IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 16 August 2022, accepted 9 September 2022, date of publication 12 September 2022,
date of current version 22 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3206382

RESEARCH ARTICLE

Analysis of Tree-Family Machine Learning Techniques for Risk Prediction in Software Requirements

BILAL KHAN[1], RASHID NASEEM[2], IFTIKHAR ALAM[1], INAYAT KHAN[3], HISHAM ALASMARY[4,5], AND TAJ RAHMAN[6]

[1]Department of Computer Science, City University of Science and Information Technology, Peshawar 25000, Pakistan
[2]Department of IT and Computer Science, Pak-Austria Fachhochschule: Institute of Applied Sciences and Technology, Haripur 22620, Pakistan
[3]Department of Computer Science, The University of Buner, Buner 19281, Pakistan
[4]Department of Computer Science, College of Computer Science, King Khalid University, Abha 62529, Saudi Arabia
[5]Information Security and Cybersecurity Unit, King Khalid University, Abha 62529, Saudi Arabia
[6]Department of Computer Science, Qurtuba University of Science and Technology, Peshawar 24830, Pakistan

Corresponding author: Inayat Khan (inayat_khan@uop.edu.pk)

This work was supported by the Deanship of Scientific Research at King Khalid University through the Large Groups Project under Grant RGP.2/201/43.

ABSTRACT Risk prediction is the most sensitive and critical activity in the Software Development Life Cycle (SDLC). It might determine whether the project succeeds or fails. To increase the success probability of a software project, the risk should be predicted at the early stages. This study proposed a novel model based on the requirement risk dataset to predict software requirement risks using Tree-Family -Machine-Learning (TF-ML) approaches. Moreover, the proposed model is compared with the state-of-the-art models to determine the best-suited methodology based on the nature of the dataset. These strategies are assessed and evaluated using a variety of metrics. The findings of this study may be reused as a baseline for future studies and research, allowing the results of any proposed approach, model, or framework to be benchmarked and easily checked.

INDEX TERMS Risk in requirements, risk dataset for requirements, tree family machine learning technique.

I. 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 necessary, change their requirements to keep on satisfying their purpose [2]. This specific software usually comprises a base system liable for the fundamental functionalities, alongside a part that screens the base system, examines the data, and responds suitably to ensure that the system keeps on executing its necessary functions. RE is regarded as the most fundamental stage in software development since it primarily involves eliciting, documenting, and maintaining stakeholders' requirements [3]. Meeting and ensuring that stakeholders' essential needs are met regularly is one of the primary reasons for producing a high-quality software system [4], [5].

There is consistently a casual of inexact procedures during the time spent in the Software Development Life Cycle (SDLC), which may likely defeat software organization or software development. These questionable procedures are known as software risks. The risks burst from various risk influences 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 [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 catastrophes in the project [4], [6].

Keeping the earlier issue of risk prediction at the early stage of software needs in mind, numerous researchers assessed

The associate editor coordinating the review of this manuscript and approving it for publication was Jolanta Mizera-Pietraszko.

and created several models applying various categorization algorithms. However, any broad-spectrum preparation to kick-start the use of these techniques is tough to come up with. Overall, despite significant variances in the experiments, it was revealed that no one methodology confers higher precision to different approaches based on additional data. Most studies have employed various assessment measures to increase accuracy. Still, to our knowledge, no one has concentrated on decreasing the error rate, which is also a critical feature of any prediction model [12], [13]. This study has the following two primary objectives.

i To present a risk prediction model (in TF-ML models) that will aid in cost and schedule reductions and improve project quality by lowering the likelihood of project failure.

ii To compare the results of classification models to find the best efficient methodology for risk prediction in the SDLC Requirement phase.

This study's primary contributions are as follows:

i We analyzed ten alternative TF-ML approaches for risk prediction in software requirements (CDT, CS-Forest, DS, Forest-PA, HT, J48, LMT, RF, RT, and REP-T).

ii We reveal the insight of the experiments using RAE, MAE, RRSE, RMSE, recall, precision, F-measure, MCC, and accuracy metrics.

iii We do several tests on the software requirements risk dataset from Zenodo repository, available at https://zenodo.org/record/1209601#.Xpa9mUAzZdg.

The rest of the paper is divided into six sections. Section 2 describes the experimental methodology, Sections 3 and 4 discuss model assessment and comparison and the details of all applied techniques, respectively, and Section 5 gives practical results and discussions. The concluding section is covered in Section 6.

II. LITERATURE REVIEW

Requirement Engineering (RE) is an organized and systematic approach to gathering users' requirements for a software system [7]. It usually comprises a system accountable for the basic functionalities, examines the data, and responds suitably. The RE is considered the essential stage in software development since it mainly consists of eliciting, documenting, and maintaining stakeholders' requirements [8]. There is consistently a casual of inexact procedures during the time spent in the SDLC, which may likely downfall a software organization or software development process. These questionable procedures are known as software risks [9]. If threats/risks are not handled appropriately, they may get liable for the disaster of the project [6]. The dangers have a significant influence on software requirements. They turn out to be the cause of software or stakeholder harm. As a result, risks must be predicted early in the SDLC to increase project success possibilities because risk evaluation at this point will be more helpful and will increase software production [10], [47]. When risks are appropriately handled, it also helps to reduce the likelihood of software project failure.

Frequent solutions for predicting software risk at different phases in SDLC are available. In contrast, infrequent methods are available to predict risks in the software requirements phase in the literature [6], [11]. A risk prediction model encompasses data mining classification methods and is proposed to predict risks on the project's Software Requirement Specifications (SRS). The TF approach is one of the strangest techniques for organizing the most significant variables and their interactions between two or more variables. TFs can develop new features with more significant predicting potential for object variables. It needs less data purification than other modelling methodologies. It is not biased to a considerable degree by outliers and missing data [17], [18], [19].

The reasoning for utilizing TF-ML techniques has been considered one of the optimum and most often used supervised learning methods [12], [13]. Tree-based techniques increase predictive models' accuracy, stability, and interpretability [14]. TF-ML techniques effectively map nonlinear interactions utilizing heterogeneous linear models. When dealing with all sorts of obstacles, they are adaptive (regression or classification). Both continuous and categorical input and output variables can be used with these approaches [15], [16].

We analyzed Tree Family Machine Learning (TF-ML) methods for software requirement risk prediction. Some of the TF-ML techniques include the Decision Tree (J48), Forest by Penalizing Attributes (Forest-PA), REP-Tree (REP-T), Decision Stump (DS), Credal Decision Tree (CDT), Random Forest (RF), Random Tree (RT), Hoeffding Tree (HT), Cost-Sensitive Decision Forest (CS-Forest), and Logistic Model Tree (LMT). On the Zenodo repository dataset, several methods are employed. The studies are validated using root relative squared error (RRSE), root mean squared error (RMSE), relative absolute error (RAE), mean fundamental error (MAE), accuracy, Matthew's Correlation Coefficient (MCC), recall, F-measure (FM), and precision.

If a project fails to fulfil the user's needs, budget, or timeline, the product's quality suffers. As a result, it is more likely to fail [14]. So, to limit effort and the likelihood of failure, a product must be built within the budget and schedule constraints. The late discovery of risk has a more significant effect on project failure. It is also necessary to forecast risk early in the SDLC process (Software Requirements).

The data obtained from previous projects can be used for the growth by either using machine learning (ML) approaches, such as Artificial Network Network (ANN), and Support Vector Machines (SVM)or a mathematical methodology, including study of association and linear regression [45], [46]. Moreover, Software shortcoming prediction aim to forecast defect-prone mechanisms before the testing stage of SDLC [48].

III. EXPERIMENTAL METHODOLOGY

This research aims to analyse TF-ML approaches for risk prediction in software requirements using the Zenodo repository dataset. The dataset used contains the 13 characteristics

TABLE 1. List of attributes with distinct types.

