ABMJ: An Ensemble Model for Risk Prediction in Software Requirements

Mohammad Mahmood Otoom

Department of Computer Science and Information, College of Science in Zulfi, Majmaah University,
Al-Majmaah 11952, Saudi Arabi

Summary
Due to the rising complexity of software projects, it is quite difficult to predict the risk in software requirements which is the most profound and essential activity in SDLC. It may lead to the failure of a software project. Risk prediction in software requirements is more crucial as it is the start of any software project. In this study, we propose an ensemble classifier based on AdaBooostM1 and J48 combinedly named as (ABMJ), for risk prediction in software requirements. The performance of the proposed ABMJ is compared with seven diverse ML algorithms including A1DE, MLP, CSF, J48, NB, RF, and SVM. These ML models are evaluated on the risk dataset available at Zenodo repository based on the accuracy, MCC, F-measure, recall, and precision. The overall analysis shows the best performance of ABMJ with an accuracy of 97.6285 % and the worst performance of MLP with an accuracy of 62.0553%. This study's analysis may be used as a standard for other academic studies, allowing the outcomes of any proposed approach, framework, or model to be benchmarked and essentially established.

Keywords
Software Requirements, Risk in Requirements, Machine Learning, Decision Tree, Random Forest, Support Vector Machine

1. Introduction

Requirement Engineering (RE) is a well-organized and systematic approach to gathering users' requirements for a software system [1]. Lately, we have seen a developing enthusiasm for software systems that can screen their condition and, if important, change their requirements to keep on satisfying their purpose [2]. This specific sort of software usually comprises a base system liable for the fundamental functionalities, alongside a part that screens the base system, examines the data, and afterwards responds suitably to ensure that the system keeps on executing its necessary functions. In software development, RE is considered as the most fundamental stage that is essentially concerned about the way toward documenting, eliciting and keeping up the stakeholders' requirements [3]. Regularly, meeting and making sure about stakeholders' centre requirements is one of the main causes behind delivering a decent quality of software system [4].
There is consistently a chance of inexact procedures during the time spent in the Software Development Life Cycle (SDLC) which may prompt likely defeat of software organization or software development. These questionable procedures are known as software risks. The risks burst from various risk influences that are established in an assortment of exercises in the SDLC. If these risks are not distinguished appropriately, they may get liable for the disaster of the project [5][6]. These elements should be separated and moderated to restrict the software cost and schedule by risk estimations in the SDLC's underlying phases. Because requirement collection is the first part of SDLC, forecasting risks at this stage may boost software productivity and quality while decreasing the likelihood of project disasters [4][5][7].
The risks fundamentally affect software requirements. They end up being the justification for harm to the stakeholders or software. Therefore, risks have to be predicted earlier in SDLC to improve the chances of success of the projects. Because risk assessment at this point will be more advantageous and will increase the software's productivity. When risks are appropriately handled, it also helps to reduce the likelihood of software project failure. Frequent solutions for the prediction of software risk at different phases in SDLC are available up till, whereas infrequent methods are available to predict risks in the software requirements phase in the literature [5][8]. A Risk prediction model encompasses classification methods that are projected to envisage risks on the Software Requirement Specifications (SRS) of the project.
Keeping the aforesaid issue in mind identified with risk prediction at the starting phase of software, researchers evaluated and developed various models using several classification techniques. However, sorting any extensive range planning to offer the convenience of these techniques is difficult. Completely, it was established that, despite certain differences in the experiments, no single model outperforms other techniques slantingly on varied data.
This work, on the other hand, introduced an ensemble model ABMJ for risk prediction in software requirements. The suggested model is compared to seven ML techniques: Multilayer Perceptron (MLP), Average One Dependency Estimator (A1DE), Cost-Sensitive Decision Forest (CSF), Naive Bayes (NB), J48 Decision Tree (J48), Support Vector Machine (SVM), and Random Forest (RF). The experimental results of each approach are compared to one another. Matthew's correlation coefficient (MCC), precision (Pr), recall (Re), F-measure (FM), and accuracy

Manuscript received March 5, 2022
Manuscript revised March 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.3.93

(ACC) are used as assessment tools to evaluate each technique.
The major contributions of this research are as follow:
We proposed an ensemble model for risk prediction in software requirements based on AdaBoostM1 and J48.
We compare the proposed model with seven ML techniques (A1DE, MLP, CSF, J48, NB, RF, and SVM) for risk prediction in software requirements.
We conduct a series of experiments on the risk prediction dataset available on the Zenodo repository.
To provide insight into the experimental results, evaluation is carried out using Pr, Re, FM, MCC, and Acc.
Hereinafter, Section 2 details the research methodology, and Section 3 presents the research framework. Whereas, Section 4 details the analysis and discussion of the results, and Section 5 concluded the study.

2. The Research Method

This exploration expects to determine the prediction of late identification of risks in software projects and their impact on the quality, timetable, and spending plan of the going through software project. Since the latest risk prediction models can quantify risks in the forthcoming phases, normally from the software designing stage or code of the SDLC, thus, these methodologies can distinguish risks, however, have restricted capacity in staying away from these risks from happening [5][9]. Risks in a software project are triggered via numerous aspects throughout the SDLC that leads to failure of the software project [8], [10]–[11]. The biggest reasons for software failure are technical difficulties, which are the result of fewer software engineering principles, theories, and procedures. These vulnerabilities should be addressed as soon as possible to decrease the possibility of the software project failing abruptly. The major characteristics of the software project that will be improved are a general nature, timeline, and budget of the project by anticipating risks. The risk forecast model will assist in recognising the risk level of an instance (software requirements) of another organisation by using a risk dataset. The project/risk manager, on the other hand, will aim to alter and control the entire risk prediction process. The basic notion of risk prediction using classification approaches has been presented. This model is comprised of four primary components, which are represented in Figure 1 and explored in further detail below:

- Risk Identification

Risk identification is considered the first stage in the risk prediction model, where the risk/project manager will separate the requirements generally, and it is conducted using a "checklist." SRS requirements including risk threats were marked and examined for further investigation. When the checklist was completed, It made a beeline for the next stage [12], [13].

