*], Rakesh Kumar Yadav[1], Mohd Waris Khan[2], Hitendra Singh[3]
[1]Department of Computer Science and Engineering, MUIT, Lucknow, India
[2]Department of Computer Application, Integral University, Lucknow, India
[3]Department of Electronic s and Communication Engineering, MUIT, Lucknow, India
Emails: shabbir.aec@gmail.com; rkymuit@gmail.com; wariskhan070@gmail.com; hit.singh111@gmail.com

Abstract

Software development is inherently associated with a high degree of uncertainty, often arising from unforeseen activities during different phases of the SDLC. As software systems expand in scale and complexity, the likelihood of failures and project delays also increases. Such situations, which are usually not anticipated, are known as software risks. They arise due to different reasons, which affect activities like essentials of engineering, making, putting into usage, and test. These risks need to be identified and managed in the initial phase for delivering software-related products that are both excellent and can be relied upon. While it has been standard practice in assessing software risks to depend upon human skills and previous experiences, it has been observed they lead to issues in consistency and often are reported to be unreliable. The current study is an attempt to tackle this issue through usage of predictive models that have their roots in machine learning (ML). Borrowing from existing data, software risks are identified and classified through five popular machine-learning tools. To improve correctness and make it more robust, selection techniques of selection with multiple features are implemented. Among the other models, the Support Vector Machine (SVM) exhibited the maximum performance, achieving a classification accuracy of approximately 80%, with a precision of 84%, recall of 80%, and an F1 score of 80%. In terms of performance, Mutual Information was found to be best in methods of applied feature selection. The study indicates the ability of ML based methods in predicting and managing software risks. Additionally, this research highlights the potential of computationally intelligent techniques to assist project managers in early risk identification, proactive decision-making and enhancing the overall success rate of s/w projects.

Keywords: Software Risk; Software Risk Prediction; Risk Management; Requirement Analysis; AI & Machine Learning

1. Introduction

In Software Engineering (SE), it is important to anticipate risks because it leads to the success of a project. Therefore, it is not just a safety measure but also a critical phase in SE, which is an organized and methodical approach to designing, developing, and continuing software systems. All the previously mentioned steps in SE stem from Software Development Life Cycle (SDLC), which guides all phases of the process [1]. In spite of all this, there may be delays of entire projects might be unsuccessful because of the vulnerability of SDLC [2]. Such vulnerabilities are the result of unfinished, ambiguous or constant demands of changes in the project and are known as software risks [3]. It is vital to anticipate risks correctly because it is strongly related to outcome of any project. Hence, proper risk management becomes critical for project in general and SDLC in particular [4, 5].

Analysing risk is important in SDLC as it includes identifying possible risks and application of mitigation plans. To overcome shortcomings, it becomes important to be proactive towards risk [6]. Incorrect identification or mismanagement of risk may have damaging results, even complete collapse of the project [7]. In SDLC, assessing

DOI: https://doi.org/10.54216/JCIM.170214
Received: April 02, 2025 Revised: June 16, 2025 Accepted: August 14, 2025

risk should be an ongoing critical part as their proper and timely identification may reduce costs and effort, which in turn enhances effectiveness and results.

Significantly, not only project management but also cybersecurity and knowledge management are also linked to software risks. Problems in digital infrastructure may result in breach of data resulting in significant losses of both information and privacy. It can also threaten security resulting in cyberattacks such as malware injection, denial-of-service, or insider threats. Likewise, improper risk management might hamper secrecy, accuracy and accessibility, all of which form the core of an efficient information management system. Therefore, anticipating risks properly means protecting vital information and making sure standards of security are met specially in defence, medicine and financial services where it can be even more damaging.

An important upcoming area in research is predicting risks in SDLC. To make it more accurate and reliable, intelligent techniques are being increasingly utilized in evaluation of risks. Since they can occur anywhere in SDLC, it becomes necessary to adopt a proactive approach to risk management. In spite of proactivity, risk vulnerability may remain for projects due to lack of finances, limited time durations and issues of quality management for bypassing them may lead to problems all across the project [8, 9]. It is expected that as a project moves on, a good system of anticipates, checks and manages risks continuously so that they are noticed and corrective steps taken well in time leading to project success [10].