S. No.	Name	Type	Distinct
1	Requirements	Nominal	292
2	Project Category	Nominal	4
3	Requirement Category	Nominal	10
4	Risk Target Category	Nominal	22
5	Probability	Numeric	81
6	Magnitude of Risk	Nominal	7
7	Impact	Nominal	5
8	Dimension of Risk	Nominal	13
9	Affecting No of Modules	Numeric	9
10	Fixing Duration (Days)	Numeric	12
11	Fix Cost (% of Project)	Numeric	10
12	Priority	Numeric	293
13	Risk Level	Nominal	5

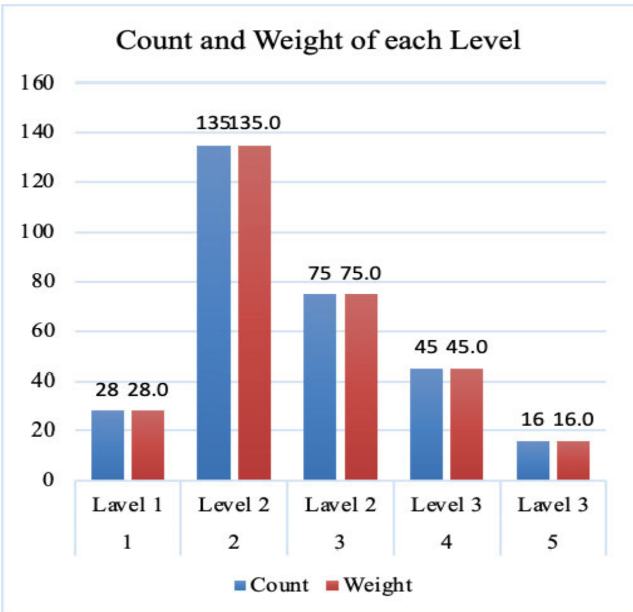

FIGURE 1. Count and weight of each class (level).

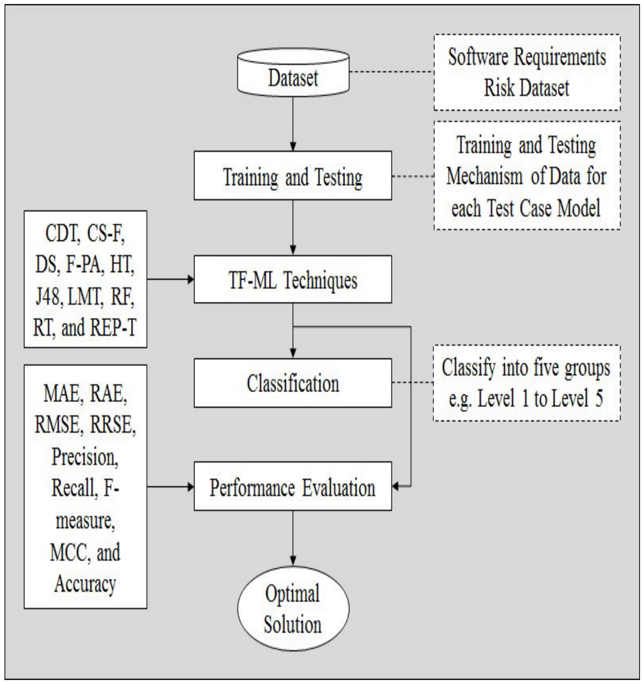

FIGURE 2. Methodology workflow.

TABLE 2. Training and testing mechanism for each test case model.

S. No.	Test Case Model	Training	Testing
1	Case 1	90 %	10 %
2	Case 2	80 %	20 %
3	Case 3	70 %	30 %
4	Case 4	60 %	40 %
5	Case 5	50 %	50 %
6	Case 6	40 %	60 %
7	Case 7	30 %	70 %
8	Case 8	20 %	80 %
9	Case 9	10 %	90 %
10	Case 10	10-Fold Cross-validation	

stated in Table 1 and 299 occurrences. The data is divided into five categories: level 1, level 2, level 3, level 4, and level 5 [6]. Figure 1 depicts the count and weight of each level. Figure 2 depicts the whole workflow of this investigation. Data is separated into 90% and 10% for training and testing, respectively, and this procedure is applied in different test scenarios, where data testing is raised while training is dropped by 10%. Training and testing are 80% and 20% in the second scenario, respectively. These examples determine the most effective testing and training splitting criteria. Table 2 lists the testing and training instances. We have performed 10 different types of experiminations based on data splitting for training and testing purposes to show the better data splitting mechanism in this regards. In case 1, the data is divided into 90% for training and 10% for testing, in case to we decrease the training and increase the testing by 10%, so 80% is used for training and the rest of 20% I sused for testing and so on upto 10% for training and 90% for testing. In the last scenario, 10-fold cross-validation is used. Many studies advocate 10-fold cross-validation as a benchmark [15], [16].

The employed techniques are evaluated using standard evaluation measures presented in the subsequent.

IV. MODEL EVALUATION AND COMPARISON

Various assessment metrics evaluate ten TF-ML approaches, including J48, HT, CDT, RF, RT, LMT, CS-Forest, and REP-T. A 10-fold cross-validation procedure is employed for training and testing where the dataset is partitioned into ten subdivisions of equivalent dimensions. One subdivision is utilized for testing in the first fold, while the remaining nine are used for training. Furthermore, the second subdivision in the second fold is utilized for testing, while the remaining nine are used for training. This method will be repeated until each subdivision has been tested [15]. Assessment is done based on MAE [17], [18], RAE [17], [19], RMSE [20], [21], RRSE [17], [19], precision [22], [23], recall [22], [24], F-measure [25], [26], MCC [25], [27], and accuracy [26], [28], [29], where, P_{ij} is the rate of prediction by the precise model, T_j is the goal value for record ji, I stands for record j (out of n records), n is the number of errors, $|y_i - y|$ is the absolute error. However, TP is used for the total of true-positive classification. At the same time, FN denotes the count of false-negative classification, FP is the count of false-positive classifications, and TN is the count of true-negative classification. These assessments can be calculated using the following equations:

$$MAE = \frac{1}{2}\sum\nolimits_{j=1}^{n} |y_i - y| \tag{1}$$

$$RAE = \frac{\sum_{j=1}^{n} \left|p_{ij} - T\right|}{\sum_{j=1}^{n} \left|T_j - T\right|} \tag{2}$$

$$RMSE = \sqrt{\frac{1}{2}\sum\nolimits_{j=1}^{n} (y_i - 1)^2} \tag{3}$$

$$RRSE = \sqrt{\frac{\sum_{j=1}^{n} \left(P_{ij} - T_j\right)^2}{\sum_{j=1}^{n} \left(T_j - T\right)^2}} \tag{4}$$

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

$$FM = \frac{2 * Precision + Recall}{Precision + Recall} \tag{7}$$

$$MCC = \frac{(TN * TP) - (FN * FP)}{\sqrt{(FP + TP)(FN + TP)(TN + FP)(TN + FN)}} \tag{8}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

V. EMPLOYED TF-ML TECHNIQUES

ML approaches are presently widely utilized to extract important information from enormous data in various fields. Recognizing communities in social networks, Cybersecurity, bioinformatics, and improving the design process to generate high-quality software systems are just a few of the real-world uses of ML [30]. ML and TF-ML-based SDP solutions have also been explored [31], [22], [32]. Table 3 presents the list of all TF-ML techniques used in this study.

VI. EXPERIMENTAL RESULTS AND DISCUSSION

This section illustrates the study's findings and discussion. Ten TF-ML approaches are used and assessed using a variety of criteria. Experimentation utilises various test case components (See Table 2). Each module examines the strategies

TABLE 3. List of TF-ML techniques used in this study.

S. No.	TF-ML Technique	References
1	Credal Decision Tree	[33] [33] [34]
2	Cost-Sensitive Decision Forest	[35] [34] [34]
3	Decision Stump	[36] [32] [36]
4	Forest by Penalizing Attributes	[26] [37] [35]
5	Hoeffding Tree	[34] [34] [36]
6	Decision Tree (J48)	[38] [39] [40]
7	Logistic Model Tree	[34] [36] [41]
8	Random Forest	[26] [42] [43]
9	Random Tree	[34] [36] [44]
10	REP-Tree	[34] [34] [44]

to discover the best solution for risk prediction in software requirements. Each TF-ML approach calculates Correctly Classified Instances (CCI) and Incorrectly Classified Instances (ICI). Table 4 depicts the complete analysis of CCI and ICI. Each column represents a test case module, indicating how data is separated into testing and training. Ten alternative scenarios have been devised to improve data analysis for this goal. The second column represents CCI and ICI concerning each test case module.

