A Machine Learning Attempt for Anatomizing Software Risks in Small and Medium Agile Enterprises

Ayesha Ziana Mohamed
Department of Computer Science
Noorul Islam Centre for Higher Education, India
ayeshaziana@outlook.com

Charles Jebapillai
Department of Software Engineering
Noorul Islam Centre for Higher Education, India
charlesj.noorul@outlook.com

Abstract: *The ultimate aim of customer satisfaction and the increasing number of unexpected risks in a changing Agile Software Development (ASD) environment, one of the most important rising demands is in the area of systematic but light-weight risk management tools and methodologies. Risk analysis is a significant phase in the process of risk assessment, which helps to evaluate the risks in order to mitigate them effectively within a limited duration. Recently, machine learning algorithms have become popular for solving problems in various domains, including software risk analysis and prioritization, due to their better performance and efficiency. With this aspect, an approach for predicting the level of software risks with the proposed risk dataset has been attempted in this study with the basic machine learning algorithms for risk classification purposes. The logistic regression, decision tree, Support Vector Machine (SVM), naïve bayes, and K-Nearest Neighbor (KNN) algorithms were implemented in the experimental analysis. The results reveal that the proposed dataset renders better outcomes with logistic regression (70% accuracy) and SVM (65% accuracy). Out of the five algorithms, the exclusion of the Agile Software Risk Identification (ASRI) framework attribute 'Risk Nature' from the overall proposed risk dataset has a more negative impact on the performance of the logistic regression, decision tree, and KNN models than the exclusion of the Goal-driven Software development Risk Management (GSRM) framework attribute 'performance goal affected'. This indicates that the 'Risk Nature' attribute plays a significant role in analyzing the risks and predicting their level of importance.*

Keywords: *Software risk analysis, software risk classification, risk prediction, machine learning, agile software risk assessment.*

Received August 26, 2024; accepted March 16, 2025
https://doi.org/10.34028/iajit/22/3/10

1. Introduction

The software industry has a keen interest in executing the agile methodology, which empowers the right guidance for a software project, thus leading to better team management and a perfectly prioritized list of changing requirements to be performed [26]. The agile methodology overcomes the traditional approach in several aspects, like low documentation, high adaptability, high user involvement, and low cost. However, it has to face unknown risks at certain times. This is due to the fact that certain risks are neglected since multiple phases of developing a project overlap, in contrast to the classical approach [1].

Agile projects require a meticulous approach for characterizing, evaluating, and managing risks due to the fact that there are chances for risks with high negative effects to be ignored, thus leading to problems in decision-making. This drawback of agile leads to the conclusion that most of the failures faced by agile projects are because of ineffectiveness in assessing and managing risks, and hence it is important to develop structured risk models and techniques [2]. In general, agile environments do not offer a systematized procedure for managing risks. Instead, they make use of an empirical control process in order to observe the project without fail; however, it is insufficient for diminishing risks. A methodology that lessens the effort and maintains due respect to the agile manifesto principles will ensure an efficient risk management process in agile projects [9].

Agile involves continuously assessing and addressing risks, engaging with stakeholders regularly, and using agile ceremonies like retrospectives to identify and mitigate issues before they escalate. Improve the integration of security operations into agile development techniques [39]. A better risk analysis has been discovered and reported as one of the vital success factors of an Agile Software Development (ASD) project in a Systematic Literature Review conducted by [33]. They also insist that the identified risks need to be analyzed, assessed, and mitigated with the best procedures after evaluation in every sprint.

Risk analysis while neglecting the absence of determination and laxity can cause the misleading of data, thus generating extensive errors. Thus, risk management is the process of discriminating, inclining, and modulating such potential issues before they turn out to be hazardous [27]. Risk analysis is the process that serves as a helping hand to learn about the nature

and level of risks (demonstrated in terms of the effects and causes of the risks). A risk-seeking attitude is essential for an organization to attain innovation, whereas measuring those risks is equally important to know their short-term and long-term negative impacts [17].

Risk analysis aims to supply the necessary information on the negative impacts of a project, thus assisting as a decision-support mechanism. Risk analysis highly influences the entire process of managing risks since it calculates the time involved and provides precision in risk management. Hence, it is vital to make a good decision while choosing a suitable risk analysis methodology [35]. Framing the right decision implies executing the risk analysis process. Risk analysis is the methodical utilization of accessible information to ascertain the probability and negative pay-offs of certain events. Their target is to reserve the role as assistance for decision-building when the possible outcomes are known ahead. The deep exploration of the outcomes by perfect risk analysis techniques results in revealing both threats and opportunities [19].

The research gap in this study lies in the need for more comprehensive and advanced machine learning approaches for risk prediction in ASD. While basic algorithms such as logistic regression and Support Vector Machine (SVM) showed moderate accuracy, the potential of deep learning, ensemble models, or hybrid techniques remains unexplored. Additionally, the dataset used may not fully represent the diverse and dynamic nature of risks in ASD, highlighting the need for larger, real-world datasets for better generalizability. Another gap is the limited analysis of feature importance and its impact on predictive performance, suggesting that more refined feature selection and engineering techniques could enhance accuracy. Finally, the study lacks real-world validation, necessitating further research on implementing these models in industry settings to assess their practical effectiveness.

Our study makes two important contributions. First, we propose a dataset with our suggested attributes, including those from the literature, and obtain the data through a survey procedure and expert opinions. Next, we report the results of the performance of different base classifier models in predicting the risk levels and perform comparison with the existing risk analysis framework attributes.

The structure of the article is compiled as follows: Section 2 comes up with a detailed review of the literature. Section 3 furnishes the proposed framework and the underlying theoretical concepts. Section 4 unveils the results of the empirical study. Section 5 discusses the limitations and threats to the validity of the study. Lastly, section 6 puts up the conclusion and the future work to be carried out.

2. Literature Review

Numerous contributions focusing on software risk analysis exist in the literature. The systematic literature review in [18] illustrates several risk analysis and management models and tools, including the probabilistic model, machine learning models, risk breakdown structure model, and so on. Various risk analysis techniques have been discussed by Boranbayev *et al*. [4]. Their proposed web application with expert data storage serves as risk analysis assistance for even experts with absolutely no or less experience in the earlier stages of the development lifecycle.

Szwaczyk *et al*. [35] analyses several risk analysis methodologies in terms of criteria, which include dependency analysis, online reaction, historical data, risk prediction, scalability, performance, and output. Finally, the authors suggest the Bayesian Network (BN) model as the best-fit method, despite the drawbacks it has. Canavese *et al*. [6] propose an automated risk analysis technique to safeguard the software applications by assigning risk indices for the paths of attack. The qualitative risk analysis in [21] is performed using a Delphi-based technique named Risk Planning Poker, in which the risk analysis process is executed among the team members while preserving their anonymity. However, the absence of quantitative procedures for analyzing risks is a major drawback of the methodology.

The risk pyramid model does not produce the causal relationship that exists between the risks. It also handles only one risk at a time during risk resolution. The agile risk network model in [1] overcomes these limitations with the Risk Structure Matrix (RSM) in order to reveal the relationship between causes and impacts of a risk identified ahead. Using the model, the causal risks with the maximum number of effects identified by their outgoing edges can be eliminated or deducted. The primary advantage of this model is that this mapping of risks with respect to their causes and effects gives a clearcut idea about the importance of each risk while developing a project, and based upon the influence element, the risks are prioritized for resolution.

The qualitative risk model proposed in [2] focuses on efficient risk assessment and management without losing its agile nature and flexibility. The model puts forward 7 qualitative variables, which include 5 mitigation variables named reliability, availability, resilience, robustness, detectability, and 2 impact variables named severity and occurrence, where the agile practitioners gave a qualitative rating using a 10-point scale to each variable. The paper concludes that the classical qualitative risk models are hefty and not flexible, thus violating the flexibility requirement, which is one of the primary principles of agile. Their proposed model overcomes such limitations by offering a structured and flexible way of risk assessment without diminishing agility. However, the authors take into

consideration only four major hazards for assessment using the model.