It has been proven in recent research that good results are achieved when several methods of assessing risks are combined. Significant risk identification was achieved when dataset of NASA 93 was used along with Adaptive Neuro-Fuzzy Inference System (ANFIS) and Evolutionary Computation and Swarm Algorithms (ECSA) [11]. Similarly, ensemble AI models have surpassed individual models in detecting buggy Java classes, especially within open-source projects like Apache Commons. These findings indicate that ensemble techniques can significantly enhance risk prediction accuracy. Furthermore, machine learning continues to gain interest in the area of software risk prediction [12]. Research indicates that models such as Decision Trees & Random Forests are effective in classifying risk levels, aiding project managers in strategic decision-making

Building on these insights, this paper aims to compare and evaluate multiple intelligent approaches for software risk prediction, with a particular emphasis on their relevance to cybersecurity and information management. Through empirical analyses, the study seeks to regulate the operative techniques for identifying and mitigating risks in software projects. The results are expected to provide practical guidance to both researchers and practitioners, ultimately strengthening software reliability, safeguarding sensitive data, enhancing the resilience of information systems against security threats and increasing the likelihood of project success.

Our key findings from this research can be summarized as follows:

- Machine learning models demonstrated superior capability in predicting software risks, establishing themselves as state-of-the-art approaches for risk assessment.
- The study highlighted that selecting critical features significantly improves the predictive accuracy of intelligent risk assessment models.
- Applying intelligent risk prediction techniques during the early stages of the SDLC proved effective in detecting potential risks, thereby reducing the chances of delays and failures.
- AI-driven risk prediction models exhibited both scalability and adaptability, making them applicable to projects of varying sizes-from small-scale developments to large-scale enterprise systems.

The rest of this paper is structured as follows: Section 2 provides a short review of standing literature, outlining the methodologies employed and their corresponding explanations. Section 3 introduces the classification models, offering detailed insights into their design and functionality. Section 4 describes the model setup and configurations adopted in this study. Section 5, being the core of the research, presents the results and discussion, supported by analytical interpretations of the empirical findings. Finally, Section 6 concludes the paper and outlines actionable recommendations based on the insights obtained.

2. Review of related studies

Table 1 presents a consolidated summary of various software risk prediction techniques, highlighting their authors, research titles, applied techniques and brief description.

DOI: https://doi.org/10.54216/JCIM.170214
Received: April 02, 2025 Revised: June 16, 2025 Accepted: August 14, 2025

Table 1: Summary on Software Risk Prediction Techniques

Author(s)	Title	Techniques Used	Description
Arash Salehpour, et al., 2025 [13]	A critical analysis of the impact of feature selection and feature engineering on credit risk assessment using machine learning techniques: A systematic review	Feature selection (Filter, Wrapper, Embedded, Hybrid), Dimensionality Reduction (PCA, ICA, LDA, NMF), Discretization, Transformation, Ensemble Models	The study systematically reviewed on credit risk assessment, analyzing how feature selection and engineering methods affect machine learning and deep learning performance. Results showed that CFS and PCA were the most effective techniques. Random Forests and SVM reached the highest accuracy and robustness. The review highlights dataset imbalance, reproducibility issues, and the need for advanced ensembles and alternative data sources.
Yasiel Pérez Vera., et al, 2024 [14]	Comparing Machine Learning Techniques for Software Requirements Risk Prediction	Logistic Regression, MLP-NN, 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. The study focuses on predicting total project risk and individual risk factors using historical data. Results indicate that ANNs achieve up to 99.12% accuracy in total risk prediction, aiding project managers in 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 author explores how big data can be leveraged to design intelligent security analysis methods aimed at countering advanced cyber threats, such as APT attacks. The study introduces a behavior-based, correlation-focused approach that analyzes system logs, network traffic, and activity patterns to detect and prevent attacks before they occur. Furthermore, it integrates data from multiple sources with advanced analytics to overcome the limitations of signature-based detection, emphasizing intelligent real-time security frameworks for enhanced threat detection and organizational defense.
Rashid Naseem, et al., 2021 [17]	Empirical Assessment of Machine Learning Techniques for Software Requirements Risk Prediction	Machine Learning (ML) Techniques	In this paper, the author highlighted that the risk levels can be anticipated during the estimation of s/w requirements by employing different ML approaches. Empirical assessments were carried out by the researchers, which shown the effectiveness of these approaches and provide meaningful insights into their applicability within real-world s/w development contexts.