In contrast, the proportion of CCI and ICI attained by each approach for each test case module is represented by the remainder of the columns. The best test case that we consider is 10-Fold cross-validation, the most utilized standard. In Table 3, CDT and F-PA (in two cases) outperform other techniques depending on different test case modules. While using the best test case module (10-fold cross-validation), CDT surpasses the other approaches used. Tables 5, 6, 7, and 8 show the results of the MAE, RMSE, RAE %, and RRSE % analyses, respectively. The first column in each table represents the test case modules, while the remaining columns indicate the results of each approach. Each table displays the most outstanding performance of CDT and J48 (in one example) to minimise error rate by employing different test case modules. The best outcomes of each technique are presented in bold text in the respective table. If there is a need to decrease the error rate in forecasting risks in software requirements, this study suggests CDT and J48 techniques. However, we have mostly seen researchers split the data into 20 % to 40% for testing and 60% to 80% for training or suggest 10-Fold cross-validation. In these cases, this study recommends the CDT technique to reduce error rates compared to the other utilized techniques.

Tables 9, 10, 11, and 12 show the outcome analysis of average precision, recall, F-measure, and MCC. CDT, F-PA, and J48 exceed other approaches in each table to achieve better results. A "?" sign appears in Tables 9, 11, and 12. Due to the "0" value in the confusion matrix, this is a Weka auto-generated symbol. If there is a need to divide a value and that value becomes "0," we know that "0" is not divisible,

TABLE 4. Number of CCI and ICI obtaining from different percentage split test cases.

Test Modes		CDT	CS-F	DS	F-PA	HT	J48	LMT	RF	RT	REP-T
10/90 (269)	CCI	190 (70.6%)	156 (57.9%)	184 (68.4%)	174 (64.7%)	194 (72.1%)	231 (85.9%)	182 (67.6%)	149 (55.4%)	124 (46.1%)	119 (44.2%)
	ICI	79 (29.4%)	113 (42.1%)	85 (31.6%)	95 (35.3%)	75 (27.9%)	38 (14.1%)	87 (29.4%)	120 (44.6%)	145 (53.9)	150 (55.8%)
20/80 (239)	CCI	213 (89.1%)	153 (64%)	164 (68.6%)	217 (90.8%)	204 (85.4%)	213 (89.1%)	191 (79.9%)	153 (64.1%)	166 (69.5%)	105 (43.9%)
	ICI	26 (10.9%)	86 (36%)	75 (31.4%)	22 (9.2%)	35 (14.6)	26 (10.9)	48 (20.1%)	86 (35.9%)	73 (30.5%)	134 (56.1%)
30/70 (209)	CCI	195 (93.3%)	145 (69.4%)	141 (67.5%)	181 (86.6%)	182 (87.1%)	206 (98.6%)	175 (83.7%)	139 (66.5%)	91 (43.5%)	93 (44.5%)
	ICI	14 (6.7%)	64 (30.6%)	68 (32.5%)	28 (13.4%)	27 (12.9%)	3 (1.4%)	34 (16.3%)	70 (33.5%)	118 (56.5%)	116 (55.5%)
40/60 (179)	CCI	174 (97.2%)	122 (68.2%)	121 (67.6%)	170 (95%)	161 (90%)	175 (97.8%)	155 (86.6%)	135 (75.4%)	82 (45.8%)	82 (45.8%)
	ICI	5 (2.8%)	57 (31.8%)	58 (32.4%)	9 (5%)	18 (10%)	4 (2.2%)	24 (13.4%)	44 (24.6%)	97 (54.2%)	97 (54.2%)
50/50 (149)	CCI	143 (96%)	106 (71.1%)	99 (66.4%)	144 (96.6%)	132 (88.6%)	145 (97.3%)	134 (89.9%)	104 (69.8%)	66 (44.3%)	66 (44.3%)
	ICI	6 (4%)	43 (28.9%)	50 (33.6%)	5 (3.4%)	17 (11.4%)	4 (2.7%)	15 (10.1%)	45 (30.2%)	83 (55.7%)	83 (55.7%)
60/40 (120)	CCI	117 (97.5%)	91 (75.8%)	81 (67.5%)	114 (95%)	106 (88.3%)	116 (96.7%)	110 (91.7%)	93 (77.5%)	58 (48.3%)	53 (44.2%)
	ICI	3 (2.5%)	29 (24.2%)	39 (32.5%)	6 (5%)	14 (11.7%)	4 (3.3%)	10 (8.3%)	27 (22.5%)	62 (51.7%)	67 (55.8%)
70/30 (90)	CCI	87 (96.7%)	67 (74.4%)	61 (67.8%)	89 (98.9%)	83 (92.2%)	88 (97.8%)	81 (90%)	64 (71.1%)	39 (43.3%)	39 (43.3%)
	ICI	3 (3.3%)	23 (25.6%)	29 (32.2%)	1 (1.1%)	7 (7.8%)	2 (2.2%)	9 (10%)	26 (28.9%)	51 (56.7%)	51 (56.7%)
80/20 (60)	CCI	59 (98.3%)	39 (65%)	40 (66.7%)	57 (95%)	56 (93.3%)	58 (96.7%)	57 (95%)	50 (83.3%)	24 (40%)	24 (40%)
	ICI	1 (1.7%)	21 (35%)	20 (33.3%)	3 (5%)	4 (6.7%)	2 (3.3%)	3 (5%)	10 (16.7%)	36 (60%)	36 (60%)
90/10 (30)	CCI	30 (100%)	18 (60%)	19 (63.3%)	30 (100%)	26 (86.7%)	29 (96.7%)	28 (93.3%)	23 (76.7%)	11 (36.7%)	11 (36.7%)
	ICI	0 (0%)	12 (40%)	11 (36.7%)	0 (0%)	4 (13.3%)	1 (3.3%)	2 (6.7%)	7 (23.3%)	19 (63.3%)	19 (63.3%)
10Fold (299)	CCI	293 (98%)	219 (73.2%)	209 (69.9%)	286 (95.7%)	193 (64.5%)	288 (96.3%)	278 (93%)	249 (83.3%)	215 (72%)	135 (45.2%)
	ICI	6 (2%)	80 (26.8%)	90 (30.1%)	13 (4.3%)	106 (35.5%)	11 (3.7%)	21 (7%)	50 (16.7%)	84 (28%)	164 (54.8%)

according to many formulae. Weka displays the "?" sign instead of an error message. The best techniques that increase the rate of precision, recall, F-measure, and MCC here are CDT, F-PA, and J48. However, CDT and F-PA outperform on best test case modules, e.g. 10% to 40% for testing and 60% to 90% for training, on 10-Fold cross-validation. The outcomes of REP-T, DS, and CF-F for Tables 9, 11, and 12 do not generate accurate results due to multiple 0 values as a divider in the confusion matrix. Moreover, on test modules 10 and 90 for training and testing, respectively, no technique performs well. The recommendation of analyzing employed techniques via precision, recall, F-measure, and MCC on the best test case module is the CDT technique for risk prediction is software requirements.

Table 13 shows the detailed accuracy of the particular technique evaluated on each test case module. The analysis highlighted in this table represents that CDT, F-PA, and J48 outperform well instead of other employed methods. Moreover, these three techniques, CDT, F-PA, and J48, CDT and F-PA (only in two cases), are recommended to better predict risk in software requirements on the best test case module of 10-Fold cross-validation. While for other best test case

TABLE 5. MAE analysis of individual techniques on each test case module.

