Identification of Risk Factors in the Software Design Stage Using the C4.5 Algorithm

M. Akiyasul Azkiya[1], Deva Sindi Maulita[2], Jumanto[3]
Departement of Computer Science, Universitas Negeri Semarang[1,2,3]
akiyasulazk@students.unnes.ac.id[1], devasm145@students.unnes.ac.id[2], jumanto@mail.unnes.ac.id[3]

Article Info

Article history:

Received Jun 14, 2023
Revised Aug 8, 2023
Accepted Feb 15, 2024

Keyword:

Risk Factors
Software Design
Data Analysis
Algorithm C4.5
Data Mining

ABSTRACT

A strong design phase is necessary for good software. However, design errors in software can cause serious issues with its creation and use. Therefore, the goal of this study is to find risk variables that could have an early impact on software development. In this study, a machine learning technique called technique C4.5 is employed to create decision tree models. 100 respondents with software design experience participated in the online surveys and questionnaires that collected the data for this study in 2022. The C4.5 Algorithm was used in this study to analyze the data and determine the risk variables that affect the success of software design. The study's findings show that the C4.5 Algorithm-based model has a high level of accuracy (93.33%), which means that the data can offer crucial insights into understanding potential risks that may arise during the software design stage, enabling software developers to take the necessary precautions to lessen or eliminate these risks. In order to enhance the caliber and effectiveness of software design, this research is anticipated to provide a significant contribution to practitioners and academics in the field of software development.

Corresponding Author:

M. Akiyasul Azkiya
Department of Computer Science
Universitas Negeri Semarang
Building D, Sekaran, Gunung Pati, Semarang City, Central Java 50229, Indonesia
Email: akiyasulazk@students.unnes.ac.id

1. INTRODUCTION

One of the crucial phases in the development of software is software design. Methodologies for software development can be used on a variety of tasks, both straightforward and sophisticated. Reduce the likelihood of project failure is one of the objectives [1]. The success of the project and the level of user satisfaction are initially determined by this stage. It is necessary to distinguish between the quality of a software product and the quality of the development process when defining software quality [2]. Software is created to satisfy the unique requirements of the organization, to comprehend user requirements, and for individual use. Creating a workable design concept is the hardest task to accomplish in the early stages of design [3]. To effectively elicit user requirements in interactive device development, new methodologies, tools, and practices are needed [4]. Increasing the level of satisfaction with the defined demands is essential for a project's success through requirements engineering. The degree to which the system or product being built satisfies the demands and expectations of stakeholders is referred to as meeting these needs [5]. Guide architecture can assist development teams in large-scale software development projects in providing management with methodological advice and providing software with a greater level of abstraction [6]. To help software engineers overcome risks and lessen their negative effects, it is crucial to identify risk factors throughout the program design stage. The use of software measurement techniques may enhance software engineering process management, decrease necessary time and costs, and result in higher-quality software [7]. Furthermore, the appropriate prevention or mitigation can be carried out to lower the chance of failure and enhance product quality by identifying risk factors at the software design stage.

There have been various conversations about risk management in distributed software development in earlier research [8]. Prior research on the C4.5 technique covered attention-based lane shifts and a highway collision risk prediction model [9]. There has also been prior research on software risk, specifically a software vulnerability management methodology to reduce system risk and attack surface [10]. The approach and process of software engineering in blockchain-oriented software development are also covered in later studies [11]. The design of software systems, algorithms, and midstream implementation as downstream prerequisites will all be covered in later study by experts in the field of software development. They will also talk about product requirements and Innovative Analysis Ready Data (ARD) methods [12].

The complexity of the project and the hazards involved in software development are the issues that software developers must deal with. Project size, technology employed, functional and non-functional needs, and other elements can all influence how difficult a project is. Technical risks, project management risks, financial risks, and other risks are all possible when developing software. These risks, which include delayed project completion, poor product quality, and higher project expenses, may affect the success of the project. An essential component of a software project's success is accurately estimating the effort and expense involved [13]. To maximize the success of a software development project, it is crucial to undertake a risk analysis. This is since there are still numerous project management approaches that require more research [14]. Users' health may be in danger if there is software damage or failure on the gadget [15]. Finding potential risk factors is one technique to mitigate risk during the software design phase. Developers can create more effective risk management techniques by identifying risk factors. However, the method utilized to identify risk factors in projects involving software is still insufficient for dealing with the complexity and diversity of risk variables.

