Risk Prediction Applied to Global Software Development using Machine Learning Methods

Hossam Hassan[1], Manal A. Abdel-Fattah[2], Amr Ghoneim[3]
Information Systems Department, Helwan University, Egypt[1, 2]
Computer Sciences Department, Helwan University, Egypt[3]

***Abstract*—Software companies aim to develop high-quality software projects with the best global resources at the best cost. To achieve this global software development (GSD), an approach should be used which adopts work on projects across multiple distributed locations, and this is also known as distributed development. When companies attempt to implement GSD, they face numerous challenges owing to the nature of GSD and its differences from traditional methods. The objectives of this study were to identify the top software development factors that affect the overall success or failure of a software project using exploratory data analysis to find relationships between these factors, and to develop and compare risk prediction models that use machine learning classification techniques such as logistic regression, decision tree, random forest, support vector machine, K-nearest neighbors, and Naive Bayes. The findings of this study are as follows: in GSD, the top 18 factors influencing the software project are listed; and experiments show that the logistic regression and random forest models provide the best results, with an accuracy of 89% and 85%, respectively, and an area under the curve of 73% and 71%, respectively.**

***Keywords*—Global software development; distributed development; risk prediction model; machine learning**

I. Introduction

The entire software development approach has permanently changed in the last two decades to support distributed development environments with distributed teams [1]. This strategy can be described as a contract between two parties, with the client representing advanced countries and the vendor representing developing countries, with the goal of achieving mutual interests [2]. Therefore, the main reason for the widespread use of global software development (GSD) is that clients worldwide need highly specialized resources and tools at a reasonable price [3].

In addition, GSD has seen a considerable increase in contracts and business in recent years. The use of distributed development teams in various time zones and geographic locations may be referred to as the ‘new age’ of development projects employing GSD [4]. The affordable price of GSD is a significant factor contributing to its appeal. Consequently, there has been great success in the mutual benefit between clients and vendors [5], [6]. Some benefits of adopting GSD include sharing knowledge, using the most recent technologies, access to resources, economic benefits, lower expenses, and successful overall project completion [7], [8].

In addition, the challenges and limitations that have a significant effect on GSD should be pointed out. For example, it can be difficult for distributed teams to communicate with each other and work together due to language barriers, cultural norms and limits, time zones, leadership, team capabilities, and project management [9]–[11]. One of the most serious issues confronting GSD is the location, distance, and communication between the distributed teams [12]. In addition, the problem of team communication has been solved owing to the benefits of using agile methods such as scrum [8].

However, risks remain when clients attempt to adopt and use this approach in their projects. It can also yield the opposite results if it is misused. In the beginning, the term "risk" can be identified as a collection of software project characteristics, situations, and regulations that present a hazard to a project's overall success. It is also important to determine how often these risks occur, and how to prepare for them [13].

The Project Management Institute (PMI) shows that most risk management methods and procedures are ignored and thrown out, especially in the IT industry, because they are too general or only work in a specific situation [14]. Despite this, software projects that use techniques and tools to predict risk can detect approximately 70% and avoid 90% of harmful risks [15].

So, companies need to know the benefits and the risks of adopting the GSD approach, in an early stage of the development, to avoid any financial loss. In addition, companies need to also know if adopting GSD approach is suitable for their project or it will have negative results. Therefore, a software risk prediction model using the machine learning classification techniques was provided in this study, to make a prediction of the success or failure of the software project in the domain of GSD.

In this study, the following are discussed: First, previous systematic literature reviews were reviewed to identify the top software risk factors affecting GSD. Second, a dataset was collected from software projects in various regions of the world. Third, exploratory data analysis (EDA) was conducted to find different insights and correlations between these factors and each other. Fourth, software risk prediction models were built using different supervised machine learning classification techniques. Finally, software risk prediction models were evaluated and represented to determine the best model suitable for the GSD approach.

As a result, this study answers two main questions in the section between parentheses. RQ1: Which software risk factors

are essential to the GSD domain and significantly affect the software risk prediction? (Section III-A)

RQ2: What are the best machine learning techniques for software risk prediction in GSD? (Section V)

The remainder of this paper is organized as follows. Section II presents related work. The methodology is described in detail in Section III. Section IV presents an examination and measurement of the precision. Section V presents the results of the proposed model. Section VI discusses the validity threats. In Section VII the conclusion is presented, finally, in Section VIII, additional work is listed to be considered in the future.

II. Related Works

This section presents two types of studies. The first one concerns a systematic literature review related to GSD factors, and the second one is related to software risk prediction using machine learning and other techniques.

A. Systematic Literature Review for GSD Factors:

In [5], an empirical investigation was conducted to figure out the top requirements of engineering (RE) practices in GSD. Among the 66 practices, the results showed that only six key factors play an important role in GSD, as listed below:

1) *Identify and consult with system stakeholders.*
2) *Prioritize requirements.*
3) *Define system boundaries.*
4) *Define standard templates for requirements.*
5) *Check if requirements document meet your standards.*
6) *Uniquely identify each requirement.*

The dataset was collected by conducting an online survey questionnaire. For the evaluation of these factors, 56 experts from GSD were involved. Limitation and future work: the questionnaire relied only on closed questions and focused only on the company size, testing these factors, and trying to develop a framework to be used in the future.

In [16], the authors tried to prioritize the success factors that affect requirement change management (RCM) in the GSD. Fuzzy logic analytical hierarchy progress (FAHB) was used to conduct the prioritization. The result of this study was to find out the RCM success factors and categorize them into four groups: team, technology, process, and organizational management. The dataset was collected by conducting a questionnaire survey and retrieved around 81 responses. Evaluation metrics for the prioritization were conducted by using experts' responses. Limitations and future work: sample size of the dataset needs to be widened, and organization size and types should be considered, in addition, success factors, barriers, and best practices need more investigation and analysis.