- Risk Analysis

During this step, a classifier uses a risk dataset to analyse and validate requirements. A1DE, MLP, CSF, J48, NB, RF, and SVM are evaluated to attain the most suitable classifier for risk-associated situations [5]. The classification method is chosen based on its better precision as compared to other classifiers conversed in the subsequent section.

- Risk Prioritization

This is the model's output step, when the studied risk is prioritised, with high likelihood and high effect risks being moved to the top of the list and low probability and low impact risks being pushed to the bottom [7].

- Requirement risk Dataset

The risk processes for the software requirements dataset is accessible on the Zenodo repository datasets1 [14].

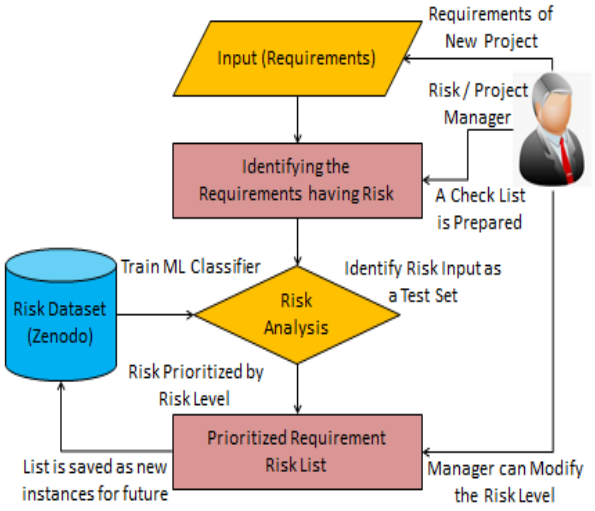


Fig. 1 Requirement Risk Prediction Model

3. Proposed Model

The proposed model is based on AdaBoostM1 and J48 (ABMJ). AdaBoost is a meta-algorithm that can be used to improve the effectiveness of various classifiers. It works based on the iterative running of a particular base or weak model on diverse disseminations on training data. After that, it combined the base learners into a single model. The working mechanism of the proposed AdaBoostM1 is illustrated in Figure 2 (A), while Figure 2 (B) and Figure 2 (C) respectively presents the proposed J48 and ensemble model. We tuned the number of parameters for each classifier to optimize the performance of the model.

[1] https://zenodo.org/record/1209601#.Xpa9mUAzZdg

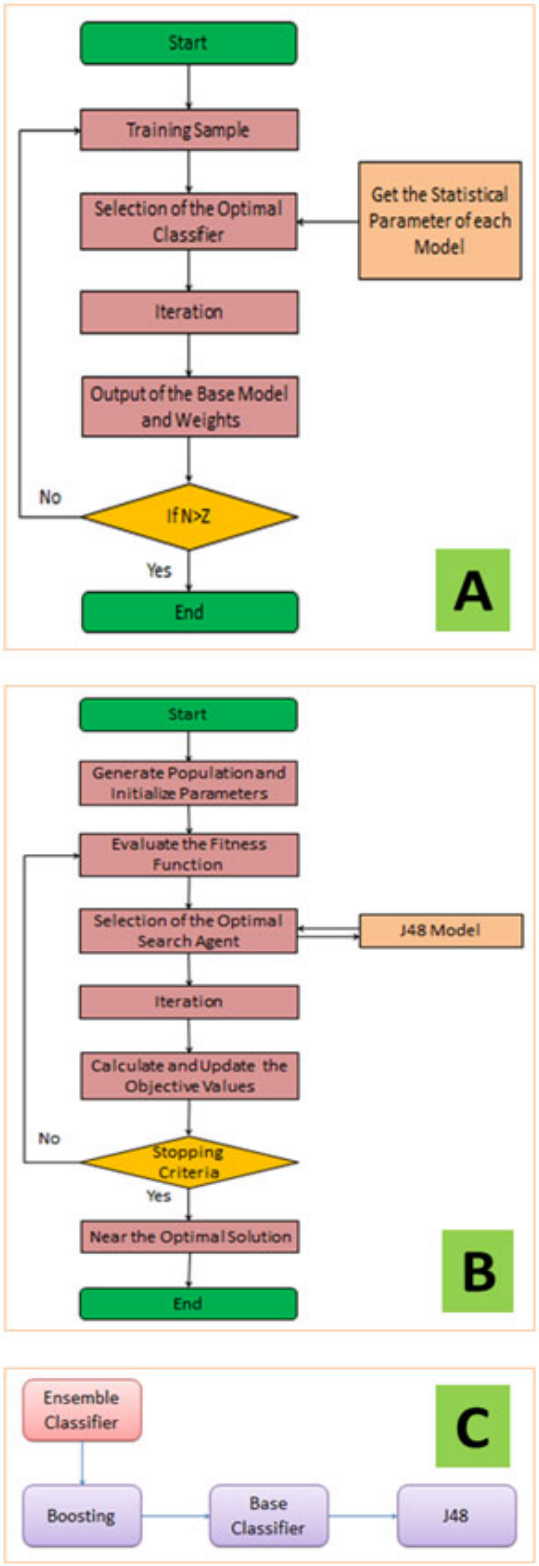


Fig. 2 Flowcharts of AdaBoostM1, J48 and Ensemble Model, (A) AdaBoostM1, (B) J48, (C) Ensemble

Parameters for AdaBoostM1: The batchSize selected for AdaBoostM1 is 100. The classifier inside AdaBoostM1 is J48. Number Decimal Places are 2, while the number of iterations is selected 10. The seed value is 1 and the weight threshold selected is 100.