The gamification-based risk analysis approach in [9] is dedicated to agile environments, irrespective of the methodology that the team uses. The approach engulfs existing techniques like risk boards, certain features of the Delphi technique, and gamification ideologies. These ideologies include I orientation, persuasive elements, learning orientation, Y generation adaptability, amusement factors, wellbeing-oriented, research-generating, and knowledge-based. The phases of software risk analysis include identifying and classifying the potential risks, deciding the priority of each and every identified risk using a variation of Fibonacci series-based risk planning poker, updating the risk board, and finally obtaining feedback using a questionnaire. With respect to the agile manifesto principles, the technique got affirmative feedback from 81.9% of participants, saying that it boosts the involvement of team members.

The Bayesian Network-based Causality Constraints (BNCC) risk analysis model in proposes a V-structure discovery algorithm. This is superior to many other algorithms, such as C4.5 and Naïve-Bayes, in terms of strong interpretation capability and sufficient prediction accuracy. The explicit knowledge proffered by the model indicating the causation between the risk factors and the end results of a project can immensely help in implementing an efficient way of risk analysis and the subsequent steps of planning the risks.

Han [12] uses in the study a three-layered neural network (input, hidden, and output layers) with a back propagation algorithm to increase the accuracy of prediction in order to identify if a project is risky or not. The model with an accuracy of 100% outperformed the previous logistic regression model, whose accuracy was 87.5%.

These literature works stirred us into making a snippet of a contribution to proposing a risk analysis approach in the zone of software project management.

3. Proposed Work

Risk analysis is the business of grading risks with respect to one or more parameters. The traditional methodology generally takes into account the probability and impact of the risks by computing their product, called risk criticality, which pertains to the individual significance of a risk, or with the assistance of a diagram. Designing risks as independent makes the process of assessing indirect, complex impacts impracticable. Moreover, when a single person performs risk analysis, it is pretty obvious that there are high possibilities for issues regarding bias and adaptability.

The Failure Mode and Effect Analysis (FMEA) gauges the failure mode consequences with respect to the experts' knowledge, and the entire accessible information is not used. This type of analysis makes use of uncomplicated checklists and is more appropriate for the designing phase of system development, and deeper analysis asks for even more complex techniques [38]. Risk analysis techniques such as Fault Tree Analysis (FTA), reliability block-diagram analysis executed in [2], and Event-Tree Analysis (ETA) gather their input from the outcome of FMEA, but they lack the capability to assess the deeper viewpoints of software risks [37].

The software reliability analysis methods are designed based on the number of eliminated bugs and aid in grasping the causes and effects of a system only to some extent. Hence, they cannot be considered useful tools for analyzing risks [37]. The Dynamic Flowgraph Methodology (DFM) does not confer any means to discover the causes of a failure [38]. The current automatic BN methodologies are not capable of distinguishing correlation from causation, and this needs to be addressed since causation provides immense support in finding the factors that have a great impact on the outcome of the project.

Risk analysis intends to appraise the detriments, frequency of usage, and likelihood of failure based on the shortcomings. To circumvent time and cost overruns during software development, it is necessary to quantitatively evaluate all risks pertaining to the high-level requirements so that it helps in making effective decisions while allotting resources to those requirements [29]. Quantitatively Analyzing Risks (QRA) is generally useful in evaluating the level of risks, and the related measurements are affixed with an uncertainty degree. A classic QRA methodology encompasses steps including identifying hazards, constructing models, estimating risks, and making effective decisions. A consummate framework may include sensitivity analysis and tolerance analysis in addition [41].

As per the ISO 12207 standard, which describes the processes carried out in the Software Development Life Cycle (SDLC), an objective can be related from several points of view, some of which include budget-related, health-related, safety-related, and related to the environment. There are chances for risk to occur at various levels of the enterprise, resulting from a wide range of both internal and external factors, which can have great control over the likelihood of a risk and its effects on the business objectives. It is essential to group and prioritize the extensive list of risks obtained to provide a clear-cut representation of which of the risks need to be handled and allocate the resources accordingly. Some of the risk assessment categories defined by the ISO 31010 standard include: intuition of the experts; auditing by the experts; easy scoring methods; weighted scores; a calculus of preferences; and probabilistic methods, depending on subjective data, historical data, and empirical research data. Stakeholders' participation in risk management throughout helps to increase communication and

knowledge regarding the risks, thus leading to efficient rectification actions [26].

The application of machine learning algorithms in the risk management process has gained tremendous popularity in recent years among software companies [34]. Recently, the trend of using machine learning algorithms for assessing risks has become quite popular. Especially supervised learning algorithms such as decision trees, naive bayes classifiers, neural networks, and SVMs are commonly used [34].

Software risks can be predicted in the following ways:

1. By classifying the level of a risk as high, low, medium, and so on, thus assisting in differentiating between different levels of risk.
2. By classifying whether a project is risky or not [12].

The main purpose of risk classification is to obtain a united perspective regarding a group of factors, which in turn can assist practitioners in discovering the group with the highest risks. Risk classification is an inexpensive methodology for risk analysis along with their root causes by categorizing risks of similar nature into a class [14]. Choosing the relevant features plays a vital role in the process of prediction, and the perfect number of features is capable of influencing the model's accuracy [15].

The initial phase in qualitatively analyzing risks is to brand them as per their origin. If multiple risks originate from a single point, addressing the risk source initiates an effective solution [20]. Quantitative analysis, characterized by a precise estimation of the risk impacts on project objectives, succeeds the qualitative process of analysis [20]. Subjective analysis by experts is commonly used in managing risks. However, it lacks replicability and suffers from obscureness. It is therefore important to develop more objective and decision-making assistance tools for managing risks.

Software risk management can be designed to concentrate on goal-based methodology in order to get rid of the risks entangled in the SDLC phases [27]. Since "risk" can create random changes in the project's anticipated outcome, it is impossible to put forward a single goal throughout the SDLC phases, and hence it is essential to customize each goal particular to each potential risk event [27].

The goal-based risk analysis methodology in [3] handles the risks occurring as a result of certain events through a few plans of action. This may include analyzing the cost of the solutions produced by the candidates, prioritizing the risks to distinguish the higher-priority risks from those of the lower ones, and analyzing the cost and risk so as to give a clear picture of the budget and the risks associated with each of the solutions provided. Based on this phenomenon and inspired by [32], we chose the goal values of our proposed 'performance goal affected' attribute to be 'time', 'cost', and 'quality'. The remarkable contributions of propelled us to propose an additional goal value, 'employee satisfaction'. Their findings acknowledge the fact that 'employee satisfaction' has the potential to influence the profit growth, sales growth, and return on investment of an organization in a positive manner. Hence, letting it be vulnerable will be a risk to the organization for sure.

According to the National Aeronautics and Space Administration (NASA), a good way of planning risks is "to ensure if the risk sources and impacts are known ahead and plan the more important risks initially". "Finding out the cause of a risk" is a requisite of the risk analysis phase as per the policies of the Software Engineering Institute of Carnegie Mellon University (CMU/SEI). Thus, we come to the conclusion that deep learning about the causes and consequences is critical in the risk planning phase.

The main objectives of the study are to answer the following research questions:

- How to form a categorical dataset for the purpose of risk analysis in an agile-specific environment.
- How to build a machine learning model with the proposed risk dataset with better performance in prediction.
- Does the Goal-driven Software development Risk Management (GSRM) framework attribute have more significance in risk analysis than the Agile Software Risk Identification (ASRI) framework attribute?

With this aim, the overall flowchart of the building process of the model is depicted in Figure 1. The detailed execution steps are described in further sections.