The authors in [17], focused on scaling agile projects in the domain of GSD. They mapped 44 agile practices to the SAFe Framework. Instructions were given for how the SAFe practices can be used in agile global software development (AGSD) projects. The dataset was collected by reviewing 86 studies. Of these studies, only 24 papers discussed the scaling of agile, from which the authors selected 44 practices to be mapped on the SAFe Framework. Limitations and future work: (AGSD) practices need to be evaluated and should also be tested in the real industry. In addition, the mapped process of these practices needs to be evaluated.

B. Software Risk Prediction Models:

In [15], a software risk prediction model was created based on risk analysis of the project by using its context history and project characteristics in the software development life cycle (SDLC) as shown in Fig. 1. The model is called the Atropos model and consisted of six main phases listed below:

1) *Data Gathering through interface and bulk uploading.*
2) *Similarity by characteristics of the project.*
3) *Store context histories of the project.*
4) *Similarities by context histories.*
5) *Recommendation of any potential risks.*
6) *Risk management and monitoring.*

The dataset was collected based on 153 software projects from a financial company. Evaluation metrics of the model showed an acceptance rate of 73% and an accuracy rate of 83%, and these results were assessed by 18 experts. Limitations and Future work for the model are to improve the model's accuracy, to improve the proposed model and methodology, additional use of prototype, the number of practitioners, and the duration of the study (5 weeks only).

In [9], artificial neural network (ANN) model was created to predict the risk factors in GSD. The model used algorithms such as Levenberg–Marquardt, Bayesian Regularization, and Scaled Conjugate Gradient. The dataset was collected by sending 760 questionnaires to companies. 390 were received, and 116 were rejected, leaving 274 responses that were used as the primary data set. Evaluation metrics of the model were conducted by using least mean square error (MSE), and the results showed that Bayesian Regularization gave better results as compared with the other two approaches and matched the results from these studies [18], [19]. Limitations and Future work for the model are the sample dataset needs to include many companies and random data collection should be used to generalize the model, also the author recommended to use deep learning to get more insights and accurate results in the future.

Fig. 1. Shows the Atropos *S*ix-Stage *M*odel [15].

In [20], the authors provided a software reliability prediction algorithm. They used fuzzy logic and ANN in their model. The dataset was collected from John Musa of Bell Laboratories and received from the IEEE repository. Evaluation metrics of the model were conducted by using root mean square deviation error (RMSE) and showed that the fuzzy-neural method was the best compared to other algorithms. Limitations and future work for the model: the model is restricted to one factor (time to failure). In addition, many software risk factors should be used to evaluate this model better.

The authors in [21]developed a fuzzy logic hybridized framework for software risk prediction models during the decision-making process. Technique for order of preference by similarity to ideal solution (IF-TOPSIS), fuzzy decision-making trial and evaluation laboratory (DEMATEL), and crow search algorithm (CSA), optimized adaptive neuro-fuzzy inference system (ANFIS) were used for the software model prediction. The dataset consisted of 93 software projects, 70% used for training and the remaining used for testing and validating the model. The results showed that integrated fuzzy was accurate in software risk prediction.Limitations and future work: make a set of decisions and use many software factors and advanced machine learning techniques to improve and validate the results.

To reduce cost risks, the authors of [22] amplified the constructive cost model (COCOMO-II) in the GSD context. The dataset was collected by conducting a questionnaire and receiving around 175 responses. Evaluation metrics of the model were conducted by using Magnitude of Relative Estimates (MRE) and experts' judgment. Limitations and Future Work: the model is in an early stage and needs more validation and evaluation. In addition, mathematical or machine learning (ML) techniques may be used in the future.

In [23], the authors developed ML models for defect prediction in the domain of software reliability and performance. The models were built using ANN, random forest (RF), random tree (RT), decision table (DT), linear regression (LR), Gaussian processes (GP), SMOreg, and M5P. The dataset for these models was from the NASA promise repository. The results showed that the combination of different ML algorithms is effective in the prediction of software defects. Evaluation matrices used were correlation coefficient (R^2), mean absolute error (MAE), (RMSE), relative absolute error (RAE), and root relative squared error (RRSE). Limitation and Future works: different datasets and ML algorithms can be used to evaluate the results. In addition, more investigation into software factors should be conducted to improve these results.

Most previous studies concentrated on a limited number of factors, as summarized in Table I. In addition, the dataset needs to be enlarged to include more regions, and (ML) techniques need to be improved and evaluated using real data from software companies, as will be provided in the subsequent section.

TABLE I. OVERVIEW OF PREVIOUS RESEARCH STUDIES FOR SOFTWARE PREDICTION MODELS

Reference	Dataset	ML Techniques and algorithms	Evaluation metrics	Limitation and Future work
(Filippetto et al, 2021) [15]	The dataset was collected based on 153 software projects from a financial company.	Risk analysis of the project by using its context history and project characteristics in the (SDLC).	Acceptance rate of 73% and an Accuracy rate of 83%, and these results were assessed by experts	1. Improve the proposed model methodology and accuracy. 2. Additional use of prototype. 3. Number of practitioners and the duration of the case study should be increased.
(Iftikhar et al, 2021) [9]	The dataset was collected by sending 760 questionnaires to companies. 390 were received, and 116 were rejected, leaving 274 valid responses.	(ANN) model was created to predict the risk factors in GSD such as: Levenberg–Marquardt, Bayesian Regularization, and Scaled Conjugate Gradient.	MSE	1. the sample dataset needs to include many companies and random data collection should be used to generalize the model. 2. Deep learning should be used to get more accurate results.
(Sahu et al, 2018) [20]	The dataset was collected from John Musa of Bell Laboratories and received from the IEEE repository.	Fuzzy logic and ANN were used for building a software reliability prediction model.	RMSE	1. Model was restricted to one factor (time to failure). 2. Many software risk factors should be used to evaluate this model better.
(Suresh et al,2021) [21]	The dataset consisted of 93 software projects, 70% used for training and the remaining used for testing and validating the model.	Fuzzy logic hybridized framework for software risk prediction models during the decision-making process.	CSA	1. Make a group of decisions making and use sophisticated ML techniques 2. Use many software factors
(Khan et al,2021) [22]	The dataset was collected by conducting a questionnaire and receiving around 175 responses	Amplified COCOMO-II Model in the context of GSD.	(MRE, experts' judgment	1. The model is in an early stage and needs more validation. 2. Mathematical or ML techniques may be used in the future.
(Assim et al,2020) [23]	The dataset for these models was from the NASA promise repository	ANN, RF, RT, DT, LR, GP, SMOreg, and M5P were used.	(R^2), (MAE), (RMSE), (RAE) and (RRSE)	1. different datasets and ML algorithms can be used to evaluate the results. 2. Investigation into more software factors to improve these results.