Parameters for J48: The seed value and batch size, number of decimal places are the same for AdaBoostM1 and J48. The confidence factor value is 0.25, while the minimum number for objective is 2. Total numbers of folds for J48 are selected 3.

4. The Research Framework

The dataset for risk prediction in software requirements, includes the features that are associated to the requirements and risks of the software project. These risks destructs the success of software development. This dataset contains 13 attributes and 253 instances with 5 levels of risks identified. The count and weight of each risk level are presented in Figure 3 while the list of attributes is presented in Table 1. 10-fold cross validation is used for data training and testing, which is a standard criterion [15][16]. The requirements set is utilised as input to the model in the projected model, and the model outputs the amount of risk in the requirements. These results will be used by the project manager or domain expert to easily prepare and minimise risks sooner. Based on the requirements, the project manager or domain expert has the right to delete, add, or change the findings. The research is separated into three stages: software risk prediction model, dataset filtration, and model validation. In the sections that follow, these stages are further addressed. The proposed study framework is depicted in Figure 4.

Table 1: List of Attributes with Distinct and Types

S. No.	***Name***	***Distinct***	***Type***
1	Requirements	292	Categorical
2	Project Category (PC)	4	Categorical
3	Requirement Category (RC)	10	Categorical
4	Risk Target Category (RTC)	22	Categorical
5	Probability	81	Numeric
6	The magnitude of Risk (MR)	7	Categorical
7	Impact	5	Categorical
8	Dimension of Risk (DR)	13	Categorical
9	Affecting No of Modules (ANoM)	9	Numeric
10	Fixing Duration (Days)	12	Numeric
11	Fix Cost (% of Project)	10	Numeric
12	Priority	293	Numeric
13	Risk Level (RL)	5	Categorical

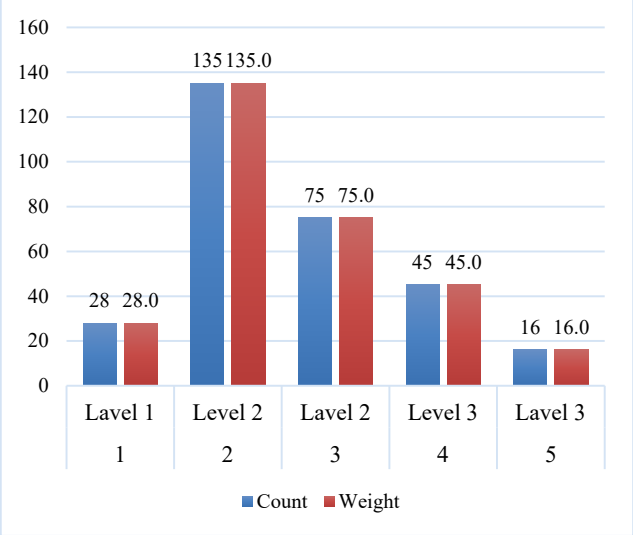


Fig. 3 Count and Weight of each Level (Class)

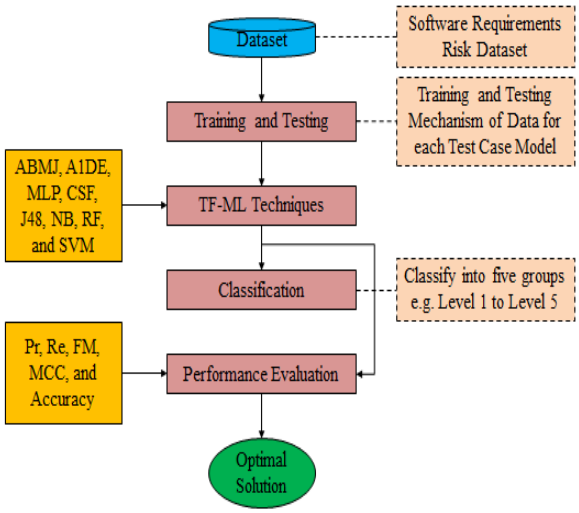


Fig. 4 Research Framework

4.1 Model Evaluation and Comparison

A model is proposed based on AdaBoostM1 and J48 for risk prediction in software requirements. The outcomes of the proposed ABMJ are compared with seven different ML techniques including Cost-Sensitive Decision Forest (CSF), Naïve Bayes (NB), Multilayer Perceptron (MLP), J48 Decision Tree (J48), Average One Dependency Estimator (A1DE), Random Forest (RF), and Support Vector Machine (SVM). The 10-fold cross-validation mechanism is used for data training and testing. Overall models are evaluated using precision (Pr), recall (Re), F-measure (FM), Matthew's correlation coefficient (MCC), and accuracy (Acc). All these assessment measures can be calculated using the following equations where δ presents the true positive predictions, β presents the false positive prediction, α presents the true negative and μ presents the false-negative predictions.