Test Modes	CDT	CS-F	DS	F-PA	HT	J48	LMT	RF	RT	REP-T
10 & 90	0.1319	0.2668	0.1649	0.193	0.1157	0.0654	0.1289	0.2303	0.2386	0.2713
20 & 80	0.0527	0.2712	0.1677	0.1489	0.0596	0.0435	0.0856	0.2191	0.1606	0.2756
30 & 70	0.0347	0.2694	0.1653	0.1379	0.0572	0.0057	0.0674	0.2088	0.2434	0.2777
40 & 60	0.0176	0.2714	0.1659	0.2072	0.0499	0.0118	0.0605	0.203	0.2789	0.2799
50 & 50	0.0219	0.2709	0.1703	0.1836	0.0552	0.0142	0.0573	0.1960	0.2775	0.28
60 & 40	0.0121	0.2678	0.1695	0.148	0.0513	0.0144	0.0377	0.1934	0.2607	0.2804
70 & 30	0.0174	0.2622	0.1697	0.2077	0.0421	0.013	0.038	0.2056	0.2753	0.2812
80 & 20	0.0109	0.2628	0.1751	0.2042	0.0409	0.0172	0.0245	0.2002	0.2795	0.284
90 & 10	0.0054	0.2655	0.1834	0.1461	0.0564	0.018	0.0379	0.2028	0.2872	0.2891
10Fold	0.0126	0.2538	0.1681	0.1635	0.1439	0.0183	0.0321	0.1912	0.1428	0.2796

TABLE 6. RMSE analysis of individual techniques on each test case module.

Test Modes	CDT	CS-F	DS	F-PA	HT	J48	LMT	RF	RT	REP-T
10 & 90	0.2936	0.3447	0.2982	0.294	0.319	0.2238	0.3254	0.3317	0.4079	0.3786
20 & 80	0.1916	0.3459	0.2966	0.2317	0.2102	0.2086	0.2621	0.3107	0.3265	0.3782
30 & 70	0.1586	0.3421	0.3006	0.2282	0.2016	0.0758	0.2322	0.2963	0.4061	0.3778
40 & 60	0.1066	0.3458	0.2979	0.2804	0.1814	0.0938	0.2086	0.2858	0.3768	0.376
50 & 50	0.1256	0.3443	0.3015	0.2646	0.1918	0.1028	0.198	0.2807	0.3783	0.378
60 & 40	0.1004	0.3411	0.3003	0.2108	0.1821	0.1137	0.1572	0.2724	0.4003	0.3765
70 & 30	0.1151	0.3352	0.2981	0.2749	0.1557	0.0906	0.1877	0.2881	0.3798	0.3775
80 & 20	0.0806	0.3371	0.3011	0.2762	0.1615	0.1109	0.1115	0.2769	0.3877	0.3816
90 & 10	0.014	0.3404	0.3095	0.2029	0.2136	0.1138	0.1483	0.2802	0.403	0.3879
10Fold	0.0888	0.3262	0.29	0.2332	0.2737	0.12	0.1472	0.2655	0.2718	0.374

TABLE 7. RAE% analysis of individual techniques on each test case module.

Test Modes	CDT	CS-F	DS	F-PA	HT	J48	LMT	RF	RT	REP-T
10 & 90	47.4115	95.8801	59.2414	69.3684	41.585	23.5115	46.3262	82.7636	85.7455	97.5013
20 & 80	18.8936	97.1778	60.0882	53.379	21.3573	15.5947	30.6865	78.5254	57.5599	98.7766
30 & 70	12.3843	96.2585	59.0425	49.2527	20.4235	2.0514	24.0967	74.5864	86.9468	99.2038
40 & 60	6.2681	96.4239	58.9401	73.607	17.7102	4.1961	21.4863	72.1323	99.0926	99.43
50 & 50	7.789	96.3102	60.5401	65.2549	19.6403	5.0442	20.3576	69.6666	98.6381	99.5415
60 & 40	4.299	95.1436	60.2337	52.5887	18.2279	5.1325	13.3936	68.7302	92.6396	99.6172
70 & 30	6.1808	92.9407	60.1609	73.6286	14.9221	4.6075	13.4713	72.8683	97.5757	99.6788
80 & 20	3.8428	92.2879	61.4793	71.7035	14.3788	6.0277	8.6021	70.307	98.1447	99.7413
90 & 10	1.8714	91.6634	63.3165	50.4208	19.4666	6.217	13.092	70.0144	99.1382	99.8055
10Fold	4.498	90.5256	59.9518	58.309	51.3256	6.5157	11.4635	68.2109	50.9484	99.7374

modules, e.g. 10% to 40% for testing and 60% to 90% for training, CDT and F-PA (only in two cases) both outperform other techniques. Figure 3 also describes the accuracy percentage of each technique concerning the individual test case module.

VII. DISCUSSION

This research focuses on the performance analysis of TF-ML approaches to suggest an optimal solution for risk prediction in software requirements. In a nutshell, ended our analysis with outccomes that best cases for training and testing on the aforementioned datases are the first 4 data training and testing cases that are 90% and 10% for training and testing to 60% and 40% for training and testing, and the last case that is 10-fold cross-validation. Now, if the goal is to reduce the error rate, our study shows that CDT outperforms other applied strategies on all of the selected (best test case) modules in Figures 4 (MAE and RMSE) and 5 (RAE% and RRSE%). Similarly, in the cases of recall, precision, F-measure, MCC, and accuracy, as shown in Figures 6 and 7, CDT outperform

TABLE 8. RRSE% analysis of individual techniques on each test case module.

Test Modes	CDT	CS-F	DS	F-PA	HT	J48	LMT	RF	RT	REP-T
10 & 90	77.9801	91.5502	79.2039	78.0653	84.7217	59.4383	86.4283	88.0812	108.3193	100.5403
20 & 80	50.7739	91.6528	78.5915	61.4035	55.7019	55.2756	69.4584	82.3395	86.5208	100.2266
30 & 70	42.0357	90.6822	79.6791	60.4745	53.431	20.0828	61.5491	78.5371	107.6214	100.1186
40 & 60	28.3614	91.9879	79.2443	74.5964	48.2664	24.9598	55.5051	76.0319	100.2393	100.0279
50 & 50	33.2542	91.1323	79.8098	70.0319	50.7674	27.2177	52.4186	74.3044	100.1337	100.0569
60 & 40	26.6719	90.6352	79.7809	56.0112	48.3856	30.1959	41.755	72.3749	106.3628	100.0242
70 & 30	30.5042	88.8161	78.9966	72.8564	41.2657	24.001	49.7326	76.3534	100.6475	100.0268
80 & 20	21.1181	88.3789	78.9325	72.4146	42.3328	29.0702	29.2304	72.5892	101.6425	100.048
90 & 10	3.6009	87.7937	79.8258	52.3342	55.104	29.3517	38.2495	72.2861	103.96	100.0697
10Fold	23.741	87.2203	77.5487	62.3448	73.1888	32.0907	39.3501	70.9953	72.6893	99.998

TABLE 9. Precision analysis of individual techniques on each test case module.

Test Modes	CDT	CS-F	DS	F-PA	HT	J48	LMT	RF	RT	REP-T
10 & 90	?	?	?	?	?	?	?	?	?	?
20 & 80	?	?	?	?	0.856	0.890	0.782	?	?	?
30 & 70	0.938	?	?	?	0.876	0.986	0.838	?	0.367	?
40 & 60	0.973	?	?	0.956	0.909	0.979	0.875	?	?	?
50 & 50	0.965	?	?	0.968	0.897	0.975	0.911	?	?	?
60 & 40	0.977	?	?	0.955	0.890	0.970	0.921	0.818	0.395	?
70 & 30	0.967	?	?	0.990	0.933	0.981	0.913	?	?	?
80 & 20	0.984	?	?	?	0.947	0.973	0.953	0.860	?	?
90 & 10	1.000	?	?	1.000	0.908	0.970	0.945	?	?	?
10Fold	0.980	0.772	?	0.957	0.794	0.964	0.930	0.851	0.748	?

TABLE 10. Recall analysis of individual techniques on each test case module.