The C4.5 algorithm, a machine learning algorithm used to create decision trees that may be used for classification and prediction, is one method that can be used to identify risk variables. The fundamental idea is to gather data and transform it into a decision tree by using the rules required to produce the desired outcome [16]. Compared to other ways, this one can help developers more accurately detect risk variables. However, it is still uncommon to employ the C4.5 method to identify risk concerns during the software design phase.

In general, prior research did not verify and evaluate the methodologies used, focusing more on the explanation of the theory and fundamental ideas about the detection of risk factors at the software design stage. The C4.5 method is tested and evaluated more in this study to detect risk concerns at the software design stage. The usage of the C4.5 method, which is uncommonly employed in the context of identifying risk factors during the software design stage, makes this research unusual as well.

It is anticipated that this research will aid in the creation of software that is more successful and efficient. Additionally, this research can assist software developers in reducing risk and raising the caliber of the program they produce. It is envisaged that the C4.5 algorithm would make it simpler for developers to establish more effective and efficient risk management methods by identifying risk concerns at the software design stage. As a result, the findings of this study can have a big impact on the growth of the software business as well as software developers.

2. RESEARCH METHOD

The classification of elements at the risk stage of software design using the C45 method is a key objective of this study. Problem finding, the first phase of this research, focuses on the difficulties experienced by software developers during the design phase. This research was also supported by a review of pertinent literature to get a deeper understanding of this problem, which strengthened the research foundation based on earlier investigations.

Additionally, this project involves the supervision of 100 software developers who are dispersed across 4 continents. India, the United Arab Emirates, Canada, Germany, the United States, Singapore, and other nations participated in the study. To assure the accuracy and authenticity of the data, a crucial pre-processing stage is necessary before the data can be analyzed using the C4.5 algorithm. The C4.5 method, one of the most widely used and successful algorithms for classifying data, will next be used to handle the processed data.

To better understand the risk variables at the software design stage, the outcomes of the data processing stage using the C4.5 algorithm will be carefully assessed. The review will include a thorough analysis of the collected data, which will give software developers and other stakeholders important new information. There are various steps that must be taken to accomplish this goal, and each one is thoroughly outlined in Figure 1.

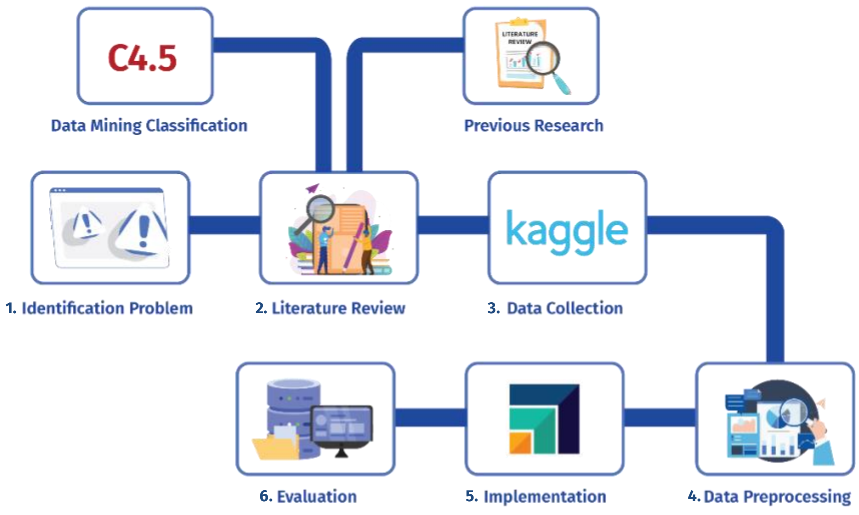

Figure 1. Research Stage

2.1. Problem Identification Stage

We conducted extensive data analysis on datasets linked to new risk factors in software development that were downloaded from the kaggle.com platform at this early stage. The dataset contains a variety of issues that are frequently encountered during the software design phase, including issues with requirements documents (RD), incorrect architecture designs (AD), programming languages (PL), physical model activities (PM), problems with design verification (VD), issues with defining design activities (SD), and issues with properly documenting design activities (DD).The dataset is then statistically analyzed to determine the degree to which these risk factors are present and have an impact on software development. We can see patterns and trends in the analysis that correspond to each risk factor. For instance, we discovered that problems with requirements documents (RDs) were one of the most prevalent risk factors, indicating that shortcomings or inaccuracies in requirements papers could result in mistakes throughout the software design stage. Additionally, we discovered that incorrect architectural design (AD) issues and programming language (PL) issues both significantly affect risks during the software design stage.