Precision: It is the total of positive forecasts divided by the sum of positive class values projected. It can be calculated as:

$$Pr = \frac{\delta}{\delta+\beta} \quad (1)$$

Recall: It is demarcated as the proportion of true positive units with high estimation to the sum of positive modules. It can be found as:

$$Re = \frac{\delta}{\delta+\mu} \quad (2)$$

F-Measure: It is also known as F1-Score. F1-score carries the balance between precision and recall. It can be assessed as:

$$FM = \frac{2 * Pr * Re}{Pr + Re} \quad (3)$$

MCC: It is a correlation coefficient measured using all four values in the CM. This can be found as:

$$MCC = \frac{(\alpha*\delta)-(\mu*\beta)}{\sqrt{(\beta+\delta)(\mu+\delta)(\alpha+\beta)(\alpha+\mu)}} \quad (4)$$

Accuracy: It is the opinions to that how much the prediction is accurate. It can be calculated as:

$$Acc = \frac{\delta+\alpha}{\delta+\alpha+\beta+\mu} \quad (5)$$

4.2 Employed Techniques

ML methods are now widely used in a number of sectors to extract valuable information from massive volumes of data. ML applications are employed in a range of real-world scenarios, including cyber-security, bio-informatics, social network community recognition, and enhancing development processes to build high-quality software systems [17]. The subsections that follow provide a quick overview of the ML approaches used in this study.

A. Average One Dependency Estimator

It is a probabilistic model that is commonly used for classification challenges. It thrives in colossally precise categorization by being an average of a tiny space of many NB-like models with lesser disinterest promises than NB [18].

B. Multilayer Perceptron

The most important modules of a neural network (NN) are an input layer, at least one hidden layer, and an output layer [19][20]. When data is given as the input layer for a NN, the network neurons begin deviousness in the succeeding layer until an output value is produced at each of the output neurons [21][22].

C. Cost-Sensitive Decision Forest

As an optional component of the extract used by the C4.5 decision tree, CSF executes a cost-sensitive extract, which cuts the tree if the reliable sum of misclassification for future minutes does not expand considerably as a result of

the extract. Furthermore, unlike the Cost-Sensitive Decision Tree (CS-T), the CSF contains a tree that must evolve entirely before being retrieved [23][24].

D. J48 Decision Tree

The J48 decision tree is a more advanced variant of the C4.5 decision tree. This strategy employs divide-and-conquer tactics. To make a tree, it executes the clipping procedure. J48 is a company-wide approach for entropy or information-gathering activities [25][26].

E. Naïve Bayes

The term "naive" references preventive individuality amongst features. The "naive" assumption losses calculation complexity to a simple growth of probabilities. That's for the intention that it is the simplest method within classification models. As an influence of this sincerity, it can promptly indenture with an informational index with profuse facilities [27][28].

F. Random Forest

This approach classifies all trees in the forest by preparing the anticipation of the tree structure, which is then analysed using the same diffusion and random vector values [28][24].

G. Support Vector Machine

It is a supervised learning approach with applications in categorization, pattern recognition, and bio-photonics [30]. It was created with binary classification in mind, but it may be used to a wide range of classification [31]. In binary classification, the main idea of SVM is to define a line across classes of data in order to use the distance between data points sitting next to it as a criterion. If the data is linearly inseparable, mathematical functions are employed to transform it to a higher-dimensional space, where it may be divided linearly [32][33].

5. Analysis and Discussion on Exam Results

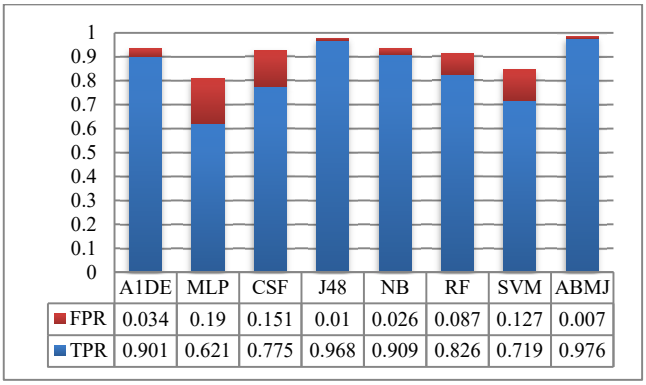
This study focuses on seven diverse ML classification techniques for the prediction of risk in software requirements. The techniques are evaluated on a dataset taken from the Zenodo repository using multiple assessment measures. All the aforementioned techniques are modeled based on some parameters. The common parameters in all techniques are batch size = 100, number decimal place = 2, and random seed value =1. Some parameters that are different for individual techniques are:
AIDE used frequency limit = 1, and weight =1.0.
MLP used hidden layer =3, learning rate = 0.3, momentum = 0.2, and training time = 500.
CSF used confidence =0.25, cost goodness = 0.2, minRectLeaf = 10, numTrees = 60, and separation = 0.3.
J48 used confidence = 0.25, minNumObj = 2, and numFolds = 3.
NB used only the common tuning parameters that are batch size and number decimal place.

Table 2: Confusion Matrix Value Achieved through each Technique

Technique	*Class*	*a*	*b*	*c*	*D*	*e*
A1DE	Level 5 = a	13	0	0	0	1
	Level 2 = b	1	106	3	2	1
	Level 3 = c	2	4	55	0	2
	Level 1 = d	0	1	1	23	0
	Level 4 = e	2	1	3	1	31
MLP	Level 5 = a	0	0	11	0	3
	Level 2 = b	0	98	15	0	0
	Level 3 = c	0	11	52	0	0
	Level 1 = d	0	24	0	1	0
	Level 4 = e	0	0	32	0	6
CSF	Level 5 = a	7	1	0	0	6
	Level 2 = b	0	112	1	0	0
	Level 3 = c	0	15	47	0	1
	Level 1 = d	0	24	0	1	0
	Level 4 = e	0	3	6	0	29
J48	Level 5 = a	14	0	0	0	0
	Level 2 = b	0	111	2	0	0
	Level 3 = c	0	0	61	0	2
	Level 1 = d	0	1	0	24	0
	Level 4 = e	1	0	2	0	35
NB	Level 5 = a	12	0	0	0	2
	Level 2 = b	0	100	9	4	0
	Level 3 = c	0	1	60	0	2
	Level 1 = d	0	1	0	24	0
	Level 4 = e	2	0	2	0	34
RF	Level 5 = a	1	1	1	0	11
	Level 2 = b	0	112	1	0	0
	Level 3 = c	0	1	61	0	1
	Level 1 = d	0	17	1	6	1
	Level 4 = e	0	1	8	0	29
SVM	Level 5 = a	5	0	1	0	8
	Level 2 = b	0	100	10	3	0
	Level 3 = c	0	13	44	0	6
	Level 1 = d	0	14	0	11	0
	Level 4 = e	5	0	11	0	22
ABMJ	Level 5 = a	14	0	0	0	0
	Level 2 = b	0	111	2	0	0
	Level 3 = c	0	0	61	0	2