Test Modes	CDT	CS-F	DS	F-PA	HT	J48	LMT	RF	RT	REP-T
10 & 90	0.706	0.580	0.684	0.647	0.721	0.859	0.677	0.554	0.461	0.442
20 & 80	0.891	0.640	0.686	0.908	0.854	0.891	0.799	0.640	0.695	0.439
30 & 70	0.933	0.694	0.675	0.866	0.871	0.986	0.837	0.665	0.435	0.445
40 & 60	0.972	0.682	0.676	0.950	0.899	0.978	0.866	0.754	0.458	0.458
50 & 50	0.960	0.711	0.664	0.966	0.886	0.973	0.899	0.698	0.443	0.443
60 & 40	0.975	0.758	0.675	0.950	0.883	0.967	0.917	0.775	0.483	0.442
70 & 30	0.967	0.744	0.678	0.989	0.922	0.978	0.900	0.711	0.433	0.433
80 & 20	0.983	0.650	0.667	0.950	0.933	0.967	0.950	0.833	0.400	0.400
90 & 10	1.000	0.600	0.633	1.000	0.867	0.967	0.933	0.767	0.367	0.367
10Fold	0.980	0.732	0.699	0.957	0.645	0.963	0.930	0.833	0.719	0.452

the other used methodologies. According to these analyses, this study recommended CDT as the best technique for forecasting risks in the software requirements. It can be seen from Figures 4-7 that in each scenario whether it is reducing the error rate or increasing the accuracy, CDT is recommended as the best solution as compared to the rest of the employed techniques.

A. THREATS TO VALIDITY

This section discusses the impacts that might jeopardize the validity of this study endeavour.

1) INTERNAL RELIABILITY

The analysis in this study is represented by a set of well-known assessment measures employed in prior studies.

TABLE 11. F-measure analysis of individual techniques on each test case module.

Test Modes	CDT	CS-F	DS	F-PA	HT	J48	LMT	RF	RT	REP-T
10 & 90	?	?	?	?	?	?	?	?	?	?
20 & 80	?	?	?	?	0.846	0.890	0.788	?	?	?
30 & 70	0.933	?	?	?	0.871	0.986	0.833	?	0.383	?
40 & 60	0.972	?	?	0.950	0.901	0.978	0.865	?	?	?
50 & 50	0.960	?	?	0.966	0.887	0.973	0.896	?	?	?
60 & 40	0.975	?	?	0.948	0.883	0.967	0.916	0.729	0.409	?
70 & 30	0.966	?	?	0.989	0.923	0.978	0.901	?	?	?
80 & 20	0.983	?	?	?	0.935	0.967	0.949	0.798	?	?
90 & 10	1.000	?	?	1.000	0.869	0.966	0.934	?	?	?
10Fold	0.980	0.699	?	0.953	0.691	0.963	0.929	0.805	0.684	?

TABLE 12. MCC analysis of individual techniques on each test case module.

Test Modes	CDT	CS-F	DS	F-PA	HT	J48	LMT	RF	RT	REP-T
10 & 90	?	?	?	?	?	?	?	?	?	?
20 & 80	?	?	?	?	0.800	0.870	0.728	?	?	?
30 & 70	0.915	?	?	?	0.826	0.981	0.781	?	0.161	?
40 & 60	0.965	?	?	0.932	0.863	0.972	0.818	?	?	?
50 & 50	0.946	?	?	0.958	0.844	0.966	0.871	?	?	?
60 & 40	0.967	?	?	0.935	0.836	0.958	0.891	0.678	0.188	?
70 & 30	0.952	?	?	0.986	0.898	0.975	0.866	?	?	?
80 & 20	0.979	?	?	?	0.919	0.962	0.932	0.767	?	?
90 & 10	1.000	?	?	1.000	0.846	0.958	0.917	?	?	?
10Fold	0.975	0.613	?	0.946	0.601	0.952	0.905	0.766	0.596	?

TABLE 13. Accuracy details of each technique concerning individual test case module.

Test Modes	CDT	CS-F	DS	F-PA	HT	J48	LMT	RF	RT	REP-T
10 & 90	70.6%	57.9%	68.4%	64.7%	72.1%	85.9%	67.6%	55.4%	46.1%	44.2%
20 & 80	89.1%	64%	68.6%	90.8%	85.4%	89.1%	79.9%	64.1%	69.5%	43.9%
30 & 70	93.3%	69.4%	67.5%	86.6%	87.1%	98.6%	83.7%	66.5%	43.5%	44.5%
40 & 60	97.2%	68.2%	67.6%	95%	90%	97.8%	86.6%	75.4%	45.8%	45.8%
50 & 50	96%	71.1%	66.4%	96.6%	88.6%	97.3%	89.9%	69.8%	44.3%	44.3%
60 & 40	97.5%	75.8%	67.5%	95%	88.3%	96.7%	91.7%	77.5%	48.3%	44.2%
70 & 30	96.7%	74.4%	67.8%	98.9%	92.2%	97.8%	90%	71.1%	43.3%	43.3%
80 & 20	98.3%	65%	66.7%	95%	93.3%	96.7%	95%	83.3%	40%	40%
90 & 10	100%	60%	63.3%	100%	86.7%	96.7%	93.3%	76.7%	36.7%	36.7%
10Fold	98%	73.2%	69.9%	95.7%	64.5%	96.3%	93%	83.3%	72%	45.2%

Some of these criteria assess the error rate, while others quantify accuracy. Along these lines, there is a risk that the renewal of specific contemporary standards as a replacement for previous standards may decrease the results achieved. Furthermore, the approaches utilized in this study can be modified with some new methods that can be combined and produce better results than the prior methods.

2) EXTERNAL VALIDITY

We ran tests on a dataset from the Zenodo archive, which can be found at: https://zenodo.org/record/1209601#.Xpa9mUAzZdg. Suppose we attach the comprehensive approaches to other data obtained from multiple software development organizations via surveys and other methods or replace this dataset with another dataset. In that case, the

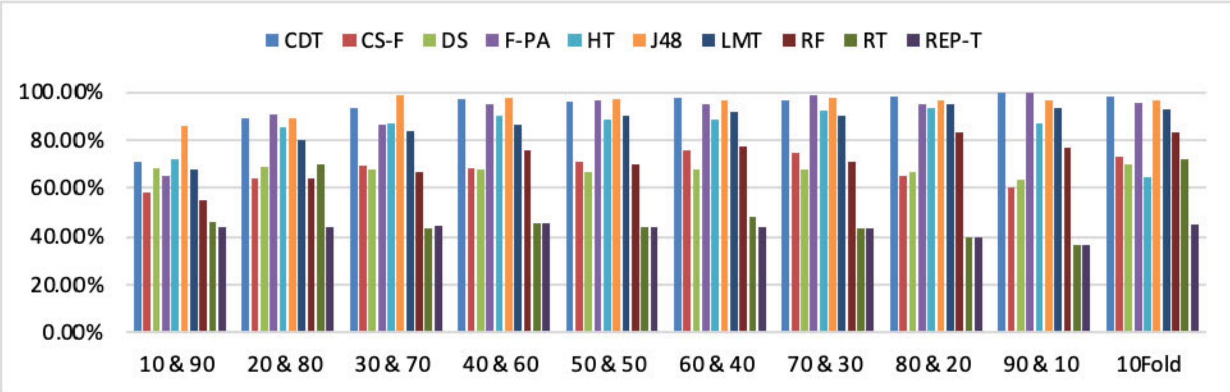

FIGURE 3. Accuracy percentage representation concerning each test case module.

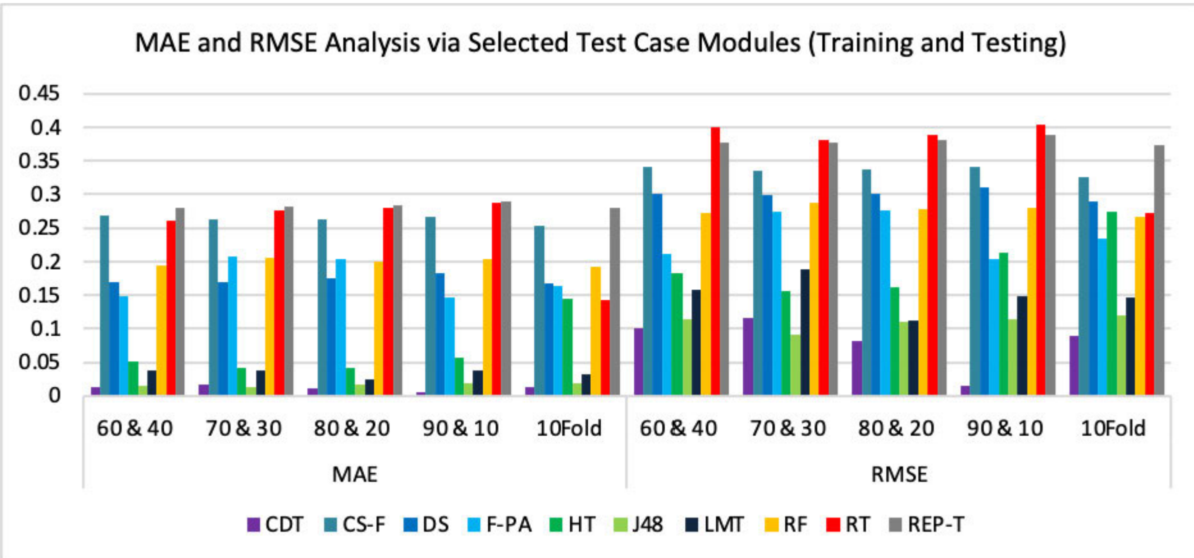

FIGURE 4. MAE and RMSE analysis via selected test case modules.