DOI: https://doi.org/10.54216/JCIM.170214
Received: April 02, 2025 Revised: June 16, 2025 Accepted: August 14, 2025

Rudolf Ferenc, et al., 2020 [18]	An automatically created novel bug data set and its validation in bug prediction	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 it through machine learning-based bug prediction.
A.O. Balogun, et al., 2019 [19]	Comparative Performance Analysis of Feature Selection Methods for Software Defect Prediction	FSS techniques—such as CFS & CNS—connect with search tactics (BFS, BAT, FS, AS, GSS, PSOS, GS), together with Decision Tree (DT)	The study examined the effect of different feature selection methods S/w Defect Prediction, showing that IG (FFR) attained the best improvements, CNS (BFS) was the top among FSS method, FFR-based models were more stable, and FS effectiveness varied across datasets and classifiers due to class imbalance.

Table 1 provides a comparative summary of recent research on software risk prediction utilizing advanced machine learning and AI techniques. Feature selection and ensemble models proved crucial for improving accuracy and stability, while ANN and Random forest consistently outperformed other approaches in handling complex risks. The key findings consistently report notable improvements in prediction accuracy, resource optimization, and cost reduction, with AI-driven approaches outperforming traditional methods. Furthermore, several works emphasize practical advantages such as real-time monitoring, proactive risk mitigation, and improved decision-making, underscoring the expanding role of AI and ML in managing software project risks across diverse domains.

3. Classification Models

A. Naive Bayes (NB)

The Naive Bayes classifier is often utilized in tasks related to classification of text such as spam filtering and analysis of sentiments. An application of Bayes' Theorem, this probabilistic model works on the given observed features and calculates class probability. Here it is assumed that in a certain class label, these features tend to be independent conditionally [19]. Although the assumptions of independence are quite strong, NB often performs surprisingly well in practice, especially in scenarios involving count-based features like word frequencies. It is highly efficient, scalable, and interpretable but may struggle when features are highly correlated or when applied to complex datasets with intricate dependencies.

B. K-Nearest Neighbours (KNN)

K-Nearest Neighbors, known commonly by the abbreviation KNN, is a machine-learning algorithm, which appears simple but is actually very powerful. Its' primary usage is for classifying tasks and in performing regression. KNN works on the given input through identification of their 'k' nearest data points on some chosen metric of distance, for example Euclidean distance. This algorithm categorizes input based on common class among its neighbors or calculates the average of neighbor's value and makes a prediction in the case of regression [19]. Since KNN does not assume anything about the underlying dispersal of data, it is non-parametric in nature. Thus, it is highly flexible but in the case of larger datasets, it tends to become quite costly in terms of computation. KNN performs well when the data is well structured and noise-free, but it may face challenges with high-dimensional data due to the effects of increased dimensional space.

C. Support Vector Machine (SVM)

Useful in regression as well as classification, Support Vector Machine is an algorithm with several properties. This learning algorithm is supervised and quite powerful. It operates by searching for an optimal hyperplane, which is the best separator of various classes in a high-dimensional space. This, in turn, maximizes margin between data points among different classes [19]. It changes the input space and handles data that is separable and non-linear by using kernel functions, like radial basis, polynomial and linear. In spaces that are high dimensional, SVM has been found to be robust, and can tackle both types of relationships; non-linear and linear [20]. The drawbacks of SVM are its' sensitivity to parameter tuning and tendency to be intensive computationally.

D. Logistic Regression

It is one of the models in statistics, which is quite prevalent and applies to problems related to binary and multi-class classification. While only the continuous values are predicted in linear regression, logistic regression determines whether any particular input can be classified in some definite class. This is done by using sigmoid function to a combination of features of input, linear in nature [19, 21] wherein the output is mapped between 0 and 1, making it ideal with respect to probabilistic interpretation. Logistic regression is simple, efficient, and interpretable, often serving as a baseline model in many applications. However, the assumption that the

DOI: https://doi.org/10.54216/JCIM.170214
Received: April 02, 2025 Revised: June 16, 2025 Accepted: August 14, 2025

relationship between features of input and the log-outcome log-odds is linear, may limit its performance on complex, non-linear datasets.