III. Research Methodology

This section describes the methodology used to develop the GSD-applicable software risk-prediction model. Fig. 2 illustrates the six phases of the proposed model. Systematic Literature review (SLR) analysis (Section III-A), dataset selection (Section III-B), dataset preprocessing (Section III-C), modeling (Section III-D), experimental evaluation (Section IV), and risk prediction results (Section V).

A. Systematic Literature Review Analysis:

The proposal to build the software risk prediction model was based on many systematic literature reviews (SLR) that collected the software risk factors that affect the software in GSD. SLRs included empirical studies published between 2018 and 2022. After reviewing these studies, a list of 145 factors essential to software project success was created. (Available in Appendix "I").

Then, these factors were analyzed and reprocessed to determine the most significant factors in the GSD domain. To do this, the following three steps were followed:

1) Merging step: There were several duplicates; therefore, the first step in removing these duplicated factors was to merge the duplicates, which helped lower the total number by more than half.

2) Filtration step: After the merging stage, the factors were ranked and filtered by selecting only those with a frequency rate of greater than 50 percent. In this manner, the top 18 factors that affect software in the GSD domain can be collected.

3) *Categorization step:* In this phase, the top 18 factors were categorized into four categories: requirements, management, technical, and cultural, as shown in Table II to answer RQ1.

B. Dataset Selection

This subsection describes the data collection procedure and descriptive analysis of the dataset used to construct the model. The dataset was collected through a questionnaire survey and interviews with software companies and experts from various global regions. The main target was to focus on organizations that had extensive experience with outsourcing and were already using the GSD method. The dataset consists of information from 140 software projects in the GSD domain. Then, the data was gathered by conducting a questionnaire survey and interviews with companies and experts. The questionnaire is based on the top 18 factors listed in Table II. The data were collected from various regions that support a wide range of clients and vendors, including western Europe, central and eastern Europe, Africa, and the Middle East.

The dataset attributes were project ID, region, job, experience, company type, requirement clarity, project scope, requirement changes, project planning, project size, project management, communication, cost, commitment, modern technologies, roles and responsibilities, skilled staff, organizational stability, language and culture, time difference, progress, team size, methodology, and project status.

Then, the attributes were classified into numerical and categorical categories. The independent and dependent attributes were then determined. The dependent attribute is "Project Status." The remaining 23 attributes were independent attributes. Table III presents a more detailed description of the attributes of the dataset. In addition, Appendix "II" provides a sample of the questionnaire with attributes represented as questions.

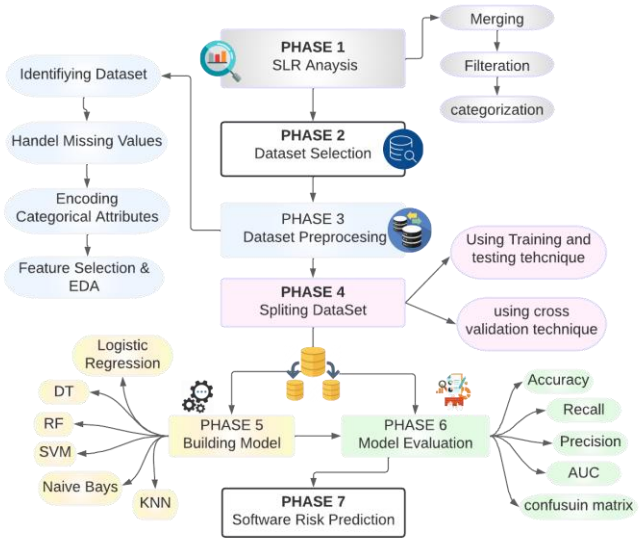

Fig. 2. A Proposed Model for Software Risk Prediction.

TABLE II. Top 18 Software Factors that Affect GSD

Requirement Factors
1. Requirement ambiguity 2. Requirement changes 3. Requirements scope 4. New technologies 5. Project size
Management Factors
6. Competence level of project manager 7. No planning or inadequate planning 8. Low commitment of stockholders 9. Progress measure 10. Cost balance 11. Lack of roles and responsibilities 12. Team size
Technical factors
13. Staff does not have required skills 14. Unstable organizational environment 15. Methodology followed 16. Communication infrastructure and process
Cultural factors
17. Language and culture differences 18. Time zone difference