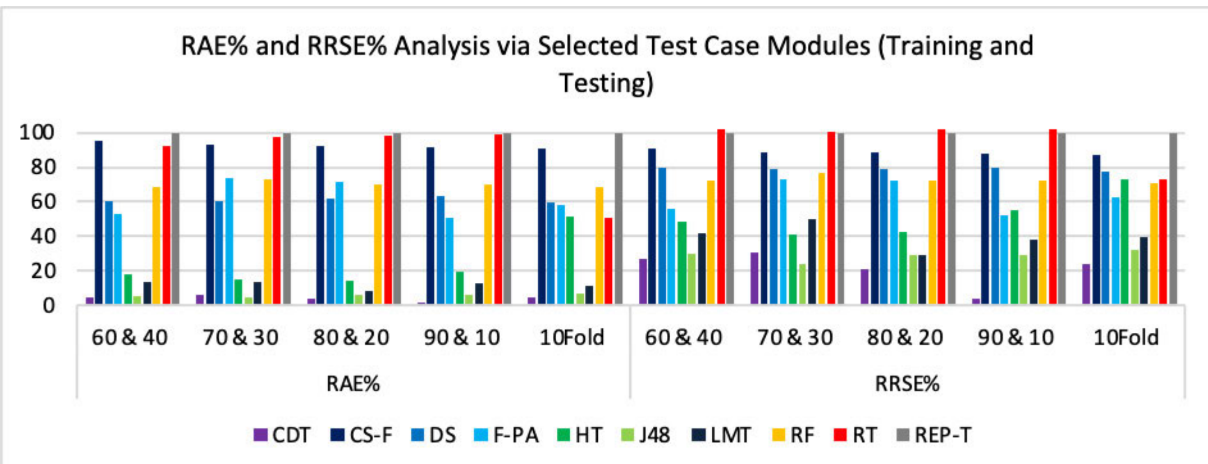

FIGURE 5. RAE% and RRSE% analysis via selected test case modules.

findings when calculating the error rates may be thrown off. Similarly, using varied datasets, the comprehensive approaches may not be able to provide improved predictions in outcomes. Following that, a thorough examination was carried out on a dataset taken from the Zenodo repository to determine the performance of the approaches used.

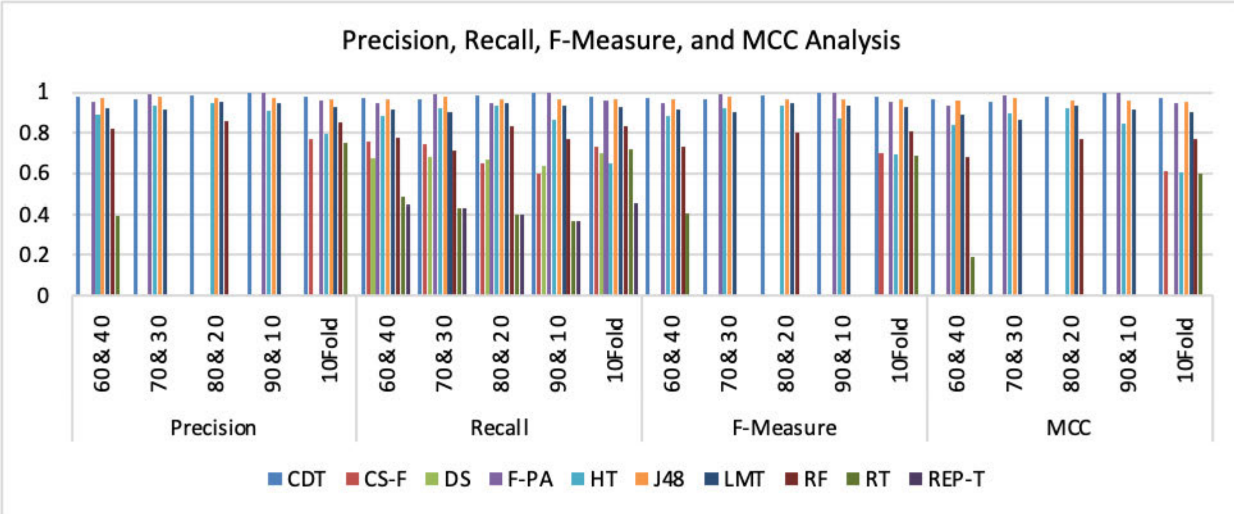

FIGURE 6. Precision, recall, F-measure, and accuracy analysis via selected test case modules.

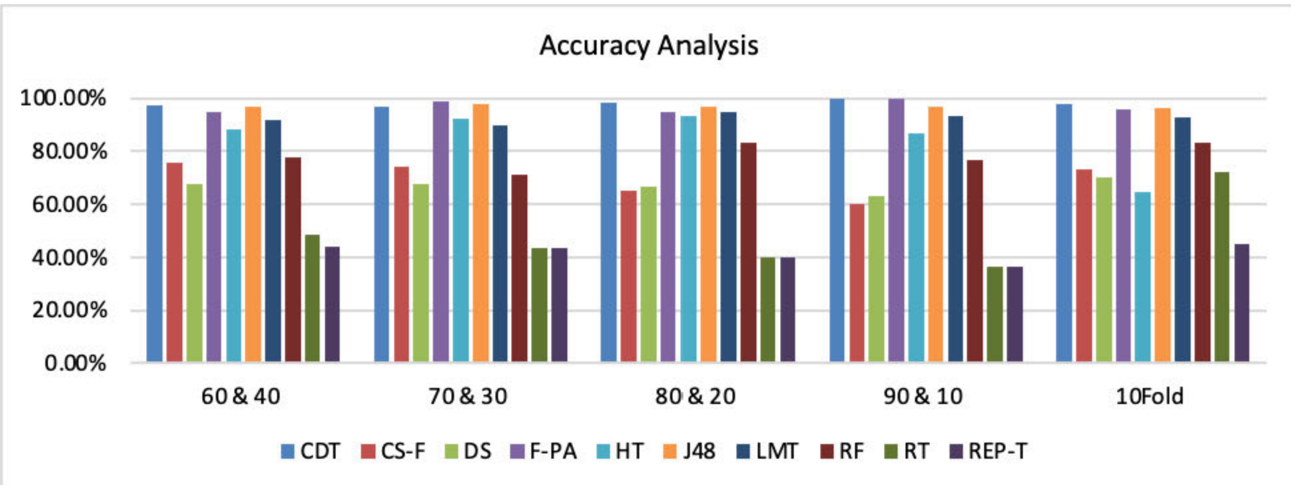

FIGURE 7. Accuracy analysis via selected test case modules.

3) CONSTRUCT VALIDITY

Several TF-ML techniques are compared against one another in this study with a few performance assessment parameters. Compared to other methodologies used by researchers in recent years, the combination of procedures used in this study is at the core of its reformist features. However, there is a threat that if we add more innovative methods, the expanded approaches will be exhausted. It's also gratifying to see that employing the most up-to-date performance evaluation measures yields better results that beat current findings.