E. Multi-Layer Perceptron (MLP)

Multi-Layer Perceptron (MLP) is a part of artificial neural networks that involves of multiple layers of consistent neurons, with an input layer, one or further concealed layers, and an output layer. MLP leverages back propagation and gradient descent algorithms to learn complex patterns by adjusting the network's weights through iterative optimization [19, 22]. It can model both linear and non- linear relationships, construction it a flexible high quality for classification and regression tasks. It has been observed that MLPs are effective in processing large-scale structured data, though their performance largely depends on the careful adjustment of hyperparameters such as the learning rate, activation functions, and hidden layer architecture.

4. Experimental Setup

A. Dataset Description

This study used a publicly available dataset containing 299 instances and 13 features, including both categorical (e.g., Requirements, Project Category, Prerequisite Category, and Risk Goal Category, Extent of Risk, Impact, Measurement of Risk) and numerical attributes (e.g., Probability, Affecting Numeral of Modules, Fixing Interval, and Priority) [23]. The target variable "Risk Level" corresponds to class and has values from 1 to 5. The dataset facilitates the analysis and classification of project risks based on multiple factors, making it suitable for machine learning experiments.

B. Data Preprocessing

The dataset underwent a series of pre-processing steps to enhance its reliability and prepare it for efficient and effective machine learning.

i. Column Stripping

All column names were stripped of leading and trailing whitespace characters to ensure consistency and prevent potential issues during processing.

ii. Handling Target Class

The mark variable, Risk Level, was converted to a numeral data type to ensure it is in the appropriate format for model training.

iii. Handling Missing Data

Missing values are imputed using the median of each column, as median imputation helps mitigate the impact of outliers that could distort the data.

iv. Text Preprocessing

In the 'Requirements' column, text is tokenized, converted to lowercase, stop words are removed, and lemmatization is performed to normalize the text. This results in cleaner, more meaningful text data, ready for feature extraction. The original 'Requirements' column is then dropped to avoid redundancy.

v. Feature Representation

The following feature representation technique was applied:

- **Ordinal Encoding:** Ordinal variables such as 'Magnitude of Risk' and 'Impact' are encoded into numerical values using predefined orders (from low to high).
- **One-Hot Encoding:** Categorical variables like 'project Category' and 'Requirement Category' are transformed into binary format through one-hot encoding; creating separate columns for each category, helping the model interpret these features better.
- **Bag of Words (BoW):** The 'Requirements Cleaned' column, which has undergone text preprocessing, is transformed into a Bag of Words representation using Count Vectorizer. This step converts text into numerical features based on word frequency, making it digestible for machine learning models [24, 25].
- **Min-Max Scaling:** Numerical features such as 'Probability' and 'Fixing Duration (Days)' are scaled to a fixed range (0 to 1) using Min-Max Scaler. This is important for ensuring equal contribution of all features to the model, especially when using algorithms sensitive to feature scaling.
- **SMOTE (Synthetic Minority Over-sampling Technique):** To report class imbalance, SMOTE is applied to the training dataset, creating synthetic models of the minority class to balance the dispersal of the target variable. This ensures that the model does not favor the majority class during training.

C. Feature Selection

DOI: https://doi.org/10.54216/JCIM.170214
Received: April 02, 2025 Revised: June 16, 2025 Accepted: August 14, 2025

Feature selection helps in choosing the most important features from the dataset, which can improve model performance and reduce complexity [26].

i. ANOVA (Analysis of Variance

Used for selecting features when the target variable is categorical and the input features are numerical. It checks the means of different groups for any significant difference. A higher F-statistic value means a feature is more relevant.

ii. Chi-Square Test

Used for selecting categorical features by checking whether the target variable and the features have any relationship. This test compares observed and expected frequencies in each category. A high Chi-square value indicates a strong relationship between the feature and the target.

iii. Mutual Information

Measures the degree to which knowledge about one variable results in uncertainty reduction in the other. While correlation determines only the linear relationship, mutual information deals with the non-linear relationship as well. A higher mutual information value means the feature is important for predicting the target.

iv. Recursive Feature Elimination (RFE)

Selects features by training the model multiple times and eliminating the smallest significant feature in each iteration. It helps determine smallest features set which provides the best performing model.

v. Evaluation Metrics

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

vi. Accuracy:

It is a calculation of how many instances are correctly classified against the total occurrences in dataset. Through accuracy, it becomes evident if the overall model is correct or not. However, in the case of an imbalance in the dataset when the weightage of one class may be significantly more than others, accuracy fails to provide the right picture. It is computed by the formula:

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

vii. Precision

It is based on accuracy of constructive predictions that is positively predicted instances are accurate. This metric assumes importance where an incorrect positive prediction could have severe consequences like in detecting fraud or diagnosis of diseases. Precision is considered using the formula:

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

viii. Recall (Sensitivity or True Positive Rate)

Recall focuses on minimizing false negatives, meaning the amount 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))

ix. F1 Score

It is a well-balanced measure which has a single metric combining recall and precision. F1 Score is harmonic mean of both the measures, which ensures both positives and negatives that are false, are fully considered. F1 Score comes in handy in case of an imbalance among false positives and false negatives and a trade-off is needed between recall and precision. It is determined using the formula:

F1 Score = (2×Precision×Recall) / (Precision + Recall)

D. Parameter Settings

All classification models were executed using their default parameters. No additional optimization or fine-tuning was applied during the experiments. The use of default settings was intentional to provide a consistent and unbiased baseline comparison of model performance across different feature selection techniques. Since the main aim of this study was to evaluate the relative effectiveness of classifiers and feature selection methods, rather than to

DOI: https://doi.org/10.54216/JCIM.170214
Received: April 02, 2025 Revised: June 16, 2025 Accepted: August 14, 2025

maximize the performance of any single model, hyperparameter tuning was not performed. Moreover, tuning can be computationally intensive and highly dataset-specific, which is reserved for future research.

5. Results and Discussion

The experiments were implemented in Python using libraries such as NumPy, pandas, scikit-learn, imbalanced-learn, NLTK, Matplotlib, Seaborn, and SciPy. The research was carried out on Google Colab with 13.61 GB of RAM and 2 CPU cores, which provided adequate computational resources for efficient model training and evaluation. After preprocessing, the dataset contained 714 features due to the application of the Bag-of-Words (BoW) representation on the textual column.

Table 2 reports the performance of the five classifiers without applying any feature selection. Among them, the Support Vector Machine (SVM) achieved the highest classification accuracy, while K-Nearest Neighbours (KNN) exhibited the weakest performance. The superior performance of SVM can be attributed to its effectiveness in handling high-dimensional datasets. Regarding other evaluation metrics, the models displayed relatively similar outcomes; however, the F-score proved useful in distinguishing performance differences. Both SVM and Logistic Regression (LR) achieved higher F-scores compared to the remaining classifiers, as shown in Table 2. Figure 1 further illustrates the accuracy levels obtained by each model when trained on all available features.

Table 2: Comparative Performance of Models across all Features

Classifier	Accuracy	Precision	Recall	F-Score
NB	38.53	52.78	38.53	40.77
LR	51.88	57.25	51.87	52.24
SVM	53.53	52.99	53.53	53.03
MLP	41.87	48.08	41.87	42.99
KNN	26.87	66.09	26.87	25.23