	Level 1 = d	0	1	0	24	0
	Level 4 = e	1	0	0	0	37

RF used bagSizePercent = 100, numExcecutionSlots = 1, and numIterations = 100.
SVM used epsilon = 1.0E-12, kernel = polyKernel, and toleranceParameter = 0.001.
All of the aforementioned metrics are generated using a confusion matrix (CM) to analyse the outcome. Table 2 presents the CM for each approach. First, the true positive rate (TPR) and false positive rate (FPR) are computed. Figure 5 depicts the TPR and FPR outcomes of each approach. The study reveals that suggested ABMJ performs best with a TPR of 0.976 and MLP performs worst with a TPR of 0.621.



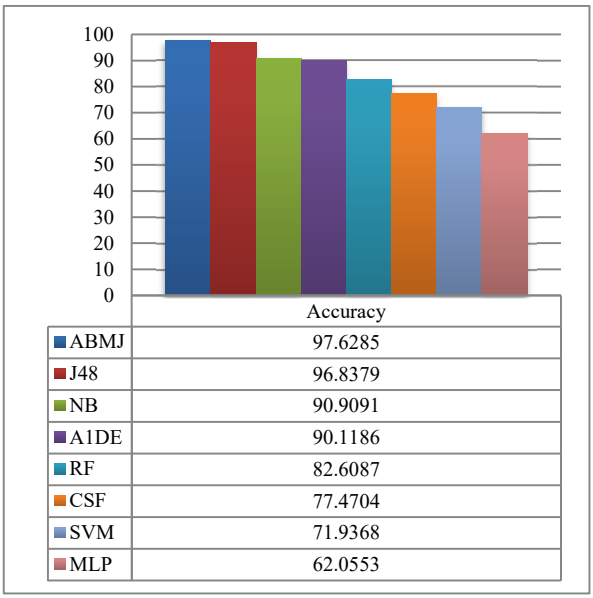
	A1DE	MLP	CSF	J48	NB	RF	SVM	ABMJ
FPR	0.034	0.19	0.151	0.01	0.026	0.087	0.127	0.007
TPR	0.901	0.621	0.775	0.968	0.909	0.826	0.719	0.976

Fig. 5 TPR and FPR Analysis of ML Techniques

The Acc of individual technique is evaluated using equation number 5. The Acc analyses are presented in Figure 6. These analyses show the best performance of proposed model with the accuracy of 97.6285%, and the worst performance of MLP with the accuracy of 62.0553%, while Figure 7 presents the percentage difference (PD) between ABMJ and all the rest of the employed techniques. PD is calculated as:

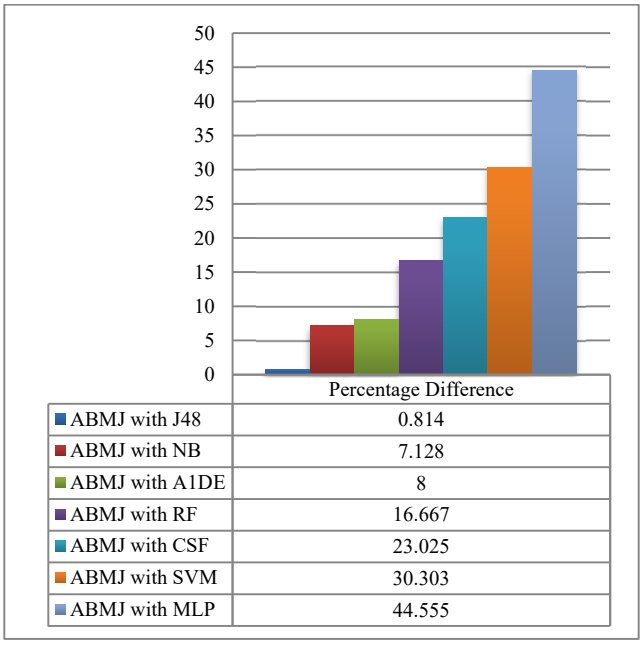
$$PD = \left(\frac{n1-n2}{\frac{n1+n2}{2}}\right) * 100 \qquad (6)$$

Where n1 represents the value of ABMJ while n2 represents the value of other techniques. The illustration shows a minimal difference between ABMJ and J48 is 0.814%, however, there is very less difference between the comparison of NB and A1DE with ABMJ that is 7.128% of NB and 8% of A1DE with ABMJ. As discussed above, the outcomes of MLP are worst in our case, the difference between ABMJ and MLP is 44.555%.