VIII. CONCLUSION

Predicting requirement risk is an essential research topic that receives increasing interest from researchers. This research aims to create a model for predicting risk in software requirements. Ten different TF-ML techniques are used to find an optimal solution for minimum error and maximum accuracy. CDT outperforms other techniques in both error rate reduction and accuracy improvement among all the employed techniques. The results of 10-fold cross-validation for MAE, 0.0888 for RMSE, 4.498 % for RAE, and 23.741 % for RRSE are 0.0126 for MAE, 0.0888 for RMSE, 4.498 % for RAE, and 23.741 % for RRSE. Furthermore, each accuracy, recall, and F-measure achieved 0.980 outcomes. The CDT, MCC result is 0.975, with a 98% accuracy. As a result, this study recommends CDT for risk prediction in software requirements. Moreover, complete findings can be utilized as a starting point for other research. Any claim about improving prediction through a new model, approach, or framework may be benchmarked and evaluated. Class imbalance issues should be committed to the databases for future development. Furthermore, feature selection and ensemble learning strategies should be investigated to improve enactment. Moreover, this research may be utilized to identify the optimal classifier for developing and deploying a model for risk prediction in software requirements.

REFERENCES

[1] M. Yaseen, A. Mustapha, and N. Ibrahim, "An approach for managing large-sized software requirements during prioritization," in *Proc. IEEE Conf. Open Syst. (ICOS)*, Nov. 2018, pp. 98–103, doi: 10.1109/ICOS.2018.8632806.

[2] B. B. Duarte, A. L. de Castro Leal, R. de Almeida Falbo, G. Guizzardi, R. S. S. Guizzardi, and V. E. S. Souza, "Ontological foundations for software requirements with a focus on requirements at runtime," *Appl. Ontol.*, vol. 13, no. 2, pp. 73–105, May 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, Jun. 2020, doi: 10.1049/iet-sen.2019.0180.

[5] M. M. Otoom, "ABMJ: An ensemble model for risk prediction in software requirements," *Int. J. Comput. Sci. Netw. Secur.*, vol. 22, no. 3, p. 710, 2022.

[6] Z. S. Shaukat, R. Naseem, and M. Zubair, "A dataset for software requirements risk prediction," in *Proc. IEEE Int. Conf. Comput. Sci. Eng. (CSE)*, Oct. 2018, pp. 112–118, doi: 10.1109/CSE.2018.00022.

[7] F. U. Hassan and T. Le, "Automated prioritization of requirements to support risk-based construction inspection of highway projects using LSTM neural network," in *Proc. Construct. Res. Congr.*, 2022, pp. 1270–1277.

[8] J. Dhlamini and I. Nhamu, "React reactive proa proactive," ACM Press, Eastern Cape, South Africa, Tech. Rep. 24, 2009, pp. 33–40.

[9] B. Charbuty and A. Abdulazeez, "Classification based on decision tree algorithm for machine learning," *J. Appl. Sci. Technol. Trends*, vol. 2, no. 1, pp. 20–28, Mar. 2021, doi: 10.38094/jastt20165.

[10] T. Zhang, R. P. Quevedo, H. Wang, Q. Fu, D. Luo, T. Wang, G. G. de Oliveira, L. A. Guasselli, and C. D. Renno, "Improved tree-based machine learning algorithms combining with bagging strategy for landslide susceptibility modeling," *Arab. J. Geosci.*, vol. 15, no. 2, pp. 1–19, 2022.

[11] 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.

[12] B. Khan, R. Naseem, M. A. Shah, K. Wakil, A. Khan, M. I. Uddin, and M. Mahmoud, "Software defect prediction for healthcare big data: An empirical evaluation of machine learning techniques," *J. Healthcare Eng.*, vol. 2021, pp. 1–16, Mar. 2021, doi: 10.1155/2021/8899263.

[13] B. Khan, R. Naseem, M. Ali, M. Arshad, and N. Jan, "Machine learning approaches for liver disease diagnosing," *Int. J. Data Sci. Adv. Anal.*, vol. 1, no. 1, pp. 27–31, 2019.

[14] T. M. Carvajal, K. M. Viacrusis, L. F. T. Hernandez, H. T. Ho, D. M. Amalin, and K. Watanabe, "Machine learning methods reveal the temporal pattern of dengue incidence using meteorological factors in metropolitan manila, Philippines," *BMC Infectious Diseases*, vol. 18, no. 1, pp. 1–15, Dec. 2018, doi: 10.1186/s12879-018-3066-0.

[15] C. G. Raji and S. S. V. Chandra, "Graft survival prediction in liver transplantation using artificial neural network models," *J. Comput. Sci.*, vol. 16, pp. 72–78, Sep. 2016, doi: 10.1016/j.jocs.2016.05.005.

[16] P. Guo, T. Liu, Q. Zhang, L. Wang, and J. Xiao, "Developing a dengue forecast model using machine learning: A case study in China," *PLoS Negl. Trop. Dis.*, vol. 11, no. 10, pp. 1–22, 2017, doi: 10.1371/journal.pntd.0005973.

[17] A. Al-Anazi and I. D. Gates, "A support vector machine algorithm to classify lithofacies and model permeability in heterogeneous reservoirs," *Eng. Geol.*, vol. 114, nos. 3–4, pp. 267–277, Aug. 2010, doi: 10.1016/j.enggeo.2010.05.005.

[18] T. Menzies, A. Dekhtyar, J. Distefano, and J. Greenwald, "Problems with precision: A response to 'comments on 'data mining static code attributes to learn defect Predictors,''' *IEEE Trans. Softw. Eng.*, vol. 33, no. 9, pp. 637–640, Sep. 2007, doi: 10.1109/TSE.2007.70721.

[19] S. Vanderbeck, J. Bockhorst, R. Komorowski, D. E. Kleiner, and S. Gawrieh, "Automatic classification of white regions in liver biopsies by supervised machine learning," *Hum. Pathol.*, vol. 45, no. 4, pp. 785–792, Apr. 2014, doi: 10.1016/j.humpath.2013.11.011.

[20] H. Jin, S. Kim, and J. Kim, "Decision factors on effective liver patient data prediction," *Int. J. Bio-Sci. Bio-Technol.*, vol. 6, no. 4, pp. 167–178, 2014, doi: 10.14257/ijbsbt.2014.6.4.16.

[21] H. Tong, B. Liu, and S. Wang, "Software defect prediction using stacked denoising autoencoders and two-stage ensemble learning," *Inf. Softw. Technol.*, vol. 96, pp. 94–111, Apr. 2018, doi: 10.1016/j.infsof.2017.11.008.

[22] A. Iqbal, S. Aftab, U. Ali, Z. Nawaz, L. Sana, M. Ahmad, and A. Husen, "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.

[23] S. Jacob and G. Raju, "Software defect prediction in large space systems through hybrid feature selection and classification," *Int. Arab J. Inf. Technol.*, vol. 14, no. 2, pp. 208–214, 2017.

[24] A. Anand, L. Wilkinson, and D. N. Tuan, "An L-infinity norm visual classifier," in *Proc. 9th IEEE Int. Conf. Data Mining*, Dec. 2009, pp. 687–692, doi: 10.1109/ICDM.2009.119.

[25] C. Manjula and L. Florence, "Deep neural network based hybrid approach for software defect prediction using software metrics," *Cluster Comput.*, vol. 22, no. 4, pp. 9847–9863, 2019, doi: 10.1007/s10586-018-1696-z.

[26] 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.*, vol. 441, pp. 152–170, May 2018, doi: 10.1016/j.ins.2018.02.027.

[27] R. Malhotra and S. Kamal, "An empirical study to investigate oversampling methods for improving software defect prediction using imbalanced data," *Neurocomputing*, vol. 343, pp. 120–140, May 2019, doi: 10.1016/j.neucom.2018.04.090.

[28] J. Chen, Y. Yang, K. Hu, Q. Xuan, Y. Liu, and C. Yang, "Multiview transfer learning for software defect prediction," *IEEE Access*, vol. 7, pp. 8901–8916, 2019, doi: 10.1109/ACCESS.2018.2890733.

[29] C. J. Mantas and J. Abellán, "Credal decision trees in noisy domains," in *Proc. 22nd Eur. Symp. Artif. Neural Netw., Comput. Intell. Mach. Learn. (ESANN)*, Apr. 2014, pp. 683–688.