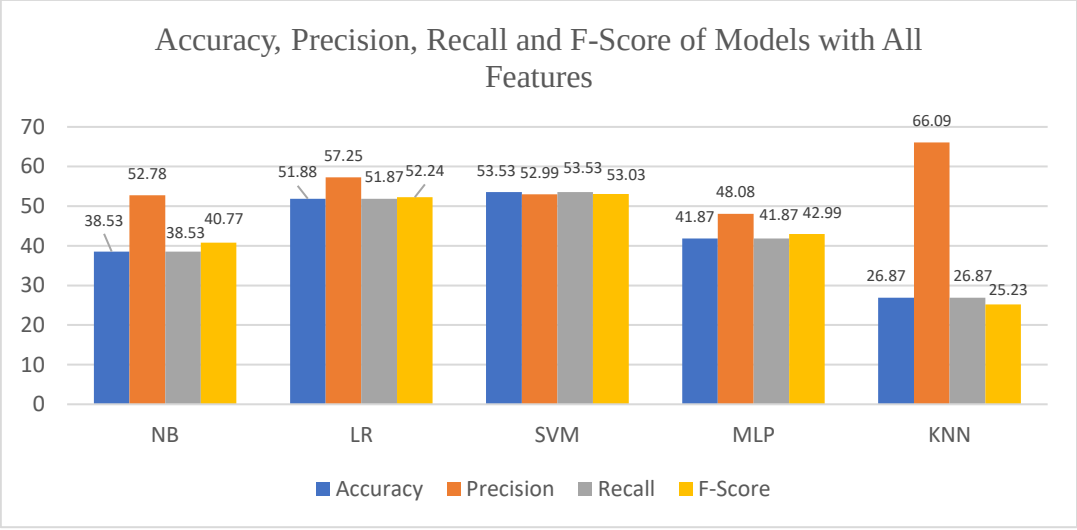

Figure 1. Performance Comparison of Classification Models Using All Features

Subsequently, the models were evaluated using five different filter-based feature selection techniques. For each technique, models were trained on feature subsets ranging from 1 to 714, with increments of 25. The best results across all models were observed when 25 features were selected. Figure 2 presents the performance of different feature selection methods under this configuration. Among them, Mutual Information (MI) consistently delivered the strongest results across all evaluation metrics. These findings align with prior research, where MI has been recognized as an effective and computationally efficient feature selection method. In contrast, techniques such as Recursive Feature Elimination (RFE) require higher computational resources, making MI a more practical choice for large-scale experiments.

DOI: https://doi.org/10.54216/JCIM.170214
Received: April 02, 2025 Revised: June 16, 2025 Accepted: August 14, 2025