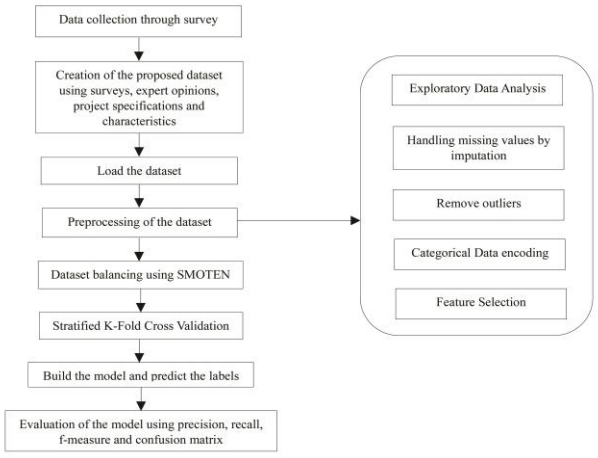


Figure 1. Building process of the machine learning model.

3.1. Risk-Based Data Collection

The proposed dataset, in order to predict the level of risk (low, medium, and high), is accumulated through various sources of data.

Among them, a survey was preferred to execute the initial phase of the risk analysis procedure. Inspired by

[32], we designed the questionnaire in such a way that the experts need to figure out each and every risk factor's probability, impact, and affected performance goal obtained through our previous research experiments on risk identification. Convenience sampling was applied for this purpose due to the insufficiency of samples. Both 'probability' and 'impact' of the risk were measured through 5-point Likert scales, and the 'performance goal affected' comprised four choices, including 'time,' 'cost,' 'quality,' and 'employee satisfaction.' A sample format of the questionnaire is shown in the appendix.

Since we had 217 risk factors from the previous risk identification phase, it is impossible to pack them into a single questionnaire, which might lead to a very poor response rate if executed. Hence, the idea of partitioning them into four different sets of questionnaires with the same objectives was accomplished. Survey 1 enlisted 51 risk factors, survey 2 enlisted 56 risk factors, and surveys 3 and 4 enlisted 55 risk factors each.

Agile projects pose unique issues due to its dynamic nature, frequent changes in project scope, and iterative procedures that are not normally addressed by standard risk management methodologies. Existing research on machine learning for risk categorization focuses on static or more predictable software environments, ignoring the flexibility and rapid change in agile projects. This gap highlights the need for specialized algorithms and risk models that can adapt to the fluidity and evolving nature of agile workflows, which is why machine learning algorithms such as SVM for high-dimensional data and decision trees for interpretability are especially valuable, as they have the potential to handle complex, evolving risk factors and provide real-time, actionable insights to agile teams.

As the population is unknown, we collected responses for the questionnaire from 120 samples (30 samples for each questionnaire to produce statistically significant results) from 9 software Small and Medium-sized Enterprises (SMEs) in Kanyakumari District, Tamil Nadu, India. The enterprises follow either traditional or hybrid methodologies, and the demographic information of the samples is depicted in Figures 2 and 3.

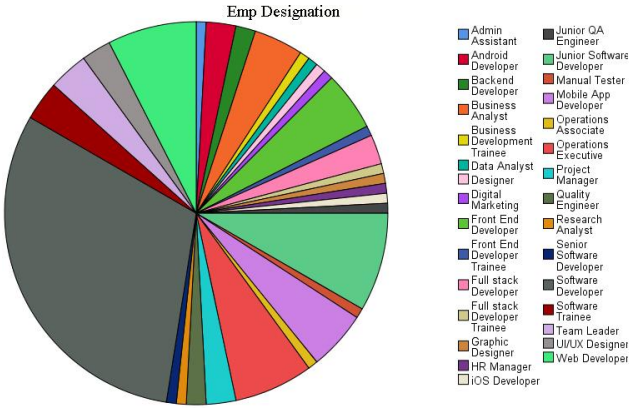


Figure 2. Pie chart of the survey participants.

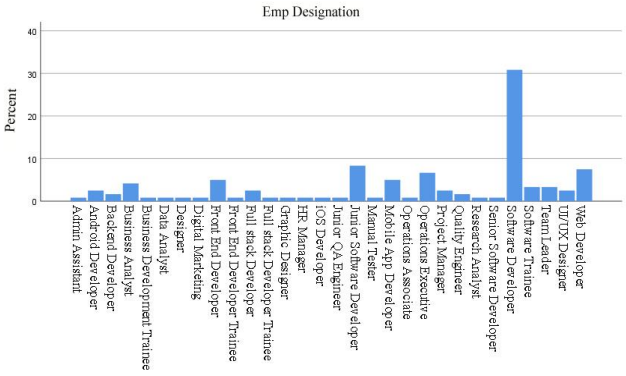


Figure 3. Bar chart representing the participants involved in the survey.

To analyze the survey data, International Business Machines Corporation-Statistical Package for the Social Sciences (IBM-SPSS) version 26 is used. The reliability analysis of the conducted surveys is manifested by the statistical test for internal consistency reliability using the Cronbach's Alpha test. It is evident that all the surveys can be declared reliable since their Cronbach's Alpha values are found to be 0.936, 0.893, 0.922, and 0.892, respectively, and are greater than 0.6.

3.2. Creation of Dataset

The dataset for this study was gathered from small and SME engaging in ASD projects [40], at each level of the agile approach, the essential risk management procedures are implemented in accordance with the ISO 31000 standard.

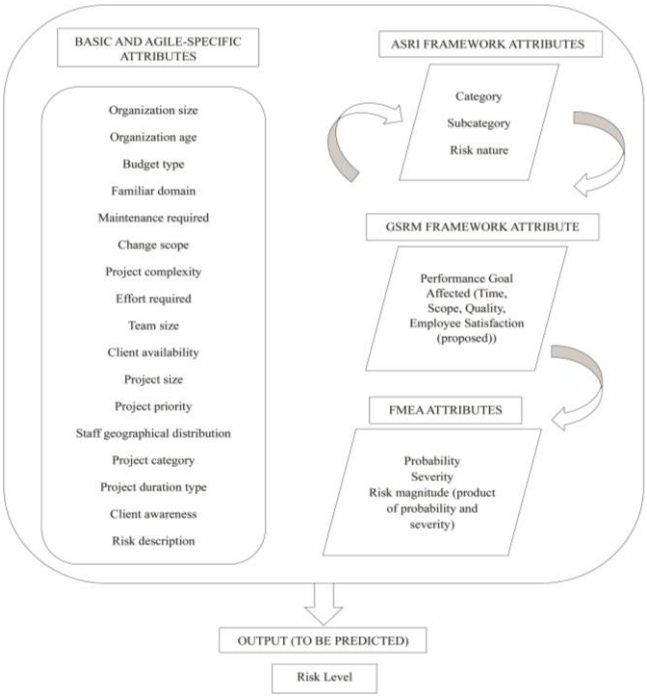


Figure 4. Attributes template of the risk dataset.

Other variables of the proposed dataset are added to the framework from various literature sources and from the project specifications. The requisite data is obtained from the opinions of experts and through analyzing the

project characteristics. The framework proposed in Figure 4 is fabricated as an assembly of several attributes described and further details about the dataset are given in Table 1.

Table 1. Dataset template for the risk attributes.