TABLE III. DATASET VARIABLES AND DESCRIPTIVE STATISTICS

n	Attributes	Type	Description	Mean	Std	Min	50%	Max
1	Project ID	Numerical (int64)	Project unique ID, which starts with one and ends by 140	___	___	1	___	140
2	Region	Categorical (object)	Region attributes lie in 5 main regions: Western Europe, Central, and Eastern Europe, Asia, Africa, and the Middle East.	___	___	___	___	___
3	Job	Categorical (object)	The job role of the person/company developer who filled the form	___	___	___	___	___
4	Experience	Numerical (int64)	Team/individual experience measured in years	3.925	4.171	1	3	30
5	Company Type	Numerical (int64)	company types measured by (national, international, and startup)	0.535	0.528	0	1	2
6	Requirement Clarity	Numerical (int64)	the level of requirements clearness is measured as (clear, moderate, unclear, and ambiguous)	2.1	0.798	1	2	4
7	Project Scope	Numerical (int64)	the project scope is measured as (clear, moderate, unclear, and ambiguous)	2.071	0.810	1	2	4
8	Requirement Changes	Numerical (int64)	the project scope is measured as (minor, normal, heavy, and messy)	2.55	0.798	1	2	4
9	Project Planning	Numerical (int64)	the project planning is measured as (clear, moderate, unclear, and ambiguous)	2.214	0.863	1	2	4
10	Project Size	Numerical (int64)	the project size is measured as (Enterprise, large, medium, and small)	2.185	0.918	1	2	4
11	Project Management	Numerical (int64)	Project manager's quality is measured as (Expert, Moderate, Basic, and None)	2.142	0.844	1	2	4
12	Communication	Numerical (int64)	communication is measured as (Excellent, Moderate, need enhancements, and worst)	2.028	0.804	1	2	4
13	Cost	Numerical (int64)	cost is measured as (balanced, moderate, and not balanced)	1.842	0.626	1	2	3
14	Commitment	Numerical (int64)	stakeholders' commitment is measured as (High, moderate, and low)	1.75	0.669	1	2	3
15	Modern Technologies	Numerical (int64)	modern technologies are measured as (many, normal, and few)	1.75	0.613	1	2	3
16	Roles and Responsibilities	Numerical (int64)	responsibilities are measured as (clear, moderate, and unclear)	1.721	0.74	1	2	3
17	Skilled Staff	Numerical (int64)	skilled staff are measured as (Agree, moderate, and disagree)	1.485	0.64	1	1	3
18	Organization Stability	Numerical (int64)	organizational stability is measured as (stable, normal, and unstable)	1.607	0.716	1	1	3
19	Language and Culture	Numerical (int64)	language and culture are measured as (reasonable, can be handled, and unreasonable)	1.55	0.627	1	1	3
20	Time Difference	Numerical (int64)	Time Difference is measured as (reasonable, can be handled, and unreasonable)	1.7	0.675	1	2	3
21	Progress	Categorical (object)	Progress level Measured by (task level, module level, sprint level, and delivery level)	___	___	___	___	___
22	Team Size	Numerical (int64)	Team size measured by team members	0.807	0.855	0	1	4
23	Methodology	Categorical (object)	the methodology was measured as (waterfall, scrum, Kanban, extreme programming, feature-driven, lean development, crystal, and dynamic system development, and rapid development)	___	___	___	___	___
24	Project status	Numerical (int64)	Project status represents this project is success or failed	0.814	0.39	0	1	1

C. Dataset Preprocessing

In this subsection, the data preprocessing techniques are presented. This phase can be considered as the initial phase for building the machine learning model. Real-world data are often incomplete, inconsistent, or incorrect (because they have outliers or mistakes). Thus, preprocessing techniques must be conducted to help refactor the dataset to keep it clean, formatted, and organized [24]. This subsection includes four steps of dataset preprocessing: identifying the dataset, finding, and handling missing values, encoding categorical attributes, and feature selection.

1) Identify dataset: During data preparation, it is essential to identify insights into the dataset because improper handling may lead to misleading software model results and serious model risks. Table III shows that the dataset is divided into two main types: categorical and numerical. It also provides a full picture of the dataset's characteristics, such as its type,

description, mean, standard deviation (Std), and minimum and maximum values.

2) Finding and handling missing attributes: Incomplete data can lead to inaccurate results. Consequently, these situations may be addressed by finding the mean of the attributes using numerical data. This is more efficient than the usual methods of treating missing values, which include omitting the entire row or column, as this might lead to data misrepresentation or bias in the dataset. Alternatively, mean, median, or mode can be used.

3) Encoding categorical data: As it is known, machine learning deals with numerical attributes only. Thus, categorical attributes can't be used until they are transformed into numerical data. As a result, only four categorical attributes which are: "Region," "Job," "Progress," and "Methodology" should be transformed into numerical attributes. The Python scikit-learn library label encoder technique was used to transform the categorical attributes into numerical attributes. In this technique, each label is assigned a unique integer based on the alphabetical ordering [25].

4) Feature selection: From the list of 24 attributes in Table III, Independent attributes that are significant to the model must be chosen. Therefore, weak attributes or attributes that do not have a relationship with the model should be excluded. To determine these relationships, a correlation analysis was conducted, which is a common multivariate (EDA) that relies on statistical techniques to measure the linear relationship between attributes and each other to obtain a better insight into the factors and their relationships [26]. The correlation coefficient is the unit of measurement used to calculate the intensity between the two variables. It has three types:

a) Positive correlation: (0 to 1) means that both attributes are in the same direction; an increase in one will increase the other, and vice versa

b) Negative correlation: (-1 to 0) means that both attributes move in the opposite direction; an increase in one will decrease the other, and vice versa.

c) Weak/zero correlation: (0) means that the two attributes do not affect each other.

The formula for the correlation coefficient can be written as:

$$\boldsymbol{R} = \frac{\boldsymbol{\Sigma}(\boldsymbol{x_i}-\bar{\boldsymbol{x}})\,(\boldsymbol{y_i}-\bar{\boldsymbol{y}})}{\sqrt{\boldsymbol{\Sigma}(\boldsymbol{x_i}-\bar{\boldsymbol{x}})^2\,\boldsymbol{\Sigma}(\boldsymbol{y_i}-\bar{\boldsymbol{y}})^2}} \quad (1)$$

where R is the correlation coefficient, usually from -1 to 1, $\boldsymbol{x_i}$ is the value from the X dataset, $\boldsymbol{x}$ is the mean value of the X dataset, $\boldsymbol{y_i}$ is the value from the Y dataset, and $\boldsymbol{y}$ is the mean value of the Y dataset. More details regarding the dataset attribute correlation are presented in Fig. 3.