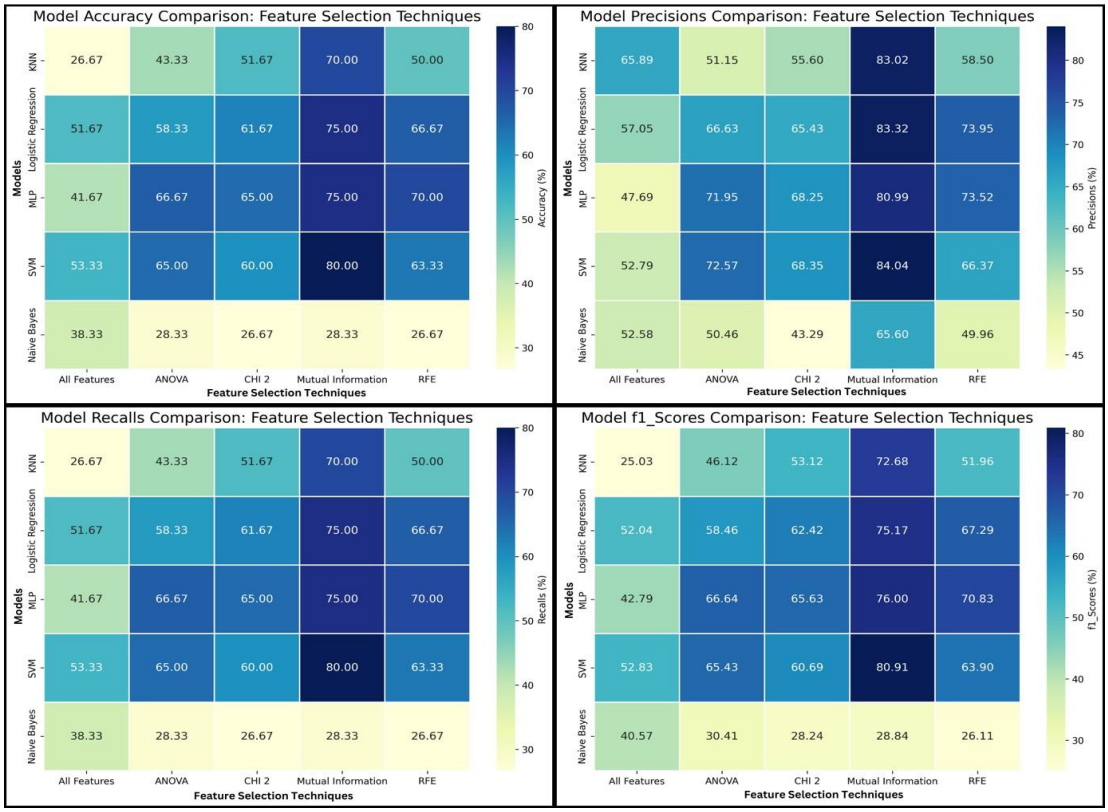

Figure 2. Performance of different feature selection techniques

Specifically, the best and worst performing configurations were compared to highlight the performance range. The findings reveal that among all the tested models, the SVM delivered the highest performance, attaining an accuracy of nearly 80%, with a precision of 84%, recall of 80% and F1-score of 80%, whereas Naïve Bayes with Chi² attained only 26.67% accuracy, 28% precision, 27% recall and 28.24% F1-score. This corresponds to substantial margins of improvement for SVM, +53.33% in accuracy, +56% in precision, +53% in recall and +53.66% in F1-score. These differences were statistically significant ($p < 0.05$), thereby validating the superiority of SVM with Mutual Information over Naïve Bayes with Chi². This strong outcome was observed particularly when trained on features selected using the Mutual Information method. Figure 2 depicts a comparative overview of model accuracies under different feature selection approaches. On the other hand, Naïve Bayes (NB) classifier consistently achieved the lowest scores across all techniques. Significantly, upon application of feature section, the performance of NB's was found to even poorer in comparison to its performance when all features were used for training. This inadequacy is mainly due to NB's simplistic probabilistic framework and rigid independence assumption, which limit its ability to help from dimensionally reduction techniques like MI.

The Multilayer Perceptron (MLP), despite being a neural network-based model, failed to surpass SVM. Although anticipated to deliver stronger results than traditional algorithms, its accuracy remained lower, most likely because of the limited training dataset, as neural network-based models generally perform optimally with larger, high-dimensional data. Overall, SVM proved to be most effective classifier, while Naïve Bayes (NB) exhibited the weakest results. Additionally, the MI technique was validated as an efficient feature selection method for software risk prediction.

6. Conclusion

To build reliable, quality and secure systems, it is essential to anticipate risks in modern especially when developing software is a fast evolving and quick process. When risks are identified and resolved in the initial phases it results in lesser problems, better resilience, and stronger confidence of the client. In modern times, complexity and threats are increasing and so developing models that can forecast risks with precision is an important area of research. This study assessed multiple predictive models, with the Support Vector Machine (SVM) delivering the highest classification accuracy, approximately 80%, when trained using features selected via the Mutual Information (MI) method. In comparison, Naïve Bayes (NB) and KNN consistently underperformed, with NB showing further accuracy loss when trained on selected features. MLP and Logistic

DOI: https://doi.org/10.54216/JCIM.170214
Received: April 02, 2025 Revised: June 16, 2025 Accepted: August 14, 2025

regression showed moderate performance but did not surpass SVM, likely due to the dataset size and complexity constraints. In summary, SVM proved to be the most effective model for predicting software risks, while NB-based models were the least effective. The study also validated the Mutual Information technique as a dependable and efficient approach for feature selection in this context.

This research highlights the potential of computationally intelligent techniques in software risk prediction; however, several promising directions remain for future exploration. Future research could explore the use of larger and more diverse datasets to strengthen the generalizability of models across varied software projects. The development of hybrid models, combining multiple learning techniques, may further enhance prediction accuracy and robustness in real-world environments. Additionally, the design of real-time and adaptive risk prediction systems could provide continuous monitoring throughout the software lifecycle. Incorporating explainable AI mechanisms will be essential for improving interpretability and fostering stakeholder trust. Finally, extending the framework to account for human, organizational, and domain-specific risk factors could lead to more comprehensive and practically applicable solutions.

References

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

[2] B. Verma and M. Dhanda, "A review on risk management in software projects," *Int. J. Innov. Res. Sci. Technol.*, vol. 2, no. 11, pp. 499–503, 2016.

[3] R. K. Bhujang and V. Suma, "A Comprehensive Solution for Risk Management in software development projects," *Int. J. Intell. Syst. Technol. Appl.*, vol. 17, no. 1, pp. 153–175, 2018.

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

[5] A. Keshlaf and S. Riddle, "Risk management for web and distributed software development projects," in *Proc. 5th Int. Conf. Internet Monitor. Protection*, Barcelona, Spain, May 2010, pp. 22–28.

[6] S. R. Bauskar *et al.*, "Predictive Analytics for Project Risk Management Using Machine Learning," *J. Data Anal. Inf. Process.*, vol. 12, no. 11, pp. 566–580, Nov. 2024.