We subsequently analyzed the dataset after selecting the most significant risk indicators to determine how these risk factors related to other dataset characteristics including the geographic location of software developers, their expertise, and the size of the project teams. This study aids in the discovery of distinctive patterns and distinctive traits that could affect risk factors during the software design phase. We can create more potent plans to lower risk and raise the caliber of software design if we have a deeper understanding of risk variables and how they relate to other aspects.

2.2. Literature Study, Prior Research, Data Mining, and Classification

The second thing we did was perform a complete literature review by looking for pertinent references in a variety of places, such as books, the internet, journal articles, and seminar sessions. Finding solutions to the issues faced by software developers requires us to use the findings of earlier studies as one of our primary references. We also employ data mining methods in this endeavor, particularly classification algorithms. Our literature review offers a greater grasp of the risk variables present during the software design phase and past solutions put forth by other researchers. We gather a variety of statistics regarding the methods applied, the data gathered, and the outcomes and conclusions reached. Our research has a solid theoretical base thanks to references from textbooks. We employ data mining techniques as a successful strategy in our efforts to address the challenges faced by software engineers. We may examine and analyze huge and complex data using data mining to find pertinent patterns, trends, and relationships. We concentrate on classification algorithms in the context of our research since they enable us to categorize risk elements throughout the software design phase. The C4.5 algorithm is one of the classification algorithms that we decided to use. A good decision tree approach for classifying data is the C4.5 algorithm. We can create a decision tree that will assist us in accurately classifying risk variables by utilizing the C4.5 algorithm. The C4.5 algorithm has the advantage of being able to handle attributes of various data types, including continuous and categorical attributes.

2.3. Data Collection

The procedure of gathering data is the next phase. We collected online quizzes and surveys to gather data in 2022, and we used the dataset from the website kaggle.com [17]. This dataset comprises of 100 samples encompassing different qualities that are dispersed throughout numerous different nations. The information in the data comprises details like User_Id, Gender, Job Profile, Experience (in years), Project Type, Organization Size, Country, and Issues found because of risk factors during the software design phase. The information listed below was taken from Table 1 and used in this investigation.

Table 1. Software design phase risk factor data set

Column	Description
Gender	Contains details about the gender and gender identity of those working on the project.
Job Profile	Provides details about the job descriptions or job titles of those working on the project.
Experience	Provides details about the project participants' degree of experience, such as years of experience.
Project Type	Contains a description of the project or dataset's subject, as well as a discussion of any issues or difficulties that were experienced.
Organization size	Provides details on the organization's or firm's size, such as whether it is a small, medium, or large company where the project's participants work.
Country	Describes the nation where the people or organizations working on the project or dataset are situated.
The problem	Contains an account of the issues or difficulties the project encountered.
Requirements Document (RD)	Describes the documentation or requirements that must be satisfied for the project to be completed.
Improper Architectural Design (AD)	Information about any flaws or mistakes in the project's software architectural architecture.
Programming Language (PL)	Details about the project's programming language.
Physical Model Activity (PM)	Explains the project-specific activities involved in creating physical models or hardware components.
Verifying Design Activity (VD)	This area may provide details regarding procedures for confirming or validating the software or hardware designs used in the project.
Specifying Design Activity (SD)	Includes details on the tasks involved in describing or defining the hardware or software design for a project.
Documenting Design Activity (DD)	Includes information on the processes involved in documenting or producing documentation pertaining to the project's software or hardware design.

2.4. Data Preprocessing

The data will be cleaned by the researcher, who will also take outliers and missing numbers. Data transformation can be used to adapt data to the needs of an algorithm. Techniques for dimensionality reduction can make complicated datasets simpler. In order to improve model performance and dataset simplicity, irrelevant features will be assessed and perhaps eliminated.

2.5. Application of the Decision Tree Method

Following the initial data processing stage, we carried out the major analysis step by using the C4.5 algorithm. A powerful decision tree algorithm for categorizing data is algorithm C4.5. We now utilize the C4.5 method to construct a decision tree that will assist us in categorizing risk concerns during the software design phase. The first step in putting the C4.5 algorithm into practice is choosing the property that is the most informative as the separator attribute at each stage of creating a decision tree. The C4.5 algorithm employs computation techniques like information gain and information gain ratio in order to choose the most informative feature. Using this strategy, we may assess and choose the characteristics that have the greatest impact on separating the data and arriving at the best conclusion.