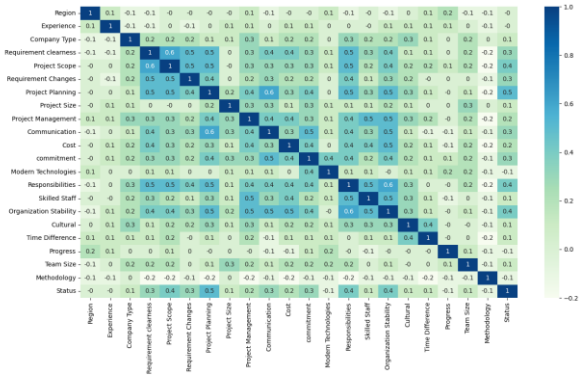

Fig. 3. Correlation Analysis for Dataset Attributes.

Fig. 4 represents the independent attributes that have R >= 0.4 (“Project Scope,” “Project Planning,” “Responsibilities,” “and Skilled Staff”) and independent attributes that have R <= 0 (“Region” and “Experience”), which will be excluded from the dataset so as not to affect the proposed model.

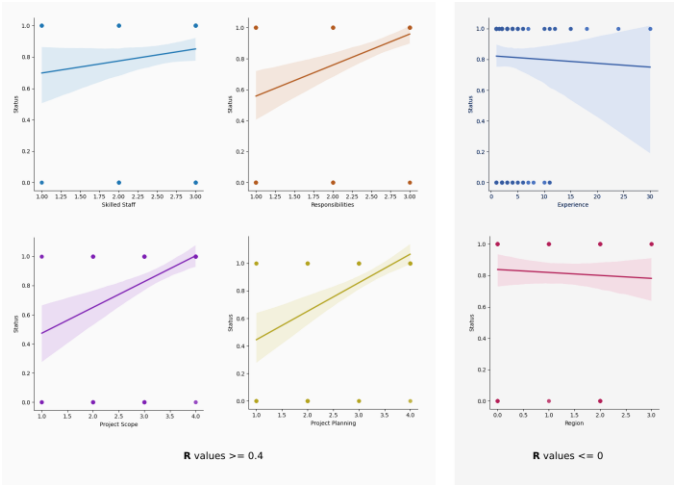

Fig. 4. Correlation Coefficient (R) between Attributes.

D. Machine Learning Model Builder

In this subsection, the implementation of ML models is discussed. The category for the proposed ML model is called a "supervised problem" because the dataset is labeled (one with the correct answer) which can be used to teach the model how to predict software risk [27]. The model is based on the top six machine-learning classification algorithms: logistic regression, (DT), (RF), support vector machine (SVM), K-nearest neighbor (KNN), and naïve Bayes.

1) *Logistic regression:* The logistic regression algorithm is a classification technique based on statistical procedures. Logistic regression is a widely used ML algorithm for binary classification that makes predictions based on the sigmoid function [28].

The sigmoid function can be defined as a mathematical procedural function that takes any real number and maps it to a probability between one and zero. The formula for the sigmoid function can be expressed as:

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad (2)$$

where $\boldsymbol{\sigma(x)}$ is the sigmoid function that returns values ranging from zero to one, $\boldsymbol{x}$ represents the sample, $\boldsymbol{e^{-x}}$ represents the inverse of the exponential function $\frac{\mathbf{1}}{\boldsymbol{e^x}}$

2) *Decision tree (DT):* The decision tree algorithm can be represented as a hierarchical or flowchart which represents the data with decisions [25]. The decision tree has many branches created by splitting the dataset into subsets based on the essential attributes, and each branch can be considered as an if-else statement. To create the hierarchical structure in the decision tree, the Gini index algorithm was used to select the best attribute selection measures (ASM) to split the data. The Gini index algorithm can be written as:

$$\boldsymbol{Gini(D) = 1 - \sum_{i=1}^{c} Pi^2} \quad (3)$$

where $\boldsymbol{c}$ is the total number of classes, and $\boldsymbol{Pi}$ is the probability of picking the data point with class $\boldsymbol{i}$

3) *Random forest (RF):* An RF is a collection of decision trees. It is a common ensemble method that aggregates the results of multiple models. RF uses the bagging technique, which allows each tree to be trained on random dataset sampling and takes the majority vote from the trees [25].

4) *Support Vector machine (SVM):* SVM is a machine learning algorithm that can be used for classification and regression analysis [26]. The purpose of SVM is to classify data based on hyperplanes in an N-dimensional (number of attributes) space, which is the border between positive and negative classes, maximizing the distance between data points from different classes.

5) *K-Neatest Neighbour (KNN):* The KNN is an analogy-based ML algorithm. In general, it uses the Euclidean distance to calculate the distance between points and each other and then assigns the label of new data based on the labels of the nearest data points. The Euclidean distance can be written as

$$\boldsymbol{d\,(y, x) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}} \quad (4)$$

where d is the Euclidean distance; $(\boldsymbol{y, x})$ is the two-point Euclidean N-space; $\boldsymbol{x_i, y_i}$ represent the Euclidean vectors; n = N-space (attribute numbers).

6) *Naïve bays:* The naive Bayes algorithm depends on Bayes' theorem, which describes the probability of an event based on prior knowledge. Naive Bayes assumes that each feature is independent of the other [27]. The calculation of naïve Bayes can be represented as:

$$\boldsymbol{P(A|B)} = \frac{\text{P(B | A) * P(A)}}{\text{P(B)}} \quad (5)$$

where $P(A|B)$ is a conditional probability, that is, the probability of an event A occurring given that B is true. $P(B|A)$ is also a conditional probability: the probability of event B occurring given that A is true, P(A) and P(B) are the probabilities of observing A and B, respectively, without any conditions.

IV. Experimental Evaluation

For model evaluation, different techniques and algorithms are described in this section. The essential part of any model is to determine whether it is accurate. Five evaluation metrics were used to measure the confusion matrix, accuracy, recall, precision, and area under the curve (AUC).

1) *Confusion matrix:* A confusion matrix is the best way to solve binary classification problems [35] because it shows the actual and predicted values and summarizes them in a matrix, as shown in Table IV.