[30] Q. He, Z. Xu, S. Li, R. Li, S. Zhang, N. Wang, B. T. Pham, and W. Chen, "Novel entropy and rotation forest-based credal decision tree classifier for landslide susceptibility modeling," *Entropy*, vol. 21, no. 2, p. 106, Jan. 2019, doi: 10.3390/e21020106.

[31] J. Abellán and A. R. Masegosa, "An ensemble method using credal decision trees," *Eur. J. Oper. Res.*, vol. 205, no. 1, pp. 218–226, Aug. 2010, doi: 10.1016/j.ejor.2009.12.003.

[32] M. J. Siers and M. Z. Islam, "Cost sensitive decision forest and voting for software defect prediction," in *Trends in Artificial Intelligence* (Lecture Notes in Computer Science), vol. 8862. Cham, Switzerland: Springer, 2014, pp. 929–936, doi: 10.1007/978-3-319-13560-1.

[33] R. Naseem, B. Khan, A. Ahmad, A. Almogren, S. Jabeen, B. Hayat, and M. A. Shah, "Investigating tree family machine learning techniques for a predictive system to unveil software defects," *Complexity*, vol. 2020, pp. 1–21, Nov. 2020, doi: 10.1155/2020/6688075.

[34] N. Nahar and F. Ara, "Liver disease prediction by using different decision tree techniques," *Int. J. Data Mining Knowl. Manage. Process*, vol. 8, no. 2, pp. 1–9, Mar. 2018, doi: 10.5121/ijdkp.2018.8201.

[35] L. Wilkinson, A. Anand, and D. N. Tuan, "CHIRP: A new classifier based on composite hypercubes on iterated random projections," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2011, pp. 6–14, doi: 10.1145/2020408.2020418.

[36] N. Adnan and Z. Islam, "PT U.S. CR," *Expert Syst. Appl.*, vol. 174, Jun. 2020, doi: 10.1016/j.eswa.2017.08.002.

[37] M. M. Saritas and A. Yasar, "Performance analysis of ANN and naive Bayes classification algorithm for data classification," *Int. J. Intell. Syst. Appl. Eng.*, vol. 7, no. 2, pp. 88–91, Jan. 2019, doi: 10.18201/ijisae.2019252786.

[38] S. Perveen, M. Shahbaz, K. Keshavjee, and A. Guergachi, "A systematic machine learning based approach for the diagnosis of non-alcoholic fatty liver disease risk and progression," *Sci. Rep.*, vol. 8, no. 1, pp. 1–12, Dec. 2018, doi: 10.1038/s41598-018-20166-x.

[39] J.-J. Liu and J.-C. Liu, "An intelligent approach for reservoir quality evaluation in tight sandstone reservoir using gradient boosting decision tree algorithm—A case study of the Yanchang formation, mid-eastern Ordos Basin, China," *Mar. Pet. Geol.*, vol. 126, Apr. 2021, Art. no. 104939.

[40] M. N. Kumar, K. V. S. Koushik, and K. Deepak, "Prediction of heart diseases using data mining and machine learning algorithms and techniques | request PDF," *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, vol. 3, no. 3, pp. 887–898, 2018, doi: 10.13140/RG.2.2.28488.83203.

[41] A. N. Arbain and B. Y. P. Balakrishnan, "A comparison of data mining algorithms for liver disease prediction on imbalanced data," *Int. J. Data Sci. Adv. Anal.*, vol. 1, no. 1, pp. 1–11, 2019.

[42] A. Gulia, R. Vohra, and P. Rani, "Liver patient classification using intelligent techniques," *Int. J. Comput. Sci. Inf. Technol.*, vol. 5, no. 4, pp. 5110–5115, 2014.

[43] K. S. Dar and S. M. U. Azmeen, "Dengue fever prediction: A data mining problem," *J. Data Mining Genomics Proteomics*, vol. 6, no. 3, pp. 1–5, 2015, doi: 10.4172/2153-0602.1000181.

[44] M. Siavvas, D. Tsoukalas, M. Jankovic, D. Kehagias, and D. Tzovaras, "Technical debt as an indicator of software security risk: A machine learning approach for software development enterprises," *Enterprise Inf. Syst.*, vol. 16, no. 5, p. 1824017, 2022.

[45] M. N. Mahdi, M. H. M. Zabil, A. R. Ahmad, R. Ismail, Y. Yusoff, L. K. Cheng, M. S. B. M. Azmi, H. Natiq, and H. H. Naidu, "Software project management using machine learning technique—A Review," *Appl. Sci.*, vol. 11, no. 11, p. 5183, 2021.

[46] I. Alam and S. Khusro, "Tailoring recommendations to groups of viewers on smart TV: A real-time profile generation approach," *IEEE Access*, vol. 8, pp. 50814–50827, 2020.

[47] C. López-Martín, "Machine learning techniques for software testing effort prediction," *Softw. Quality J.*, vol. 30, no. 1, pp. 65–100, 2022.

[48] M. N. Uddin, B. Li, Z. Ali, P. Kefalas, I. Khan, and I. Zada, "Software defect prediction employing BiLSTM and BERT-based semantic feature," *Soft Comput.*, vol. 26, pp. 7877–7891, 2022.

BILAL KHAN is currently pursuing the Ph.D. degree in computer software engineering with the University of Engineering and Technology, Mardan, Pakistan. He is also working as a Lecturer at the Department of Computer Science, City University of Science and Information Technology, Peshawar, Pakistan. His research interests include natural language processing, machine learning, and software engineering.

RASHID NASEEM was born in Landikotal, Khyber Pakhtunkhwa, Pakistan. He received the B.C.S. degree in computer science from the University of Peshawar, Pakistan, in 2008, the M.Phil. degree in computer science from Quaid-I Azam University, Pakistan, in 2011, and the Ph.D. degree in information technology from the Universiti Tun Hussein Onn Malaysia, in February 2017. He was with software industry, from 2007 to 2008. He is currently an Assistant Professor of software engineering at the Pak-Austria Fachhochschule: Institute of Applied Sciences and Technology, Haripur, Pakistan. Before, he has been a Lecturer at the Department of Computer Science, City University of Science and Information Technology, Peshawar, Pakistan, since 2012, and was promoted to an Assistant Professor, in November 2017.

IFTIKHAR ALAM received the M.S. and Ph.D. degrees in computer science from the Department of Computer Science, University of Peshawar, Pakistan. He is currently working as an Assistant Professor of computer science with the City University of Science and Information Technology, Peshawar, Pakistan. His Ph.D. research is user/group modeling on smart TV for enhancing personalization services in general and recommendations in specific. He published several papers in international journals and conferences. His research interests include software engineering, recommender systems, user modeling, group modeling, smart TV, ubiquitous computing, web mining, search engines, augmented reality, and mobile-based systems for people with special needs.

INAYAT KHAN received the Ph.D. degree in computer science from the Department of Computer Science, University of Peshawar, Pakistan. His current research is based on the design and development of context-aware adaptive user interfaces for minimizing drivers' distractions. His research interests include lifelogging, healthcare, deep learning, ubiquitous computing, accessibility, and mobile-based assistive systems for people with special needs. He published several papers in international journals and conferences in these areas.

HISHAM ALASMARY received the M.Sc. degree in computer science from The George Washington University, Washington, DC, USA, in 2016, and the Ph.D. degree from the Department of Computer Science, University of Central Florida, in 2020. He is currently an Assistant Professor at King Khalid University. His research interests include software security, the IoT security and privacy, ML/DL applications in information security, and adversarial machine learning.

TAJ RAHMAN received the B.S. degree in computer science from the University of Malakand Dir (Lower) Pakistan, in 2007, the M.S. degree in computer science from Agriculture University, Peshawar, Pakistan, in 2011, and the Ph.D. degree in computer science from the School of Computer and Communication Engineering, University of Science and Technology Beijing, China. He is currently working as an Assistant Professor at the Department of Physical and Numerical Sciences, Qurtuba University of Science and Technology, Peshawar. His research interests include wireless sensor networks and the Internet of Things.