The root attribute in the C4.5 method is selected based on the current attributes' highest gain values. computing the entropy must come before computing the value gain. The analysis of the casing set's originality or variety is gauged using entropy [18]–[20].
Entropy calculation is done using the following Formula (1):

$$Entropy\ (S) = \sum - n\ i{-}1\ pi * \log 2\ pi \qquad (1)$$

S is the set of cases being considered, n is the total number of partitions in S, and pi is the ratio of each partition Si to S in this formula.
Additionally, the acquisition value is determined using the Formula (2) below:

$$Gain\ (S, A) = Entropy\ (S) - \sum |Si||S|ni{-}1 * Entro\ py\ (Si) \qquad (2)$$

In this formula, |Si| is the number of cases in the Si partition, and |S| is the number of cases in the set S. S is the set of instances being considered, A is the attribute being evaluated, n is the total partition arising from attribute A, and S is the number of cases in the set S. The C4.5 method can determine the entropy and value gain for each potential attribute using these formulas, allowing for the most insightful choice of the root attribute to construct a successful decision tree.

The C4.5 algorithm will then create a decision tree based on the chosen separator properties [21]. Iteratively, decision trees are developed, with each stage dividing the data into smaller subgroups according to the chosen attribute values. The decision tree will keep growing until it either hits a stop condition or a condition where all of the data in that subset belong to the same class. The C4.5 algorithm additionally prunes while building the decision tree to prevent overfitting [22]. Tree branches that don't significantly contribute to classification performance are removed during pruning. Pruning allows us to create a decision tree that is both simpler and well-classifiable. Once the decision tree has been created, we may utilize it to categorize newly discovered data. Every node in the decision tree contains rules and decisions that must be followed in order to carry out the categorization process. These nodes will be traversed by fresh data until it hits a leaf that specifies the relevant class or classification.

2.6. Evaluation

Our study also includes a comprehensive evaluation step to assess how well the C4.5 algorithm is working [23]. To evaluate the quality and efficiency of the C4.5 algorithm in categorizing risk factors at the software design stage, a variety of evaluation measures, including level of accuracy, precision, and recall are considered. The classification model's performance in research will be evaluated by RapidMiner using the formulas for accuracy (Formula 3), precision (Formula 4), recall (Formula 5).

$$accuracy = \frac{TP+TN}{TP+FP+FN+TN} \qquad (3)$$

$$precision = \frac{TP}{TP+FP} \quad (4)$$

$$recall = \frac{TP}{TP+FN} \quad (5)$$

Important metrics including True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) are included in the classification model's evaluation measure. When the model accurately detects existent occurrences, the classification results are said to be true and correct, or TP. An inaccurate classification result is referred to be TN when the model accurately detects the absence of an event that is not present. The model incorrectly detects a categorization result that the FP contains even though it shouldn't. When the model fails to identify actual events, the classification results are not exact and are included in FN.

We can learn more about how well the classification model performs in classifying data by measuring it using these measures. Accuracy, precision, and recall are calculated based on these measures. The degree of accuracy indicates how well the model can categorize both events and non-events. The degree of accuracy of the model's positive classification results is referred to as precision. Recall, often referred to as sensitivity, gauges how well a model can pinpoint each and every occurrence.

3. RESULTS AND ANALYSIS

The C4.5 decision tree technique is used in this study to categorize the risk factors in each section's software design stage. RapidMiner, which has been employed, is used to reference earlier studies [24]–[26]. Data reading, filling in for missing numbers, and data splitting are the first steps. In the comparison, training data make up 70% and in-line test data make up 30%.

In this study, the risk variables related to the software design stage are identified using the C4.5 algorithm in RapidMiner. In this procedure, we gather information about finished software projects and pre-process it to make sure the dataset is accurate and comprehensive. The C4.5 method is then used to build model predictions utilizing the stages in RapidMiner. We train the model and evaluate its risk to see how well it can recognize relevant factors by utilizing training and testing subsets.

Our evaluation's findings provide important light on the dangers that have a big impact on the software design phase. Figure 2 illustrates the procedure for detecting software design risk concerns.