2) *Accuracy:* Accuracy is the most important indicator for measuring a model's performance [29]. The purpose of the accuracy was to measure the percentage of the total number of correctly classified examples predicted over the total number of examples. The metric equation can be written as.

$$\boldsymbol{ACC = \frac{TP+TN}{TP+TN+FP+FN}} \quad (6)$$

3) *Recall:* The recall evaluation metric, also known as the true positive rate (TPR), is used to determine the proportion of correctly classified positive classes [29]. The metric equation can be written as

$$\boldsymbol{R(TPR) = \frac{TP}{TP+TN}} \quad (7)$$

4) *Precision:* The primary purpose of precision metrics is to measure the positive patterns from the total predicted patterns in a positive class [29]. The metric equation can be written as

$$\boldsymbol{P = \frac{TP}{TP+FP}} \quad (8)$$

5) *AUC:* The area under the ROC curve (AUC) is a popular metric for comparing and optimizing machine-learning models [25]. A higher AUC indicates a better model performance. For classification evaluation, the AUC is more accurate than the accuracy metric, although the computational cost is high compared to the accuracy metric [25]. The AUC metric equation can be expressed as follows:

$$\boldsymbol{AUC = \frac{S_P - T_p(T_n + 1)\,/\,2}{T_p T_n}} \quad (9)$$

where $\mathbf{S_P}$ is the summation of all the positive examples, $\mathbf{T_P}$ is the number of positive examples, and $\mathbf{T_n}$ is the number of negative examples.

TABLE IV. Confusion Matrix Value Summarization

	Predicted (0)	**Predicted (1)**
Actual (0)	**True Negative (TN)**	**False Positive (FP)**
Actual (1)	**False Negative (FN)**	**True Positive (TP)**

V. Result and Discussion

In this section, the results of the software risk prediction model were discussed using six classification machine learning algorithms: logistic regression, DT, RF, SVM, (KNN), and naïve Bayes, and by using the dataset of 140 software projects in the real industry of global software development. Algorithm I present the pseudocode for the risk-prediction model using training data of 80% and 20% of the testing data.

Algorithm I: Pseudo-Code for Risk Prediction Model

Input: Import the dataset from a CSV File.

1: **Data Preprocessing Phase:** [data cleaning, missing values]

2: **Feature Transformation and Categorical Feature Encoding:**

3: **Apply EDA and Feature Selection**

4: **Dataset split: 80% for training and 20% for testing.**

5: **Set**: Model = Logistic Regression, SVM, KNN, DT, RF, and Naïve Bayes

7: **for** each Model **do**

8: **Select:** the ML model to use

9: **Use:** the training dataset to feed the proposed model

10: **Apply:** Testing the model using a training dataset

11: **Calculate:** confusion matrix

12: **Calculate:** The Accuracy metrics

13: **Calculate:** The Recall metrics

14: **Calculate:** The Precision metrics

15: **Calculate:** the AUC metrics

16:**end for**

The software risk prediction model was constructed using the top six classification techniques: logistic regression, SVM, KNN, DT, RF, and naïve Bayes. Five evaluation metrics were used to find the most optimal ML algorithms to fit the risk prediction model. The model was conducted using the programming language Python and other third-party packages such as NumPy, Pandas, Scikit-Learn, Pandas, Matplotlib, and Seaborn, running on the MacBook Pro with the following specifications: Intel Core i5, 2.0Ghz, 16GB, and 512GB SSD. Table V presents a comparison of the six ML classification techniques using different evaluation metrics. Finally, the following research question was answered:

RQ2: What are the best machine learning techniques for software risk prediction in GSD?

Table V indicates that the top three techniques with the highest accuracy, AUC, recall, and precision were logistic regression, random forest, and SVM, with accuracy percentages of 89%, 85%, and 82%; AUC percentages of 73%, 71%, and 48%; recall percentages of 96%, 92%, and 96%; and precision percentages of 92%, 92%, and 85%, respectively.

TABLE V. Summarization of Confusion Matrix Values

Model Name	Accuracy	AUC	RECALL	Precision
Logistic Regression	0.89	0.73	0.96	0.92
SVM	0.82	0.48	0.96	0.85
KNN	0.71	0.52	0.79	0.86
DT	0.71	0.62	0.75	0.90
Random Forest	0.85	0.71	0.92	0.92
Naïve Bayes	0.71	0.62	0.75	0.90

Therefore, logistic regression can be considered the optimum ML algorithm for software risk prediction in the domain of GSD, with an accuracy rate of approximately 90%. Further details regarding the algorithm's confusion matrix showing the four values of true positives, true negatives, false positives, and false negatives are shown in Fig. 5.

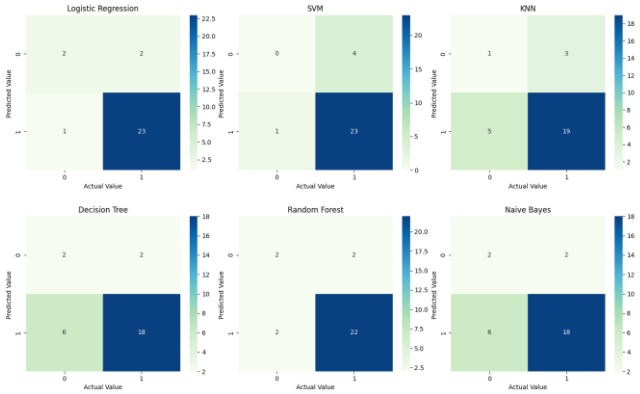

Fig. 5. The ML Confusion Matrix for the Six Algorithms.

In addition, to improve the results of other algorithms, another technique was applied for splitting the dataset called cross-validation, which is a statistical method for splitting data to test and train a model on different iterations. In other words, cross-validation split the training dataset into k smaller sets. This technique helped us improve the accuracy of most of the six algorithms, obtain better insights, and solve overfitting classification problems.