	Accuracy
ABMJ	97.6285
J48	96.8379
NB	90.9091
A1DE	90.1186
RF	82.6087
CSF	77.4704
SVM	71.9368
MLP	62.0553

Fig. 6 Accuracy Analyses through Individual Technique



	Percentage Difference
ABMJ with J48	0.814
ABMJ with NB	7.128
ABMJ with A1DE	8
ABMJ with RF	16.667
ABMJ with CSF	23.025
ABMJ with SVM	30.303
ABMJ with MLP	44.555

Fig. 7 Percentage Difference between ABMJ and other Employed Techniques

The outcome achieved through Pr, Re, FM, and MCC is present in Table 3. This table also illustrates the best performance of ABMJ compared with other employed techniques on the mentioned measures. In the last row of this table, under the value of Pr, FM, and MCC there is a "?" symbol instead of value achieved through MLP. This is due to the "0" value present in the CM. As we know that "0" cannot be divided by any value, so instead of that we

put "?" in the field. To strengthen over-analysis, we also have calculated the Receiver Operating Characteristic Area (ROCA) and Precision-Recall Area (PRCA) presented in Figure 8. The overall analyses through each measure present the better performance of proposed ABMJ for the prediction of risk in software requirements.

Table 3. Pr, Re, FM and MCC Analyses via each Employed Technique

Technique	*Precision*	*Recall*	*FM*	*MCC*
ABMJ	0.977	0.976	0.976	0.968
J48	0.969	0.968	0.968	0.958
NB	0.915	0.909	0.91	0.876
A1DE	0.904	0.901	0.902	0.866
RF	0.848	0.826	0.788	0.757
CSF	0.815	0.775	0.736	0.668
SVM	0.715	0.719	0.71	0.606
MLP	?	0.621	?	?

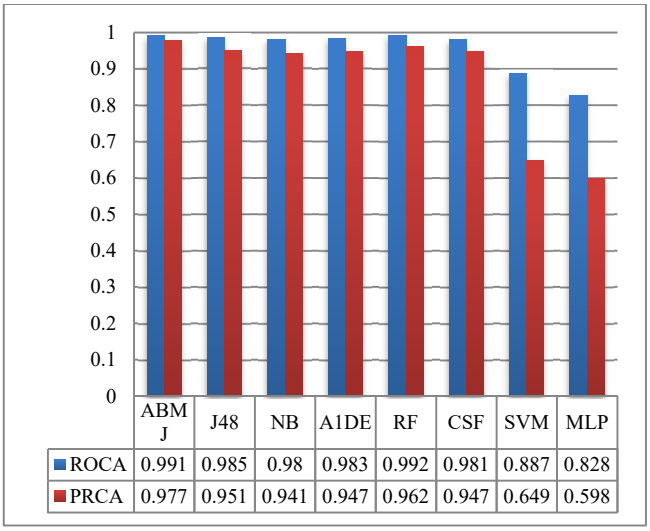


Fig. 8 ROCA and PRCA Analysis through each Employed Technique

5.1 Discussion

A software project is more likely to fail if it does not satisfy the client's budget, needs, or timeline, and the product's quality suffers as a result. As a result, a product should be built within budget and schedule constraints to reduce work and the likelihood of failure. The late risk forecasting has a greater impact on project failure.

5.1.1 Why the performance of proposed model is better then other employed models?

Ensemble techniques in statistics and ML combine many learning algorithms to achieve greater prediction performance than each of the constituent learning algorithms alone [34]. Adaptive Boosting is a Boosting approach used in ML as an Ensemble Method. The weights are re-allocated to each instance, with larger weights applied to improperly classified instances [35]. AdaBoost trains several classifiers at the same time. Each classifier is trained on examples that were more complex for the previous classifier. To that purpose, each instance is allocated a weight, and if an instance proves difficult to categorise, its weight rises [36]. Because the input parameters are not jointly optimised, Adaboost is less prone to overfitting. Adaboost can increase the accuracy of weak classifiers. Moreover, J48 demands less work for data preparation during pre-processing than other algorithms. It does not need data standardisation or scaling. Missing values in the data have no significant impact on the process of developing a decision tree [37]. The J48 (C4.8) decision tree technique is a sophisticated decision tree approach that works well on huge datasets [9], [16]. In our situation, the dataset is similarly enormous, with a large amount of data. These are the basic reasons that the proposed model performs as compare to other employed models.

5.1.2 Model Preparation

A risk prediction model is proposed, validated, and evaluated to compare and test the results of appropriate classifiers among ABMJ, J48, NB, A1DE, RF, CSF, SVM, and MLP classifiers, and the results show that ABMJ is the best appropriate classifier in the environment related to software risk prediction in the software requirements.

5.1.3 Model Evaluation

We compared J48, NB, A1DE, RF, CSF, SVM, and MLP classifiers with ABMJ and found that the proposed ABMJ classifier is the most optimal for the prediction of risk in software requirements.

5.2 Threats to Validity

The above sections present the better performance of the proposed model (ABMJ) but there are some threats to the validity of these analyses. These threats to validities are divided into three categories these are:

Internal Validity: This study's investigation is based on a variety of well-known evaluation measures that have previously been used in other studies. A risk is that the renewal of some other or the most recent standards as a replacement for employed norms may reduce the acquired results. Furthermore, the procedures used in this investigation can be replaced with some new techniques that can be hybridised with one another and provide better results than the previously used methods.

External Validity: Our hypothesis was evaluated using a dataset from the Zenodo archive, https://zenodo.org/record/1209601#.Xpa9mUAzZdg. A threat to validity may emerge if we link the extended methodologies with other data obtained from other software development firms using studies, etc., or if we replace this dataset with another dataset, which may alter

the results when calculating the error rates. Similarly, extended approaches may be unable of providing improved projections in outcomes when using several datasets.

Construct Validity: Various ML approaches are benchmarked with proposed ABMJ based on a few performance assessment metrics. The techniques combined in this study are in the centre point of their reformist highlights over numerous procedures used by scholars in previous years. However, there is a risk that if we add any other new approaches, these new techniques may exhaust the expanded techniques. It is also encouraging that using the most up-to-date performance evaluation metrics produces better results that can outperform the existing findings.