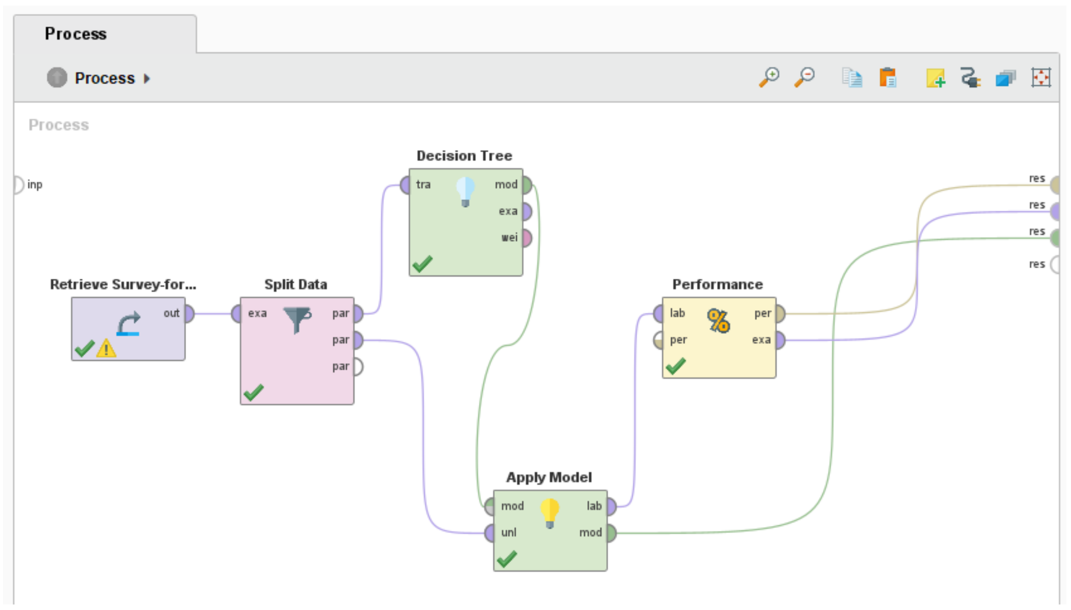

Figure 2. Risk factor classification model of software development with algorithm C4.5

The decision tree results produced by RapidMiner's C4.5 algorithm are shown in Figure 3. The decision criteria used in data classification are graphically represented by a decision tree. The decision tree is depicted in this picture with branches that describe the many attribute values utilized to make decision-making decisions.