Attributes	Data type	Value sets	Sources
Organization size	Ordinal	Small, Medium	[36]
Organization age	Ordinal	3-5 years, >10 years	Project characteristics
Budget type	Ordinal	Low, Medium, High	[13]
Familiar domain	Nominal	Yes, No	[8]
Maintenance required	Nominal	Yes, No	Project characteristics
Change scope	Ordinal	Low, Medium, High	[13]
Project complexity	Ordinal	Low, High	[13, 24]
Effort required	Ordinal	Medium, High	[5, 23, 36]
Team size	Ordinal	Medium, High	[7, 10, 11, 22, 36]
Client availability	Ordinal	Low, Medium, High	[24, 25, 28, 30]
Project size	Ordinal	Low, Medium, High	[11 13, 22]
Project priority	Ordinal	Low, Medium, High	[22]
Staff geographical distribution	Ordinal	Low, Medium, High	[22]
Project category	Nominal	Safety and security, Subcontract, Banking and financial	[31]
Project duration type	Ordinal	Short, Medium	[10]
Client awareness	Ordinal	Low, High	[24]
Risk description	String	Previously identified risks	[16]
Category	Nominal	Human, Organizational, Technical, Non-Technical, Capabilities	Our ASRI framework [16, 40],
Subcategory	Nominal	Personnel, Customer, Team, Other stakeholders, Project management, Communication, Coordination, Culture and behavior, Requirements, Design, Implementation, Testing, Documentation, Operation and maintenance, Process, Security, Environmental, Legal, Marketing, Technology, Skills and Experience, Resources	Our ASRI framework [40]
Risk nature	Nominal	Over-doing, Under-doing, Mistakes, Concept risks, Changes, Differences, Difficulties, Dependency, Conflicts, Issues, Challenges	Our ASRI framework [40]
Performance goal affected	Nominal	Time, Cost, Quality, Employee satisfaction (proposed)	(GSRM based attribute) [16]
Probability	Ordinal	Very low, Low, Medium, High, Very high	[31]
Severity	Ordinal	Very low, Low, Medium, High, Very high	[31]
Risk magnitude	Ordinal	Negligible, Very low, Low, Medium, High, Very high, Extreme	Product of probability and severity [31]
Risk level (to be predicted)	Nominal	Low, Medium, High	[31]

3.3. Data Preprocessing

Prior to building a machine learning model, it is vital to go through the preprocessing phase. This is due to the fact that raw data may encompass incompleteness, inconsistency, outliers, and mistakes. Neglecting such factors may result in a poor model with inefficiency and may not be able to predict the labels with precision. Keeping this in mind, we make the following changes to our dataset during the preprocessing phase:

1. Exploratory Data Analysis (EDA): while preparing the data for model building, it is important to deeply analyze the dataset since an absence of effective data handling may result in unexpected results and serious issues in the model. From Table 1, it is evident that our dataset is of a categorical nature with both nominal and ordinal types of attributes and a string attribute, which is again not of a numerical type. Our dataset is also an imbalanced one, with an unequal ratio of instances for all three classes.
2. Handling Missing Values: partial data can never produce better results or performance. As a consequence, it is necessary to detect if there are any missing values in the dataset and treat them accordingly as per rather than deleting the particular record or attribute. The dataset analysis ensures that it is free from missing values.
3. Remove Outliers: the presence of outliers in the dataset can negatively affect the performance of the model to a great extent, and hence it is important to figure out and exclude them. However, since our dataset is in categorical format, all values are within the range, and there is no chance for an outlier to occur. However, less frequent categories in each attribute column can also be considered outliers since their presence has a high chance of producing a negative impact. With this motive, we removed certain records with a minimal number of categories from the dataset and finally ended up with 164 records for the experimental purpose.
4. Categorical Data Encoding: prior to building a machine learning model, the categorical data must be encoded into a numerical format since it is the only understandable format for machine learning models. As a result, label encoding and one hot encoding were used for this purpose, depending on the context.
5. Feature Selection: among the list of attributes in Table 1, the independent attributes that have a significant impact on the dependent variable output must be chosen so as to discard the weaker attributes, thus eliminating noise in the data. The statistical correlation analysis is suitable for this purpose, and since all our attributes are of the categorical data type, we found the Chi-Square correlation to be the most appropriate solution. The results of the Chi-Square correlation graph are plotted in Figure 5. The higher the chi-value, the higher the importance, and the lower the chi-value, the lower the importance. However, the experiments aim to find the ramifications of the goal-based attributes and the

ASRI attributes on the performance of the model and are executed as such. It is to be noted that the highly inter-related attributes need to be detected and filtered prior to the feature selection process. For instance, the 'category' and 'subcategory' attributes are related to each other, and exerting the two in the experiments may lead to the generation of fallacious results. Hence, it is vital to exclude one among them based on their impact on the model's performance. In a similar fashion, the 'risk magnitude' attribute replaces the 'probability' and 'severity' attributes since 'risk magnitude' is nothing but the product of the two.

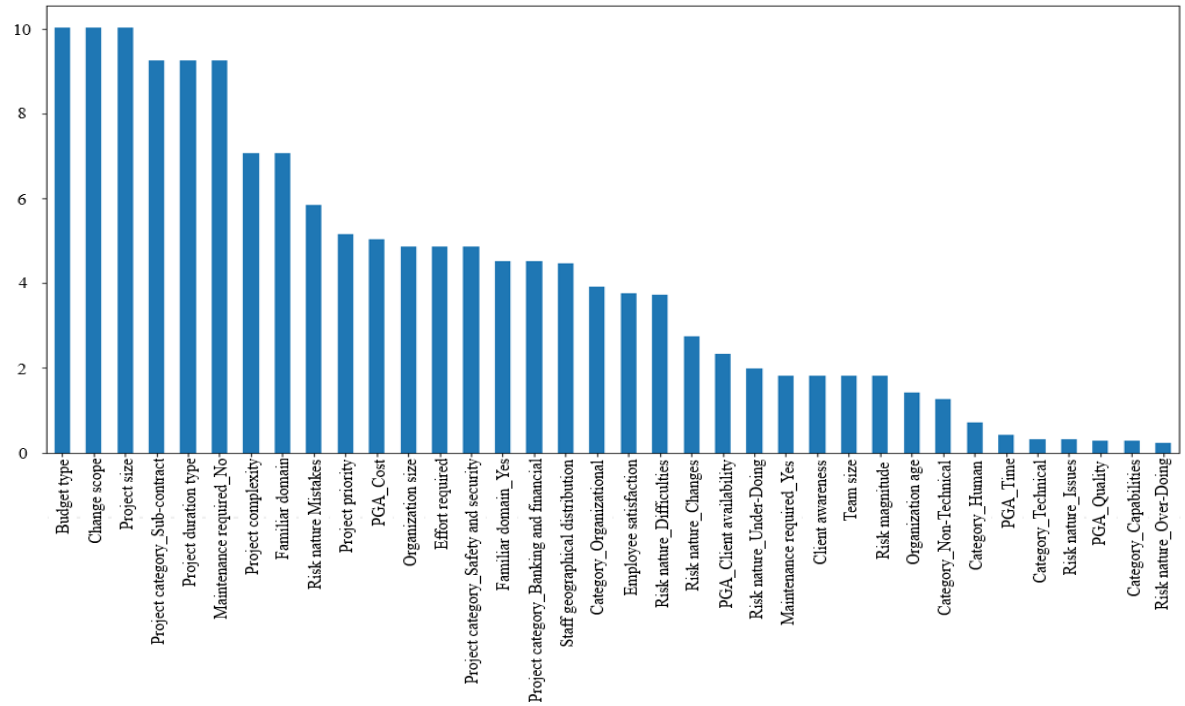


Figure 5. Bar chart representing the results of Chi-square test.

Furthermore, no feature scaling is needed since all the attributes are categorical and have an almost equal range of input values, and there is less chance for the model to prioritize one attribute over another. As mentioned already, since the dataset is imbalanced, there is a high chance of producing biased results in the prediction. To overcome this, we have used Synthetic Minority Over-sampling TEchnique for Nominal features (SMOTEN), a variant of Synthetic Minority Over-sampling TEchnique (SMOTE), specifically used for categorical attributes, to balance the dataset. In addition, the stratified k-fold cross-validation technique was used in the experimental analysis to validate the results using different iterations to improve the performance of the model.