Conclusion

Risk prediction in software requirements is an active research subject with growing research community engagement. The goal of this research is to present a methodology for risk prediction in software requirements using a requirements risk dataset. The proposed model is benchmarked with seven different ML techniques to find a better solution for risk prediction in software requirements. The analysis evaluated through Pr, Re, FM, MCC, and accuracy shows the better performance of ABMJ with the accuracy of 97.6285 % and the worst performance of MLP with the accuracy of 62.0553%. This study's comprehensive results can be utilised as a guideline for other researches. Any confirmation that a new approach, model, or framework improves prediction may be benchmarked and confirmed. Class imbalance issues should be committed to the databases for future development. To enhance accuracy, feature selection and ensemble learning techniques should be researched. Furthermore, this research may be utilised to identify the optimal classifier for developing and deploying a model for risk prediction in software requirements.

Acknowledgments

The authors would like to thank the Deanship of Scientific Research at Majmaah University for supporting this work under project number No. R-2022-65

References

[1] M. Yaseen, A. Mustapha, and N. Ibrahim, “An approach for managing large-sized software requirements during prioritization,” 2018 IEEE Conf. Open Syst. ICOS 2018, pp. 98–103, 2019, doi: 10.1109/ICOS.2018.8632806.

[2] B. B. Duarte, A. L. De Castro Leal, R. D. A. Falbo, G. Guizzardi, R. S. S. Guizzardi, and V. E. Silva Souza, “Ontological foundations for software requirements with a focus on requirements at runtime,” Appl. Ontol., vol. 13, no. 2, pp. 73–105, 2018, doi: 10.3233/AO-180197.

[3] F. Hujainah, R. B. A. Bakar, M. A. Abdulgabber, and K. Z. Zamli, “Software Requirements Prioritisation: A Systematic Literature Review on Significance, Stakeholders, Techniques and Challenges,” IEEE Access, vol. 6, pp. 71497–71523, 2018, doi: 10.1109/ACCESS.2018.2881755.

[4] I. M. Keshta, M. Niazi, and M. Alshayeb, “Towards the implementation of requirements management specific practices (SP 1.1 and SP 1.2) for small- And medium-sized software development organisations,” IET Softw., vol. 14, no. 3, pp. 308–317, 2020, doi: 10.1049/iet-sen.2019.0180.

[5] Z. S. Shaukat, R. Naseem, and M. Zubair, “A dataset for software requirements risk prediction,” Proc. - 21st IEEE Int. Conf. Comput. Sci. Eng. CSE 2018, pp. 112–118, 2018, doi: 10.1109/CSE.2018.00022.

[6] G. Viswanathan and P. Jayagopal, “MODELS FOR FOG AND EDGE COMPUTING INFRASTRUCTURES A Threat Categorization of Risk - Based approach for analyzing Security Threats early phase in SDLC,” Arab. J. Sci. Eng., no. 0123456789, 2021, doi: 10.1007/s13369-021-05602-x.

[7] B. Khan, R. Naseem, M. Binsawad, M. Khan, and A. Ahmad, “Software cost estimation using flower pollination algorithm,” J. Internet Technol., vol. 21, no. 5, pp. 1243–1251, 2020, doi: 10.3966/160792642020092105002.

[8] J. Dhlamini, I. Nhamu, and A. Kaihepa, "Intelligent risk management tools for software development," in Proceedings of the 2009 Annual Conference of the Southern African Computer Lecturers' Association, pp. 33-40, 2009, doi: 10.1145/1562741.1562745

[9] R. Naseem et al., “Investigating Tree Family Machine Learning Techniques for a Predictive System to Unveil Software Defects,” Complexity, vol. 2020, pp. 1–21, 2020, doi: 10.1155/2020/6688075.

[10] V. Bijalwan, V. Kumar, P. Kumari, and J. Pascual, “KNN based machine learning approach for text and document mining,” Int. J. Database Theory Appl., vol. 7, no. 1, pp. 61–70, 2014, doi: 10.14257/ijdta.2014.7.1.06.

[11] B. W. Boehm, “Software risk management: Principles and practices,” Softw. Manag. Seventh Ed., no. January, pp. 365–374, 2007, doi: 10.1109/9780470049167.ch11.

[12] A. Gupte, S. Joshi, P. Gadgul, and A. Kadam, “Comparative Study of Classification Algorithms used in Sentiment Analysis,” Int. J. Comput. Sci. Inf. Technol., vol. 5, no. 5, pp. 6261–6264, 2014.

[13] J. Oxenstierna, “Predicting house prices using Ensemble Learning with Cluster Aggregations,” Examensarbete 15 hp, vol. IT 17 091, no. December, pp. 1–42, 2017, [Online]. Available: https://uu.diva-portal.org/smash/get/diva2:1188674/FULLTEXT01.pdf.

[14] H. Hijazi, S. Alqrainy, H. Muaidi, and T. Khdour, “Identifying causality relation between software projects risk factors,” Int. J. Softw. Eng. its Appl., vol. 8, no. 2, pp. 51–58, 2014, doi: 10.14257/ijseia.2014.8.2.06.

[15] B. Khan, R. Naseem, F. Muhammad, G. Abbas, and S. Kim, “An empirical evaluation of machine learning techniques for chronic kidney disease prophecy,” IEEE Access, vol. 8, pp. 55012–55022, 2020, doi: 10.1109/ACCESS.2020.2981689.

[16] B. Khan et al., “Software Defect Prediction for Healthcare Big Data: An Empirical Evaluation of Machine Learning Techniques,” J. Healthc. Eng., vol. 2021, 2021, doi: 10.1155/2021/8899263.