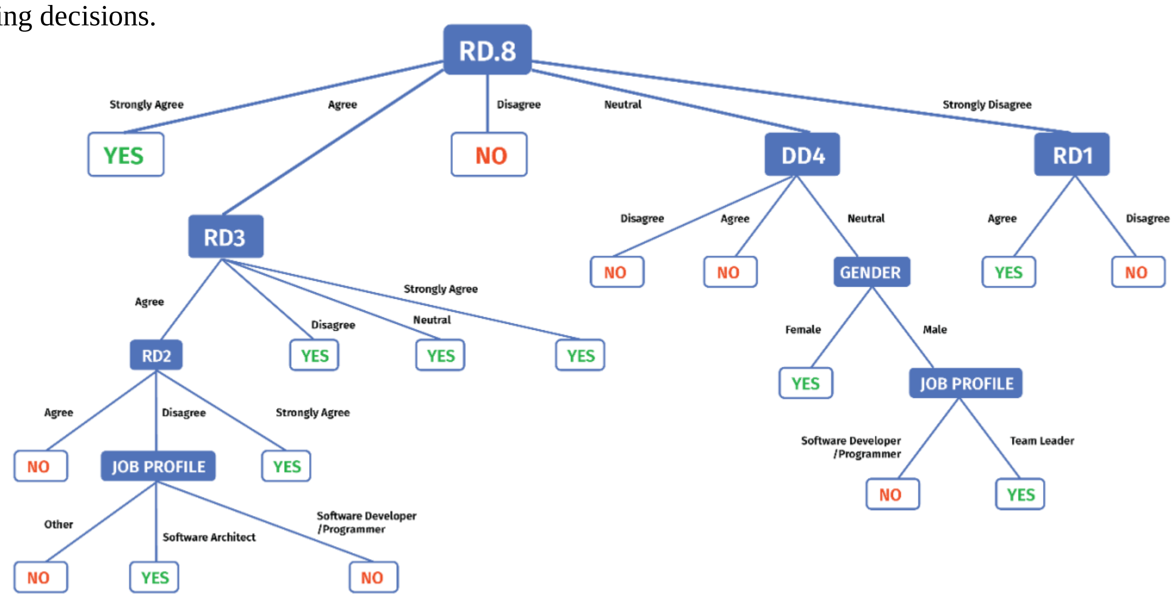

Figure 3. Model decision tree using the C4.5 algorithm

The interpretation of the data in Figure 3 because of the use of Algorithm C4.5 in this study provides a thorough grasp of the risk variables found during the software design stage. According to research, success in software design dramatically slows down positive resistance to requirements documents (RD8). In addition, when respondents agree strongly with RD8, there is a strong likelihood that questions about software design will obtain affirmative answers. Furthermore, success in software design is highly tied to good replies to specific requirements document sections, such as RD3 and RD2. Respondents who agreed with RD3 and RD2 were more inclined to give favorable responses to software design questions. As a result, this variable plays a crucial role in identifying risk issues throughout the software design stage. The importance of design documentation (DD4) in software design risk was also established. A good response to design documentation undermines success in software design. As a result, this study emphasizes the significance of effective documentation in risk management during the software design stage.

Furthermore, the "Neutral" condition RD8 reveals differences in reactions between male and female respondents toward design documentation. This could imply that responses to design documentation are influenced by gender variables, which should be considered in risk reduction efforts. The response to RD1 influenced software design as well, particularly when RD8 was graded "Strongly Disagree." Respondents who agree with RD1 are more likely to give affirmative answers to software design questions. As a result, consensus on early design concepts has a substantial impact on how software is perceived overall. Overall, the findings of this interpretation provide a comprehensive picture of the link between certain variables and risk during the software design stage. This understanding can serve as the foundation for more effective risk mitigation measures and better decision making in software development.

The outcomes of the performance assessment for the C4.5 classification model are displayed in Table 2 and Figure 4. This demonstrates that the C4.5 algorithm's average accuracy value is 99.3%, and a performance vector analysis is performed using the confusion matrix's recall and precision metrics.

Table 2. Results of Confusion Matrix's using the C4.5 algorithm

	True Yes	True No
Pred. Yes	21	2
Pred. No	0	7

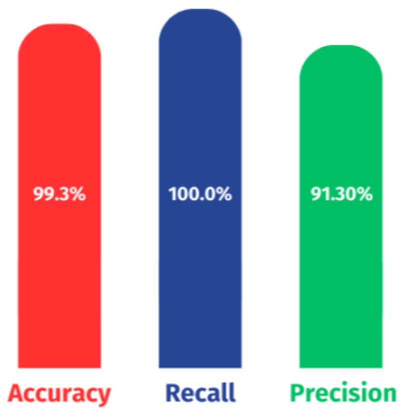

Figure 4. The results of the accuracy, recall and precision metrics using the C4.5 algorithm

Based on the findings in Tables 2 and Figure 4 above, it is possible to infer that this study was successful in recognizing risk variables throughout the software design stage with a high level of accuracy, reaching 93.33%. This accomplishment illustrates the extent to which the C4.5 Algorithm can be effective in developing a model capable of accurately classifying risk factors. This model was successful in classifying most of the data, as evidenced by the Confusion Matrix, where the proportion of correct predictions (true positive and true negative) was far more than the proportion of incorrect predictions (false negative). The recall rate is important in this study since it demonstrates the model's capacity to detect occurring risk scenarios. The "Yes" category (risk event) has a 100% recall rate, suggesting that the algorithm is fully capable of identifying actual risk factors. However, the model has a propensity to ignore possible dangers that do not exist, as evidenced by the recall rate of 77.8% for the "No" (no risk) category.

Aside from that, accuracy is a key criterion to consider while evaluating model performance. The precision of the "Yes" category is 91.30%, indicating that the model's "Yes" predictions are mostly correct. This suggests that the model can make accurate "Yes" predictions. Meanwhile, the precision of the "No" category reached 100%, indicating that the model's "No" predictions were correct. Overall, these findings indicate that the model developed using the C4.5 Algorithm can be a useful tool for detecting risk factors throughout the software design stage, with a fair level of accuracy and a good balance of recall and precision in classifying risk categories.

4. CONCLUSION

It is clear from the findings and discussion above that the goal of this study is to identify risk variables that may have an impact on software design in the first phases of development. This study was successful in creating a decision tree that offers significant insights into the risk factors that affect the success of software design by using the C4.5 Algorithm to the gathered data. The C4.5 Algorithm's decision tree generates recommendations for the variables that should be considered when designing software. The outcomes of decision tree interpretation demonstrate that replies to design challenges, such as the problem of documenting design activity (DD4), as well as responses to requirements document (RD), such as RD3 and RD8, have a substantial impact on the success of software design. With the help of this conclusion, software engineers can take the necessary precautions to lessen or eliminate potential risks that might exist during the program design phase.