3.4. Building the Machine Learning Model

This section encompasses the pros and cons of the machine learning classifiers used in the experimental study. Based on their capacity to manage classification tasks efficiently and offer insightful information on software risk prediction, logistic regression, decision tree, SVM, naïve bayes, and K-Nearest Neighbor (KNN) were chosen. Initially, SVM and logistic regression were used due to their robustness in high-dimensional and structured data settings. While SVM is renowned for its capacity to recognize optimal decision boundaries, which aids in the efficient classification of risks, particularly in complex datasets, logistic regression provides a probabilistic framework that makes it appropriate for comprehending and measuring the likelihood of various risk levels. Second, SVM was chosen for its capacity to deal with high-dimensional data, which is typical in software risk analysis when several risk factors interact. SVM is excellent at determining optimal decision boundaries and performs well in complicated datasets where a clear margin of separation is required for effective risk categorization. Finally, decision trees were incorporated due to their interpretability and capacity to represent non-linear relationships. decision trees provide explicit, understandable decision rules that can assist agile teams in efficiently identifying and managing risks in complicated contexts. Prior to the experimental analysis, let us discuss the models in detail.

1. Logistic Regression: Logistic regression is a popular machine learning algorithm for classification problems that predicts labels using a sigmoid function by mapping a real number to a probability value between zero and one [13]. Logistic regression and naïve bayes are the broadly preferable supervised parametric classification algorithms. Logistic regression is best suited for overfitting situations where the features are larger than the observations.

2. SVM: SVM is a linear classification algorithm that generates hyperplanes for making decisions by making use of statistical learning theory. The algorithm also offers better generalization and performs with equivalent accuracy even when there is sparse data. It also accomplishes better when the sample sizes are small.
3. Naïve Bayes: naïve bayes is yet another probabilistic simple classification algorithm, assuming that all the features are independent of one another. However, it also works well when the features are interconnected. Similar to logistic regression, naïve bayes is also expected to shine with a small sample size.
4. Decision Tree: the decision trees are more comprehensive when compared to neural networks. However, it is quite difficult for a normal user to learn the underlying concepts behind the decisions taken using a decision tree. Furthermore, when the complexity among the relationships increases, the computational complexity too increases, thus causing overhead.
5. KNNs: this algorithm works on the principle that data points that are close to each other belong to a particular class. It has the advantage of being robust with noisy data. However, when the dimensionality is increased, the computational complexity will also increase. In addition, deciding the number of nearest neighbors is quite challenging.

3.5. Experimental Analysis

The Python language was utilized to build the machine learning models with the assistance of requisite packages such as Pandas, NumPy, Scikit-Learn, and so on. As evaluation parameters, metrics such as precision, recall, F-measure, and confusion matrix are given higher preference in our study since the accuracy metrics alone will not make sense when the dataset is imbalanced [34]. Since we have applied the SMOTEN technique for balancing the dataset, the accuracy metrics can be considered as well to estimate the model performance. With the below metrics, the performance of all the machine learning models with different sets of attributes was evaluated and compared.

- **Accuracy:** Accuracy is an important metric when it comes to evaluating the performance of a machine learning model [13]. It is the ratio between the correctly classified instances and the total number of instances. This can be represented as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

- **Recall**: The recall metric, also known as True Positive Rate (TPR), is a measure of correctly classified positive instances with respect to the total number of positive instances [13], and is given by the equation,

$$Recall\ (TPR) = \frac{TP}{TP + TN} \quad (2)$$

- **Precision**: the idea behind using the 'precision' metric is to find out the number of actual positive instances from the totally predicted positive instances [13], which can be calculated by,

$$Precision = \frac{TP}{TP + FP} \quad (3)$$

In the above equations, TP refers to 'True Positive', TN refers to 'True Negative', FP indicates 'False Positive', and FN refers to 'False Negative' instances.

- **F-Measure**: F-measure is nothing but the harmonic means of the 'precision' and 'recall' metrics, and is given by the equation,

$$F - Measure = \frac{2 * Precision * Recall}{Precision + Recall} \quad (4)$$

- **Confusion Matrix**: this matrix provides an optimal solution for classification problems [13], since it provides insights about the correctly and incorrectly classified instances through a matrix format.

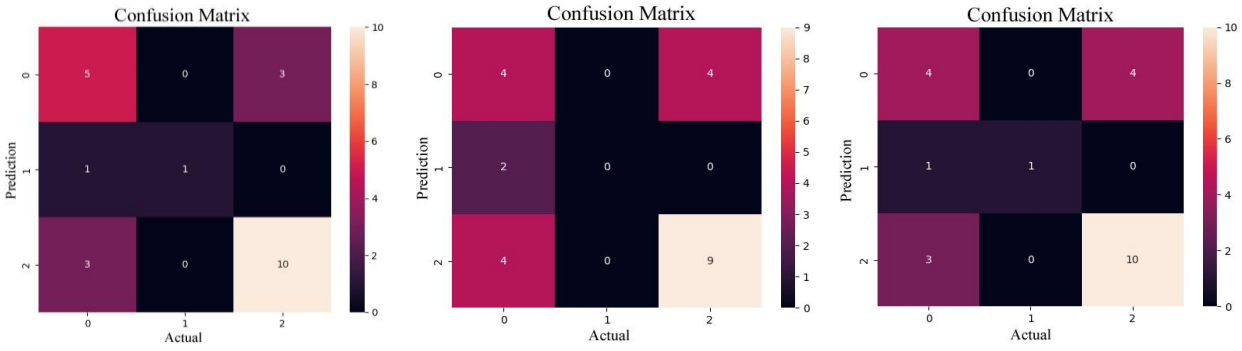
4. Finding and Results

The proposed dataset is validated using the basic machine learning models to solve the classification problem, and the summary of the findings is revealed in Table 2. The outcome of three different combinations of our risk dataset attributes is compared here. From the obtained results, the omission of the proposed 'risk nature' attribute from our risk analysis framework has a highly negative impact on the performance (testing accuracy) of the logistic regression, decision tree, and KNN models compared to the omission of the existing GSRM attribute, 'performance goal affected'. This limited dataset makes it difficult to generalize the results to SMEs operating in different regions or industries, potentially reducing the model's effectiveness in diverse settings. Despite these limitations, the findings provide practical insights for SMEs, helping them prioritize key risk factors and allocate resources more effectively. By understanding which factors most influence business risks, SMEs can make data-driven decisions to improve resilience and long-term sustainability. This is clearly evident by the decrease in precision, recall, F1-score, and accuracy (both training and testing). Thus, we can come to the conclusion that 'risk nature' is an important attribute in performing software risk analysis using machine learning models. Also, logistic regression with all our risk dataset attributes is found to be the optimal algorithm for predicting software risk levels with the highest precision of 0.71, recall of 0.70, F1-score of 0.70, training accuracy of 73%, and testing accuracy of 70%. Though it is a satisfying result, improving the number of records is a strong suggestion for performance escalation.

Table 2. Summary of the experimental analysis.