Table VI Shows a comparison of the six ML classification techniques after applying the cross-validation technique using five k-fold; four of them were used for the model training, and the remaining fold was used for validating the model.

TABLE VI. ML Algorithms are Subjected to Cross-Validation

Model Name	Accuracy	AUC	RECALL	Precision
Logistic Regression	0.80	0.72	0.92	0.85
SVM	0.81	0.75	1	0.81
KNN	0.80	0.74	0.93	0.85
DT	0.78	0.65	0.85	0.90
Random Forest	0.80	0.81	0.92	0.84
Naïve Bayes	0.80	0.84	0.84	0.91

After applying the cross-validation technique, the accuracy of the KNN, DT, and naïve Bayes were increased by 9% to reach 80%. Thus, from Table VI can be observed that the accuracy of the six algorithms is approximately 80%, and the top three algorithms are Support Vector Machine, KNN, and logistic regression, with accuracy percentages of 81, 80, and 80%, respectively.

VI. Threats to Validity

This section discusses the reality of the study, based on internal and external threats and construct validity [30].

Internal validity relates to whether the investigated software risk prediction model is affected by other factors, such as Python Scikit-learn library parameters. Unfortunately, there is no standard method to choose this parameter, but the standard parameters and best practices was used in the Scikit-learn library to solve this problem [25]. The standard parameters, best practices, and configuration related to ML implementation for the six classification algorithms are provided in Appendix "III". Another internal threat is the split of the dataset; the dataset was divided into training and test sets at proportions of 80% and 20%, respectively. Random assignments were avoided to avoid influencing the model results. In addition, another preferred technique was used, called cross-validation, which splits the dataset into smaller datasets to train and test the model and calculate the average of these results to determine the most accurate result for the risk prediction model.

External validity is related to the generalization of the software risk-prediction model [30]. Dataset samples were tried to obtain from most of the companies that outsource and apply the GSD concept in different regions; however, with a limited number of datasets, some difficulties in generalizing the findings appear. In addition, there were difficulties in obtaining the data set because outsourcing companies often had to pay for their data, which limited the size of the dataset that could use.

The final threat is related to the reliability of the proposed model, this point was considered when conducting the model to validate and analyze its performance using the confusion matrix, accuracy indicator, recall metric, precision metrics, and AUC metrics. Also, these results were compared with those of the top six classification algorithms to determine the best one for the proposed model.

VII. Conclusion

Obtaining a solid and accurate software risk-prediction model has always been difficult in global software development. The applied model will help software companies, experts, project managers, and developers predict software risk, which will reduce the amount of time and money spent on this approach.

A dataset of 140 software projects in different regions was used to build the model, and was collected using 18 software factors, which were carefully collected from past studies and reviewed by experts. The data preprocessing phase consisted of four steps: identifying the dataset, handling missing values, encoding categorical attributes, feature selection, and conducting EDA analysis. Two techniques were used for the dataset splitting. The first technique is the common traditional technique, which uses 80% for training and 20% for testing without using any random or shuffle to avoid influencing the results of the model. The other technique is cross-validation using 5-k folds, with 4-folds used for model training and the remaining used to validate the model.

The results show that the top two algorithms were logistic regression and random forest with accuracy percentages of 89% and 85%, respectively. Also, cross-validation was used technique to improve the accuracy of the other models by approximately 80% and obtained better results.

VIII. Future Work

Below are suggestions to improve the proposed model and the dataset that can be considered in the future:

1) The dataset needs to be expanded, by gathering the dataset from distributed locations that adopt the GSD approach.

2) Generalization of the findings.

3) Enhancing the accuracy of the ML algorithms.

References

[1] J. Menezes, C. Gusmão, and H. Moura, "Risk factors in software development projects: a systematic literature review," Software Quality Journal, vol. 27, no. 3. Springer New York LLC, pp. 1149–1174, Sep. 01, 2019. doi: 10.1007/s11219-018-9427-5.

[2] S. Ali, H. Li, S. U. Khan, M. F. Abrar, and Y. Zhao, "Practitioner's view of barriers to software outsourcing partnership formation: An empirical exploration," Journal of Software: Evolution and Process, vol. 32, no. 5, May 2020, doi: 10.1002/smr.2233.

[3] A. Iftikhar, S. Musa, M. Alam, M. M. Su'ud, and S. M. Ali, "A Survey of Soft Computing Applications in Global Software Development," in 2018 IEEE International Conference on Innovative Research and Development (ICIRD), 2018, pp. 1–4.

[4] University of Management and Technology (Pakistan), Institute of Electrical and Electronics Engineers. Lahore Section., and Institute of Electrical and Electronics Engineers, 3rd International Conference on Innovative Computing (ICIC) : (IC)2 2019 : 1st-2nd November 2019, Lahore, Pakistan.

[5] J. A. Khan, S. U. R. Khan, J. Iqbal, and I. U. Rehman, "Empirical Investigation about the Factors Affecting the Cost Estimation in Global Software Development Context," IEEE Access, vol. 9, pp. 22274–22294, 2021, doi: 10.1109/ACCESS.2021.3055858.

[6] M. A. Akbar, J. Sang, Nasrullah, A. A. Khan, M. Shafiq, and Fazal-E-Amin, "Towards the Guidelines for Requirements Change Management in Global Software Development: Client-Vendor Perspective," IEEE Access, vol. 7, pp. 76985–77007, 2019, doi: 10.1109/ACCESS.2019.2918552.

[7] M. A. Akbar, M. Shafiq, T. Kamal, and M. Hamza, "Towards the successful requirements change management in the domain of offshore software development outsourcing: Preliminary results," International Journal of Computing and Digital Systems, vol. 8, no. 3, pp. 205–215, May 2019, doi: 10.12785/ijcds/080301.