In addition, it was discovered that the model created using the C4.5 Algorithm has a high level of accuracy (93.33%) from the findings of the performance model interpretation. This demonstrates that the model can classify risk factors accurately during the software design phase. The high levels of recall and precision suggest that this model can offer significant insights in comprehending the risks that may occur during the software design stage, even though there is a tiny propensity to miss dangers where there are none. Overall, this study significantly advances our understanding of risk variables at the software design stage and offers significant takeaways for academics and industry professionals working in the field of software development. The findings of this study can be utilized as a foundation for determining the best course of action for minimizing these risks and enhancing the effectiveness of the overall software design.

REFERENCES

[1] A. Akhtar, B. Bakhtawar, and S. Akhtar, “EXTREME PROGRAMMING VS SCRUM: A COMPARISON OF AGILE MODELS,” *International Journal of Technology, Innovation and Management (IJTIM)*, vol. 2, no. 2, Oct. 2022, doi: 10.54489/ijtim.v2i2.77.

[2] J. Segura, “The Teaching of Usability in Software Development: Case Study in the Computer Engineering Career at the University of Matanzas,” *International Journal of Engineering Pedagogy (iJEP)*, vol. 11, no. 1, p. 4, Jan. 2021, doi: 10.3991/ijep.v11i1.14837.

[3] N. T. Al-Qemaqchi, “Transformation of Architectural Design Concepts During the Early Design Phase,” *International Journal of Engineering Pedagogy (iJEP)*, vol. 12, no. 6, pp. 85–99, Dec. 2022, doi: 10.3991/ijep.v12i6.31717.

[4] B. Kang, N. Crilly, W. Ning, and P. O. Kristensson, “Prototyping to elicit user requirements for product development: Using head-mounted augmented reality when designing interactive devices,” *Des Stud*, vol. 84, p. 101147, Jan. 2023, doi: 10.1016/j.destud.2022.101147.

[5] A. Strielkina and A. Tetskyi, “Methodology for assessing satisfaction with requirements at the early stages of the software development process,” *Radioelectronic and Computer Systems*, no. 1, pp. 197–206, Mar. 2023, doi: 10.32620/reks.2023.1.16.

[6] H. Edison, X. Wang, and K. Conboy, “Comparing Methods for Large-Scale Agile Software Development: A Systematic Literature Review,” *IEEE Transactions on Software Engineering*, vol. 48, no. 8, pp. 2709–2731, Aug. 2022, doi: 10.1109/TSE.2021.3069039.

[7] S. Zapata, F. Gallardo, G. Sevilla, E. Torres, and R. Forradellas, “Trust evaluation in virtual software development teams using BERT-based language models,” *J Comput Sci Technol*, vol. 23, no. 1, p. e04, Apr. 2023, doi: 10.24215/16666038.23.e04.

[8] A. Aslam *et al.*, “Decision Support System for Risk Assessment and Management Strategies in Distributed Software Development,” *IEEE Access*, vol. 5, pp. 20349–20373, 2017, doi: 10.1109/ACCESS.2017.2757605.

[9] Z.-N. Li, X.-H. Huang, T. Mu, and J. Wang, “Attention-Based Lane Change and Crash Risk Prediction Model in Highways,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 22909–22922, Dec. 2022, doi: 10.1109/TITS.2022.3193682.

[10] P. Sotiropoulos, C.-M. Mathas, C. Vassilakis, and N. Kolokotronis, “A Software Vulnerability Management Framework for the Minimization of System Attack Surface and Risk,” *Electronics (Basel)*, vol. 12, no. 10, p. 2278, May 2023, doi: 10.3390/electronics12102278.

[11] M. J. H. Faruk, S. Subramanian, H. Shahriar, M. Valero, X. Li, and M. Tasnim, “Software Engineering Process and Methodology in Blockchain-Oriented Software Development: A Systematic Study,” in *2022 IEEE/ACIS 20th International Conference on Software Engineering Research, Management and Applications (SERA)*, IEEE, May 2022, pp. 120–127. doi: 10.1109/SERA54885.2022.9806817.

[12] A. Baraldi, L. D. Sapia, D. Tiede, M. Sudmanns, H. Augustin, and S. Lang, “Innovative Analysis Ready Data (ARD) product and process requirements, software system design, algorithms and implementation at the midstream as *necessary-but-not-sufficient* precondition of the downstream in a new notion of Space Economy 4.0 - Part 2: Software developments,” *Big Earth Data*, pp. 1–118, Oct. 2022, doi: 10.1080/20964471.2021.2017582.