[17] D. L. Miholca, G. Czibula, and I. G. Czibula, “A novel approach for software defect prediction through hybridizing gradual relational association rules with artificial neural networks,” Inf. Sci. (Ny)., vol. 441, pp. 152–170, 2018, doi: 10.1016/j.ins.2018.02.027.

[18] S. Picek, A. Heuser, and S. Guilley, “Template attack versus Bayes classifierPicek, S., Heuser, A., & Guilley, S. (2017). Template attack versus Bayes classifier. Journal of Cryptographic Engineering, 7(4), 343–351. https://doi.org/10.1007/s13389-017-0172-7,” J. Cryptogr. Eng., vol. 7, no. 4, pp. 343–351, 2017, doi: 10.1007/s13389-017-0172-7.

[19] K. A. Otunaiya and G. Muhammad, “Performance of Datamining Techniques in the Prediction of Chronic Kidney Disease,” Comput. Sci. Inf. Technol., vol. 7, no. 2, pp. 48–53, 2019, doi: 10.13189/csit.2019.070203.

[20] S. Chatterjee, N. Dey, F. Shi, A. S. Ashour, S. J. Fong, and S. Sen, “Clinical application of modified bag-of-features coupled with hybrid neural-based classifier in dengue fever classification using gene expression data,” Med. Biol. Eng. Comput., vol. 56, no. 4, pp. 709–720, 2018, doi: 10.1007/s11517-017-1722-y.

[21] E. A. AL-Dreabi, M. M. Otoom, B. Salah, Z. M. Hawamdeh, and M. Alshraideh, "Automated Detection of Breast Cancer Using Artificial Neural Networks and Fuzzy Logic," International Journal of Sciences: Basic and Applied Research (IJSBAR), vol. 35, 2017.

[22] A. B. Nassif, D. Ho, and L. F. Capretz, “Towards an early software estimation using log-linear regression and a multilayer perceptron model,” J. Syst. Softw., vol. 86, no. 1, pp. 144–160, 2013, doi: 10.1016/j.jss.2012.07.050.

[23] M. J. Siers and M. Z. Islam, “Cost sensitive decision forest and voting for software defect prediction,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 8862, pp. 929–936, 2014, doi: 10.1007/978-3-319-13560-1.

[33] M. M. Otoom, M. Jemmali, Y. Qawqzeh, K. N. SA, and F. Al Fay, "Comparative Analysis of Different Machine Learning Models for Estimating the Population Growth Rate in Data-Limited Area," IJCSNS, vol. 19, p. 96, 2019.

[25] N. Nahar and F. Ara, “Liver Disease Prediction by Using Different Decision Tree Techniques,” Int. J. Data Min. Knowl. Manag. Process, vol. 8, no. 2, pp. 01–09, 2018, doi: 10.5121/ijdkp.2018.8201.

[26] K. Shaukat Dar and S. M. Ulya Azmeen, “Dengue Fever Prediction: A Data Mining Problem,” J. Data Mining Genomics Proteomics, vol. 06, no. 03, 2015, doi: 10.4172/2153-0602.1000181.

[27] B. T. Pham, “A Novel Classifier Based on Composite Hyper-cubes on Iterated Random Projections for Assessment of Landslide Susceptibility,” J. Geol. Soc. India, vol. 91, no. 3, pp. 355–362, 2018, doi: 10.1007/s12594-018-0862-5.

[28] A. Iqbal et al., “Performance analysis of machine learning techniques on software defect prediction using NASA datasets,” Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 5, pp. 300–308, 2019, doi: 10.14569/ijacsa.2019.0100538.

[29] H. Jin, S. Kim, and J. Kim, “Decision factors on effective liver patient data prediction,” Int. J. Bio-Science Bio-Technology, vol. 6, no. 4, pp. 167–178, 2014, doi: 10.14257/ijbsbt.2014.6.4.16.

[30] A. Alsaeedi and M. Z. Khan, “Software Defect Prediction Using Supervised Machine Learning and Ensemble Techniques : A Comparative Study,” pp. 85–100, 2019, doi: 10.4236/jsea.2019.125007.

[31] S. Khan, R. Ullah, A. Khan, N. Wahab, M. Bilal, and M. Ahmed, “Analysis of dengue infection based on Raman spectroscopy and support vector machine (SVM),” Biomed. Opt. Express, vol. 7, no. 6, p. 2249, 2016, doi: 10.1364/boe.7.002249.

[32] C. Davi et al., “Severe Dengue Prognosis Using Human Genome Data and Machine Learning,” IEEE Trans. Biomed. Eng., vol. 66, no. 10, pp. 2861–2868, 2019, doi: 10.1109/tbme.2019.2897285.

[33] M. M. Otoom, "Comparing the Performance of 17 Machine Learning Models in Predicting Human Population Growth of Countries," IJCSNS, vol. 21, p. 220, 202

[34] P. Vaitkevicius and V. Marcinkevicius, “Comparison of Classification Algorithms for Detection of Phishing Websites,” Inform., vol. 31, no. 1, pp. 143–160, 2020, doi: 10.15388/20-INFOR404.

[35] U. R. Acharya et al., “An integrated index for identification of fatty liver disease using radon transform and discrete cosine transform features in ultrasound images,” Inf. Fusion, vol. 31, pp. 43–53, 2016, doi: 10.1016/j.inffus.2015.12.007.

[36] T. C. F. Yip et al., “Laboratory parameter-based machine learning model for excluding non-alcoholic fatty liver disease (NAFLD) in the general population,” Aliment. Pharmacol. Ther., vol. 46, no. 4, pp. 447–456, 2017, doi: 10.1111/apt.14172.

[37] R. Naseem et al., “Performance Assessment of Classification Algorithms on Early Detection of Liver Syndrome,” J. Healthc. Eng., vol. 2020, 2020, doi: 10.1155/2020/6680002.