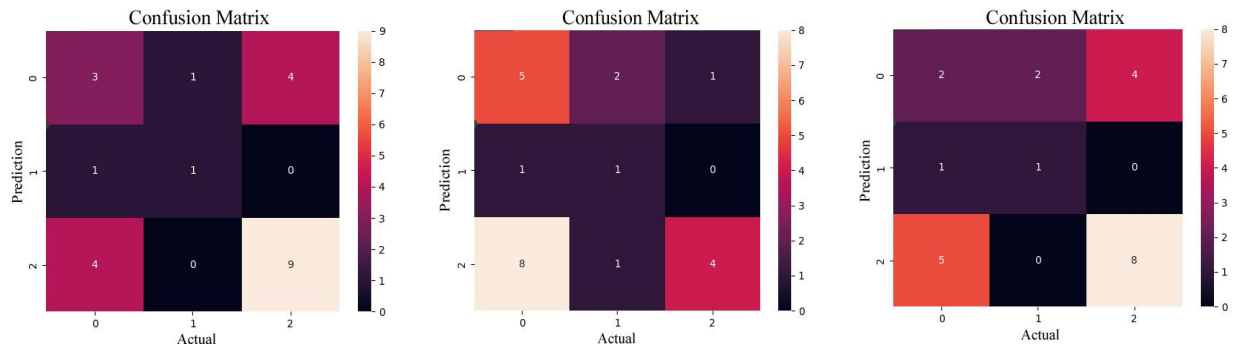
Model name	Without feature selection	Without the ASRI framework attribute 'risk nature'	Without the GSRM framework attribute 'performance goal affected'
Logistic regression (7-fold cross validation)	0.71 (') 0.70 (recall) 0.70 (F1-score) 73% (training accuracy) 70% (testing accuracy)	0.53 (precision) 0.57 (recall) 0.55 (F1-score) 73% (training accuracy) 56% (testing accuracy)	0.66 (precision) 0.65 (recall) 0.65 (F1-score) 71% (training accuracy) 65% (testing accuracy)
Decision tree (7-fold cross validation)	0.57 (precision) 0.57 (recall) 0.57 (F1-score) 90% (training accuracy) 57% (testing accuracy)	0.60 (precision) 0.43 (recall) 0.44 (F1-score) 90% (training accuracy) 43% (testing accuracy)	0.49 (precision) 0.48 (recall) 0.48 (F1-score) 91% (training accuracy) 48% (testing accuracy)
SVM (7-fold cross validation)	0.66 (precision) 0.65 (recall) 0.65 (F1-score) 69% (training accuracy) 65% (testing accuracy)	0.65 (precision) 0.65 (recall) 0.64 (F1-score) 65% (training accuracy) 65% (testing accuracy)	0.65 (precision) 0.65 (recall) 0.64 (F1-score) 64% (training accuracy) 65% (testing accuracy)
Naïve bayes (2-fold cross validation)	0.58 (precision) 0.60 (recall) 0.58 (F1-score) 66% (training accuracy) 59% (testing accuracy)	0.58 (precision) 0.59 (recall) 0.57 (F1-score) 65% (training accuracy) 59% (testing accuracy)	0.56 (precision) 0.56 (recall) 0.54 (F1-score) 65% (training accuracy) 56% (testing accuracy)
KNN (7-fold cross validation)	0.57 (precision) 0.52 (recall) 0.53 (F1-score) 72% (training accuracy) 52% (testing accuracy)	0.54 (precision) 0.52 (recall) 0.53 (F1-score) 77% (training accuracy) 52% (testing accuracy)	0.60 (precision) 0.57 (recall) 0.57 (F1-score) 76% (training accuracy) 57% (testing accuracy)

The corresponding confusion matrix visualizations for the prediction outcomes of the machine learning models are depicted in Figures 6, 7, 8, 9, and 10. One key finding of the study is that logistic regression outperformed other models, but the paper does not fully explore the reasons behind this result. Logistic regression is often preferred for its simplicity, interpretability, and effectiveness in handling binary classification problems, which may explain its superior performance in this context. Unlike more complex models, it provides clear insights into the relationship between risk factors and outcomes, making it a practical choice for SMEs seeking to understand and mitigate business risks. However, the study does not extensively discuss what this means for practical risk management in SMEs. Given its interpretability, SMEs can use logistic regression to identify the most influential risk factors and make data-driven decisions to allocate resources more efficiently, ultimately improving their resilience and long-term stability.



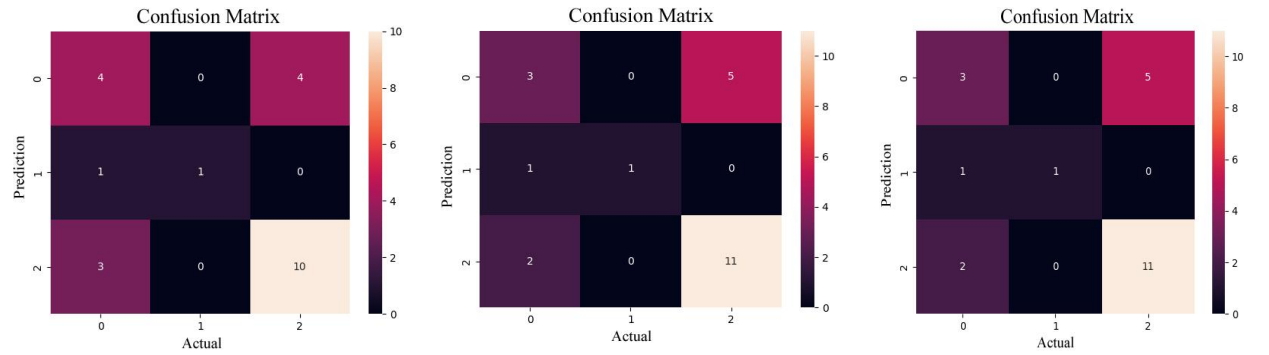
a) Without feature selection. b) Without 'risk nature' attribute. c) Without GSRM attribute.

Figure 6. Logistic regression models.



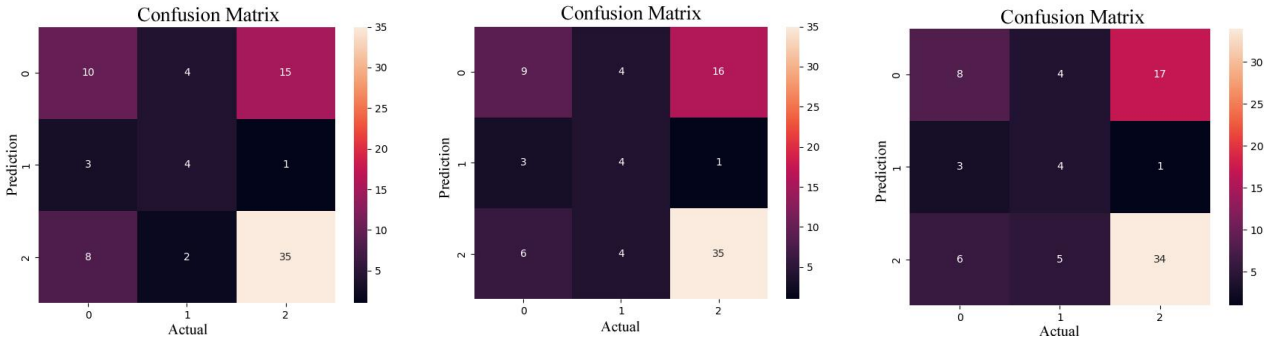
a) Without feature selection. b) Without 'risk nature' attribute. c) Without GSRM attribute.

Figure 7. Decision tree models.



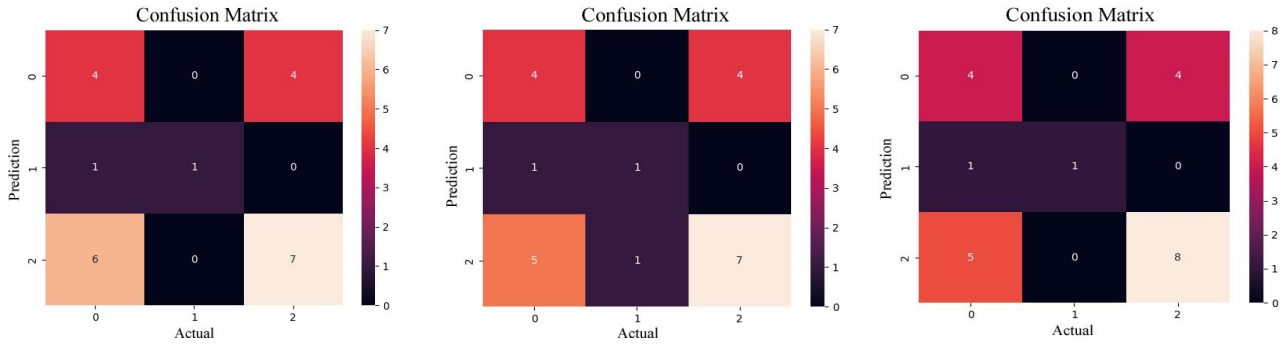
a) Without feature selection. b) Without 'risk nature' attribute. c) Without GSRM attribute.