[13] A. Jadhav, M. Kaur, and F. Akter, “Evolution of Software Development Effort and Cost Estimation Techniques: Five Decades Study Using Automated Text Mining Approach,” *Math Probl Eng*, vol. 2022, pp. 1–17, May 2022, doi: 10.1155/2022/5782587.

[14] L. M. Alves, G. Souza, P. Ribeiro, and R. J. Machado, “Longevity of risks in software development projects: a comparative analysis with an academic environment,” *Procedia Comput Sci*, vol. 181, pp. 827–834, 2021, doi: 10.1016/j.procs.2021.01.236.

[15] A. Bombarda *et al.*, “Guidelines for the development of a critical software under emergency,” *Inf Softw Technol*, vol. 152, p. 107061, Dec. 2022, doi: 10.1016/j.infsof.2022.107061.

[16] A. R. Lendra and D. Firdaus, “IMPLEMENTATION OF C4.5 ALGORITHM TO ASSIST IN THE SELECTION OF FLOOR CONSTRUCTION PROJECTS,” *IJISCS (International Journal of Information System and Computer Science)*, vol. 4, no. 3, p. 153, Nov. 2020, doi: 10.56327/ijiscs.v4i3.947.

[17] Khan T, “Software Design Phase Risk Factors,” 2023. https://www.kaggle.com/datasets/asif05amu/software-design-phase-risk-factors (accessed May 30, 2023).

[18] P. Varalakshmi, N. Vasumathi, and R. Venkatesan, “Tropical Cyclone prediction based on multi-model fusion across Indian coastal region,” *Prog Oceanogr*, vol. 193, p. 102557, Apr. 2021, doi: 10.1016/j.pocean.2021.102557.

[19] B. F. Tanyu, A. Abbaspour, Y. Alimohammadlou, and G. Tecuci, “Landslide susceptibility analyses using Random Forest, C4.5, and C5.0 with balanced and unbalanced datasets,” *Catena (Amst)*, vol. 203, p. 105355, Aug. 2021, doi: 10.1016/j.catena.2021.105355.

[20] P. K R and N. N C, “Lung Cancer Survivability Prediction based on Performance Using Classification Techniques of Support Vector Machines, C4.5 and Naive Bayes Algorithms for Healthcare Analytics,” *Procedia Comput Sci*, vol. 132, pp. 412–420, 2018, doi: 10.1016/j.procs.2018.05.162.

[21] S. Moral-García, C. J. Mantas, J. G. Castellano, and J. Abellán, “Using Credal C4.5 for Calibrated Label Ranking in Multi-Label Classification,” *International Journal of Approximate Reasoning*, vol. 147, pp. 60–77, Aug. 2022, doi: 10.1016/j.ijar.2022.05.005.

[22] X. Meng, P. Zhang, Y. Xu, and H. Xie, “Construction of decision tree based on C4.5 algorithm for online voltage stability assessment,” *International Journal of Electrical Power & Energy Systems*, vol. 118, p. 105793, Jun. 2020, doi: 10.1016/j.ijepes.2019.105793.

[23] S. Lestari, Y. Yulmaini, A. Aswin, S. Sylvia, Y. A. Pratama, and S. Sulyono, “Implementation of the C4.5 algorithm for micro, small, and medium enterprises classification,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 6, p. 6707, Dec. 2022, doi: 10.11591/ijece.v12i6.pp6707-6715.

[24] M. Z. Naser, “Machine learning for all! Benchmarking automated, explainable, and coding-free platforms on civil and environmental engineering problems,” *Journal of Infrastructure Intelligence and Resilience*, vol. 2, no. 1, p. 100028, Mar. 2023, doi: 10.1016/j.iintel.2023.100028.

[25] S. Kim and H. Lee, “Customer Churn Prediction in Influencer Commerce: An Application of Decision Trees,” *Procedia Comput Sci*, vol. 199, pp. 1332–1339, 2022, doi: 10.1016/j.procs.2022.01.169.

[26] I. Garcia-Magarino, G. Gray, R. Lacuesta, and J. Lloret, “Survivability Strategies for Emerging Wireless Networks With Data Mining Techniques: a Case Study With NetLogo and RapidMiner,” *IEEE Access*, vol. 6, pp. 27958–27970, 2018, doi: 10.1109/ACCESS.2018.2825954.