[7] Elzamly and 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, no. 1, pp. 22–38, 2015.

[8] M. H. Mahmud *et al.*, "Software Risk Prediction: Systematic Literature Review on Machine Learning Techniques," *Appl. Sci.*, vol. 12, no. 22, p. 11694, 2022.

[9] Y. Hu, X. Zhang, E. Ngai, R. Cai, and M. Liu, "Software project risk analysis using Bayesian networks with causality constraints," *Decis. Support Syst.*, vol. 56, pp. 439–449, 2013.

[10] H. A. Salih and H. H. Ammar, "Model-based resource utilization and performance risk prediction using machine learning Techniques," *JOIV Int. J. Inform. Vis.*, vol. 1, no. 4, pp. 101–109, 2017.

[11] K. Suresh and R. Dillibabu, "An integrated approach using IF-TOPSIS, fuzzy DEMATEL, and enhanced CSA optimized ANFIS for software risk prediction," *Knowl. Inf. Syst.*, vol. 63, no. 7, pp. 1909–1934, Jul. 2021.

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

[13] Salehpour, K. Samadzamini, and R. J. Moghadam, "A critical analysis of the impact of feature selection and feature engineering on credit risk assessment using machine learning techniques: A systematic review," *SSRN*, Mar. 2025. [Online]. Available: https://ssrn.com/abstract=5183264

[14] Y. P. Vera and Á. F. D. Carpio, "Comparing machine learning techniques for software requirements risk prediction," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 34, no. 1, pp. 508–519, Apr. 2024.

[15] M. N. Alatawi *et al.*, "A Data-Driven Artificial Neural Network Approach to Software Project Risk Assessment," *IET Softw.*, vol. 2023, pp. 1–19, Dec. 2023.

[16] Y. Chun, "An Empirical Study of Intelligent Security Analysis Methods Utilizing Big Data," *Webology*, vol. 19, no. 1, pp. 4672–4681, Jan. 2022.

DOI: https://doi.org/10.54216/JCIM.170214
Received: April 02, 2025 Revised: June 16, 2025 Accepted: August 14, 2025

[17] R. Naseem *et al.*, "Empirical Assessment of Machine Learning Techniques for Software Requirements Risk Prediction," *Electronics*, vol. 10, no. 2, p. 168, 2021.

[18] R. Ferenc, P. Gyimesi, G. Gyimesi, Z. Tóth, and T. Gyimóthy, "An automatically created novel bug dataset and its validation in bug prediction," *J. Syst. Softw.*, vol. 169, p. 110693, Sep. 2020.

[19] O. Balogun, A. S. Hashim, S. Basri, and S. J. Abdulkadir, "Performance Analysis of Feature Selection Methods in Software Defect Prediction: A Search Method Approach," *Appl. Sci.*, vol. 9, no. 13, p. 2764, 2019.

[20] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," in *Proc. Eur. Conf. Mach. Learn.*, Berlin, Germany, 1998, pp. 137–142.

[21] Z. Gao, H. Liu, and L. Li, "Data Augmentation for Time-Series Classification: An Extensive Empirical Study and Comprehensive Survey," *J. Artif. Intell. Res.*, vol. 83, art. 5, Jun. 2025.

[22] J. Dodge, Q. V. Liao, Y. Zhang, R. K. E. Bellamy, and C. Dugan, "Explaining Models: An Empirical Study of How Explanations Impact Fairness Judgment," in *Proc. 24th Int. Conf. Intell. User Interfaces*, Mar. 2019, pp. 275–285.

[23] F. Sebastiani, "Machine learning in automated text categorization," *ACM Comput. Surv.*, vol. 34, no. 1, pp. 1–47, 2002.

[24] R. Zhao and M. Kezhi, "Fuzzy bag-of-words model for document representation," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 2, pp. 794–804, Apr. 2017.

[25] S. R. Bauskar *et al.*, "Predictive Analytics for Project Risk Management Using Machine Learning," *J. Data Anal. Inf. Process.*, vol. 12, no. 11, pp. 566–580, Nov. 2024.

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

DOI: https://doi.org/10.54216/JCIM.170214
Received: April 02, 2025 Revised: June 16, 2025 Accepted: August 14, 2025