Figure 8. SVM models.



a) Without feature selection. b) Without 'risk nature' attribute. c) Without GSRM attribute.

Figure 9. Naïve bayes models.



a) Without feature selection. b) Without 'risk nature' attribute. c) Without GSRM attribute.

Figure 10. KNN models.

5. Limitations and Threats to Validity

This section deals with some of the threats to the validity of the study that we came across. A major internal threat to validity is the data imbalance, which can influence the model prediction results despite the use of the SMOTEN technique. However, this drawback is limited to some extent by the application of the Stratified K-Fold cross-validation method. A major internal threat to validity is data imbalance, which can influence model prediction results despite the use of the SMOTEN technique. While SMOTEN helps address class imbalance, it does not completely eliminate its effects, and oversampling may introduce the risk of overfitting. However, the application of Stratified K-Fold cross-validation mitigates this issue to some extent by ensuring a balanced class distribution across training and validation sets.

Another significant threat is related to the generalization of the proposed model, which affects its external validity. Since the survey data were collected through convenience sampling from small and medium enterprises within a single district due to time and cost constraints, the findings may not be broadly applicable. This limited dataset makes it difficult to generalize the results to SMEs operating in different regions or industries, potentially reducing the model's effectiveness in diverse settings.

6. Conclusions

This study proposed the application of machine learning techniques for software risk classification in ASD environments. The experimental analysis demonstrated that among the five implemented algorithms, logistic regression 70% accuracy and SVM 65% accuracy performed the best in predicting software risks. These findings suggest that machine learning-based risk analysis can effectively assist in identifying and prioritizing software risks, thereby contributing to

improved risk management strategies in ASD environments. The dataset used may not fully represent the diverse risk factors encountered in different agile projects, particularly in SMEs with limited resources. Additionally, the study relied on basic machine learning models, which, while effective, could be improved using advanced ensemble or deep learning techniques. Future research should focus on expanding the dataset, incorporating more sophisticated machine learning models, and improving model interpretability to ensure that risk predictions are both accurate and actionable.

Acknowledgment

The author would like to express his heartfelt gratitude to the supervisor for his guidance and unwavering support during this research for his guidance and support.

References

[1] Andrat H. and Jaswal S., "An Alternative Approach for Risk Assessment in Scrum," *in Proceedings of the International Conference on Computing and Network Communications*, Trivandrum, pp. 535-539, 2015. DOI:10.1109/CoCoNet.2015.7411239

[2] Anes V., Abreu A., and Santos R., "A New Risk Assessment Approach for Agile Projects," *in Proceedings of the International Young Engineers Forum (YEF-ECE)*, Costa da Caparica, pp. 67-72, 2020. DOI:10.1109/YEF-ECE49388.2020.9171808

[3] Bhukya S. and Pabboju S., "Software Engineering: Risk Features in Requirement Engineering," *Cluster Computing*, vol. 22, pp. 14789-14801, 2019. https://doi.org/10.1007/s10586-018-2417-3

[4] Boranbayev A., Boranbayev S., Nurusheva A., Yersakhanov K., and Seitkulov Y., "A Software System for Risk Management of Information Systems," *in Proceedings of the 12th International Conference on Application of Information and Communication Technologies*, Almaty, pp. 1-6, 2018. DOI:10.1109/ICAICT.2018.8747045

[5] Bumbary K., "Using Velocity, Acceleration, and Jerk to Manage Agile Schedule Risk," *in Proceedings of the International Conference on Information Systems Engineering*, Los Angeles, pp. 73-80, 2016. DOI:10.1109/ICISE.2016.21

[6] Canavese D., Regano L., Basile C., Coppens B., and De Sutter B., "Man-at-the-End Software Protection as a Risk Analysis Process," *arXiv Preprint*, vol. arXiv:2011.07269, pp. 1-25, 2022. https://doi.org/10.48550/arXiv.2011.07269

[7] Carvallo J., Oktaba H., and Hernandez E., "Risk Assessment Forum," *in Proceedings of the 6th International Conference in Software Engineering Research and Innovation (CONISOFT)*, San Luis Potosi, pp. 160-164, 2018. DOI:10.1109/CONISOFT.2018.8645949

[8] Filippetto A., Lima R., and Barbosa J., "A Risk Prediction Model for Software Project Management Based on Similarity Analysis of Context Histories," *Information and Software Technology*, vol. 131, pp. 106497, 2021. https://doi.org/10.1016/j.infsof.2020.106497

[9] Gasca-Hurtado G., Gomez-Alvarez M., Munoz M., and Pena A., "A Gamified Proposal for Software Risk Analysis in Agile Methodologies," *in Proceedings of the 26th European Conference on Edinburgh Systems, Software and Services Process Improvement*, Edinburgh, pp. 272-285, 2019. https://doi.org/10.1007/978-3-030-28005-5_21

[10] Ghobadi S. and Mathiassen L., "A Model for Assessing and Mitigating Knowledge Sharing Risks in Agile Software Development," *Information Systems Journal*, vol. 27, no. 6, pp. 1-33, 2016. file:///C:/Users/user/Downloads/ISJ2-May2016.pdf

[11] González-Cruz T., Botella-Carrubi D., and Martinez-Fuentes C., "The Effect of Firm Complexity and Founding Team Size on Agile Internal Communication in Startups," *International Entrepreneurship and Management Journal*, vol. 16, pp. 1101-1121, 2020. https://doi.org/10.1007/s11365-019-00633-1

[12] Han W., "Discriminating Risky Software Project Using Neural Networks," *Computer Standards and Interfaces*, vol. 40, pp. 15-22, 2015. https://doi.org/10.1016/j.csi.2015.01.001

[13] Hassan H., Abdel-Fattah M., and Ghoneim A., "Risk Prediction Applied to Global Software Development using Machine Learning Methods," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 9, pp. 111-120, 2022. DOI: 10.14569/IJACSA.2022.0130913

[14] Hoodat H. and Rashidi H., "Classification and Analysis of Risks in Software Engineering," *International Journal of Computer and Information Engineering*, vol. 56, pp. 446-452, 2009. https://www.academia.edu/573832/Classification_and_Analysis_of_Risks_in_Software_Engineering

[15] Ingale S., Paraye M., and Ambawade D., "Enhancing Multi-Step Attack Prediction Using Hidden Markov Model and Naive Bayes," *in Proceedings of the International Conference on Electronics and Sustainable Communication Systems*, Coimbatore, pp. 36-44, 2020. DOI:10.1109/ICESC48915.2020.9155895

[16] Islam S., Mouratidis H., and Weippl E., "An Empirical Study on the Implementation and

Evaluation of a Goal-Driven Software Development Risk Management Model," *Information and Software Technology*, vol. 56, no. 2, pp. 117-133, 2014. https://doi.org/10.1016/j.infsof.2013.06.003

[17] Kalluri R., "A Human Factors Study of Risk Management of Complex Agile Scrum Projects in Large Enterprises," *International Journal of Business and Management Studies*, vol. 3, no. 8, pp. 38-44, 2022. DOI:10.56734/ijbms.v3n8a6

[18] Khan M., Mirza A., and Saleem I., "Software Risk Analysis with the Use of Classification Techniques: A Review," *Engineering, Technology and Applied Science Research*, vol. 10, no. 3, pp. 5678-5682, 2020. https://doi.org/10.48084/etasr.3440

[19] Kremljak Z. and Kafol C., "Types of Risk in a System Engineering Environment and Software Tools for Risk Analysis," *Procedia Engineering*, vol. 69, pp. 177-183, 2014. https://doi.org/10.1016/j.proeng.2014.02.218