[8] M. Rizwan, J. Qureshi, A. Al-Zaidi, and R. Qureshi, "Global Software Development Geographical Distance Communication Challenges Article in International Arab Journal of Information Technology," 2017. [Online]. Available: https://www.researchgate.net/publication/308952993

[9] A. Iftikhar, M. Alam, R. Ahmed, S. Musa, and M. M. Su'ud, "Risk Prediction by Using Artificial Neural Network in Global Software Development," Comput Intell Neurosci, vol. 2021, pp. 1–25, Dec. 2021, doi: 10.1155/2021/2922728.

[10] M. Yaseen and Z. Ali, "Success Factors during Requirements Implementation in Global Software Development: A Systematic Literature Review," International Journal of Computer Science and Software Engineering (IJCSSE), vol. 8, no. 3, 2019, [Online]. Available: www.IJCSSE.org

[11] B. J. Galli, "Addressing Risks in Global Software Development and Outsourcing," Int J Risk Conting Manag, vol. 7, no. 3, pp. 1–41, May 2018, doi: 10.4018/ijrcm.2018070101.

[12] Asim Iftikhar, Muhammad Alam, Shahrulniza Musa, and Mazliham Mohd Su'ud, "Trust Development in Virtual teams to Implement Global Software Development (GSD): A Structured Approach to Overcome Communication Barriers," in 2017 IEEE 3rd International Conference on Engineering Technologies and Social Sciences (ICETSS), 2017, pp. 1–5.

[13] B. G. Tavares, C. E. S. da Silva, and A. D. de Souza, "Risk management analysis in Scrum software projects," International Transactions in Operational Research, vol. 26, no. 5, pp. 1884–1905, Sep. 2019, doi: 10.1111/itor.12401.

[14] Project Management Institute, A guide to the Project Management Body of Knowledge (PMBOK guide), 6th ed. Newton Square, PA: Project Management Institute, 2017.

[15] A. S. Filippetto, R. Lima, and J. L. V. Barbosa, "A risk prediction model for software project management based on similarity analysis of context histories," Inf Softw Technol, vol. 131, Mar. 2021, doi: 10.1016/j.infsof.2020.106497.

[16] M. A. Akbar, M. Shameem, A. A. Khan, M. Nadeem, A. Alsanad, and A. Gumaei, "A fuzzy analytical hierarchy process to prioritize the success factors of requirement change management in global software development," Journal of Software: Evolution and Process, vol. 33, no. 2, Feb. 2021, doi: 10.1002/smr.2292.

[17] M. Marinho, R. Camara, and S. Sampaio, "Toward unveiling how safe framework supports agile in global software development," IEEE Access, vol. 9, pp. 109671–109692, 2021, doi: 10.1109/ACCESS.2021.3101963.

[18] S. K. Niranjan, V. N. M. Aradhya, Amity University, IEEE-USA, Institute of Electrical and Electronics Engineers. Uttar Pradesh Section, and Institute of Electrical and Electronics Engineers, Proceedings of the 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I) : 14-17 December 2016, Noida, India.

[19] A. N. Okon, S. E. Adewole, and E. M. Uguma, "Artificial neural network model for reservoir petrophysical properties: porosity, permeability and water saturation prediction," Model Earth Syst Environ, vol. 7, no. 4, pp. 2373–2390, Nov. 2021, doi: 10.1007/s40808-020-01012-4.

[20] K. Sahu and R. K. Srivastava, "Soft computing approach for prediction of software reliability," ICIC Express Letters, vol. 12, no. 12, pp. 1213–1222, Dec. 2018, doi: 10.24507/icicel.12.12.1213.

[21] 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, doi: 10.1007/s10115-021-01573-5.

[22] J. A. Khan, S. U. R. Khan, T. A. Khan, and I. U. R. Khan, "An Amplified COCOMO-II Based Cost Estimation Model in Global Software Development Context," IEEE Access, vol. 9, pp. 88602–88620, 2021, doi: 10.1109/ACCESS.2021.3089870.

[23] M. Assim, Q. Obeidat, and M. Hammad, "Software Defects Prediction using Machine Learning Algorithms," Oct. 2020. doi: 10.1109/ICDABI51230.2020.9325677.

[24] Jacqueline Kazil and Katharine Jarmul, Data Wrangling with Python: Tips and Tools to Make Your Life Easier. O'Reilly Media, Inc. 2016.

[25] E. Bisong, Building Machine Learning and Deep Learning Models on Google Cloud Platform. Apress, 2019. doi: 10.1007/978-1-4842-4470-8.

[26] H. D. P. de Carvalho, R. Fagundes, and W. Santos, "Extreme Learning Machine Applied to Software Development Effort Estimation," IEEE Access, vol. 9, pp. 92676–92687, 2021, doi: 10.1109/ACCESS.2021.3091313.

[27] R. Saravanan and Pothula Sujatha, "A State of Art Techniques on Machine Learning Algorithms: A Perspective of Supervised Learning Approaches in Data Classification," 2018.

[28] Hilbe and J.M, Practical Guide to Logistic Regression; Chapman and Hall/CRC: Boca Raton, FL, USA. 2016.

[29] H. M and S. M.N, "A Review on Evaluation Metrics for Data Classification Evaluations," International Journal of Data Mining & Knowledge Management Process, vol. 5, no. 2, pp. 01–11, Mar. 2015, doi: 10.5121/ijdkp.2015.5201.

[30] D. A. Broniatowski and C. Tucker, "Assessing causal claims about complex engineered systems with quantitative data: internal, external, and construct validity," Systems Engineering, vol. 20, no. 6, pp. 483–496, Nov. 2017, doi: 10.1002/sys.21414.

APPENDIX

1) factors that affect software in the GSD and explaining the three steps [collected, merged, and filtered] applied to these factors: https://github.com/Al3ameed/ML_Classification_GSD/blob/main/software%20factors.docx

2) An Example of the Questionnaire form (published on GitHub): https://github.com/Al3ameed/ML_Classification_GSD/blob/main/questionnaire_samples.zip

3) ML classification and a sample of the dataset that used for model prediction: https://github.com/Al3ameed/ML_Classification_GSD