[20] Linh N., Hung P., Diep V., and Tung T., "Risk Management in Projects Based on Open-Source Software," *in Proceedings of the 8th International Conference on Software and Computer Applications*, Penang, pp. 178-183, 2019. https://doi.org/10.1145/3316615.3316648

[21] Lopes S., Souza R., Contessoto A., Oliveira A., and Braga R., "A Risk Management Framework for Scrum Projects," *in Proceedings of the 23rd International Conference on Enterprise Information Systems*, Virtual, pp. 30-40, 2021. DOI:10.5220/0010448300300040

[22] Machado J. and Do Lago Pereira S., "Automatic Risk Identification in Software Projects: An Approach Based on Inductive Learning," *Intelligent Information Management*, vol. 4, no. 5, pp. 291-295, 2012. http://dx.doi.org/10.4236/iim.2012.425041

[23] Marques R., Costa G., Mira da Silva M., Goncalves D., and Goncalves P., "A Gamification Solution for Improving Scrum Adoption," *Empirical Software Engineering*, vol. 25, no. 4, pp. 2583-2629, 2020. https://doi.org/10.1007/s10664-020-09816-9

[24] Molokken-Ostvold K. and Furulund K., "The Relationship between Customer Collaboration and Software Project Overruns," *in Proceedings of the Agile*, Washington (DC), pp. 72-83, 2007. DOI:10.1109/AGILE.2007.57

[25] Nikiforova O., Babris K., and Kristapsons J., "Survey on Risk Classification in Agile Software Development Projects in Latvia," *Applied Computer Systems*, vol. 25, no. 2, pp. 105-116, 2020. https://doi.org/10.2478/acss-2020-0012

[26] Oehmen J., Gunther A., Herrmann J., Schulte J., and Willumsen P., *Proceedings of the Design Society: DESIGN Conference*, Cambridge University Press, 2020. https://doi.org/10.1017/dsd.2020.27

[27] Patil S. and Ade R., *Information Systems Design and Intelligent Applications*: *Advances in Intelligent Systems and Computing*, Springer, 2015. https://doi.org/10.1007/978-81-322-2247-7_78

[28] Perera C. and Perera I., "The Impact of Client Involvement towards Agile Project Success in Sri Lankan Software Industry," *in Proceedings of the Moratuwa Engineering Research Conference*, Moratuwa, pp. 279-284, 2019. DOI:10.1109/MERCon.2019.8818800

[29] Ray M. and Mohapatra D., "Risk Analysis: A Guiding Force in the Improvement of Testing," *IET Software*, vol. 7, no. 1, pp. 29-46, 2013. https://doi.org/10.1049/iet-sen.2011.0081

[30] Salazar-Salazar G., Mora M., Duran-Limon H., Alvarez-Rodriguez F., and Munoz-Zavala A., "Review of Agile SDLC for Big Data Analytics Systems in the Context of Small Organizations Using Scrum-XP," *The International Arab Journal of Information Technology*, vol. 21, no. 6, pp. 1089-1110, 2024. DOI: 10.34028/iajit/21/6/12

[31] Shaukat Z., Naseem R., and Zubair M., "A Dataset for Software Requirements Risk Prediction," *in Proceedings of the IEEE International Conference on Computational Science and Engineering*, Bucharest, pp. 112-118, 2018. DOI:10.1109/CSE.2018.00022

[32] Shrivastava S. and Rathod U., "A Goal-Driven Risk Management Approach for Distributed Agile Development Projects," *Australasian Journal of Information Systems*, vol. 23, pp. 1-30, 2019. https://doi.org/10.3127/ajis.v23i0.1843

[33] Sinha R., Shameem M., and Kumar C., "SWOT: Strength, Weaknesses, Opportunities, and Threats for Scaling Agile Methods in Global Software Development," *in Proceedings of the 13th Innovations in Software Engineering Conference on Formerly Known as India Software Engineering Conference*, Jabalpur, pp. 1-10, 2020. https://doi.org/10.1145/3385032.3385037

[34] Sousa A., Faria J., and Mendes-Moreira J., "An Analysis of the State of the Art of Machine Learning for Risk Assessment in Software Projects," *in Proceedings of the 33rd International Conference on Software Engineering and Knowledge Engineering*, Pittsburgh, pp. 217-222, 2021. https://ksiresearchorg.ipage.com/seke/Proceedings/seke/SEKE2021_Proceedings.pdf

[35] Szwaczyk S., Wrona K., and Amanowicz M., "Applicability of Risk Analysis Methods to Risk-Aware Routing in Software-Defined Networks," *in Proceedings of the International Conference on Military Communications and Information Systems*, Warsaw, pp. 1-7, 2018.

DOI:10.1109/ICMCIS.2018.8398688

[36] Tavares B., Da Silva C., and De Souza A., "Risk Management Analysis in Scrum Software Projects," *International Transactions in Operational Research*, vol. 26, no. 5, pp. 1884-1905, 2019. https://doi.org/10.1111/itor.12401

[37] Thieme C., Mosleh A., Utne I., and Hegde J., "Incorporating Software Failure in Risk Analysis-Part 1: Software Functional Failure Mode Classification," *Reliability Engineering and System Safety*, vol. 197, pp. 106803, 2020. https://doi.org/10.1016/j.ress.2020.106803

[38] Thieme C., Mosleh A., Utne I., and Hegde J., "Incorporating Software Failure in Risk Analysis-Part 2: Risk Modeling Process and Case Study," *Reliability Engineering and System Safety*, vol. 198, pp. 106804, 2020. https://doi.org/10.1016/j.ress.2020.106804

[39] Thool A. and Brown C., "Securing Agile: Assessing the Impact of Security Activities on Agile Development," *in Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, Salerno, pp. 668-678, 2024. https://doi.org/10.1145/3661167.3661280

[40] Zahedi M., Kashanaki A., and Farahani E., "Risk Management Framework in Agile Software Development Methodology," *International Journal of Electrical and Computer Engineering*, vol. 13, no. 4, pp. 4379-4387, 2023. https://ijece.iaescore.com/index.php/IJECE/article/view/29151/16756

[41] Zhang L., Wang Y., and Wu X., "Cluster-based Information Fusion for Probabilistic Risk Analysis in Complex Projects under Uncertainty," *Applied Soft Computing*, vol. 104, pp. 107189, 2021. https://doi.org/10.1016/j.asoc.2021.107189

Ayesha Ziana Mohamed B.Sc. graduate in Computer Science under Manonmaniam Sundaranar University, Tirunelveli. Post graduated in Master of Computer Applications under Anna University, Chennai. A couple of years of working experience as an Assistant Professor in the Department of Computer Science, including a year as a Youth Red Cross coordinator at Infant Jesus College of Arts and Science for Women, Mulagumoodu, India. Passed the SET and NET eligibility tests for Assistant Professor conducted by the University Grants Commission of India. Currently a Full-Time Research Scholar in the Department of Computer Science at the Noorul Islam Centre for Higher Education, Kumaracoil, Thuckalay, Kanyakumari District, Tamil Nadu, India. Her main research areas of interest include Software Engineering, and Machine Learning.

Charles Jebapillai has postgraduate degrees in M.Sc.(CS) and M.Tech.(IT) from Bharathidasan University, Tamil Nadu and a Ph.D(CSE). from NICHE, Kumaracoil, Tamil Nadu. He is currently the Director of Student Affairs and an Associate Professor in the Department of Software Engineering at the Noorul Islam Centre for Higher Education, Kumaracoil, Thuckalay, Kanyakumari District, Tamil Nadu, India. He was a former Head of the Department and has more than 26 years of teaching experience and many years of different administrative experience, including 4 years on the Board of Management in the same institution. His main research areas of interest include Software Engineering, and Data Mining. He has guided many M.Phil. and Ph.D. research scholars and has contributed to the research industry with a significant number of publications.