Received 15 January 2024, accepted 13 April 2024, date of publication 22 April 2024, date of current version 3 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3392283

RESEARCH ARTICLE

FEPP: Advancing Software Risk Prediction in Requirements Engineering Through Innovative Rule Extraction and Multi-Class Integration

MUHAMMAD BINSAWAD[1] **AND BILAL KHAN**[2]
[1]Department of Information Systems, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
[2]Department of Computer Science, City University of Science and Information Technology, Peshawar 25000, Pakistan

Corresponding author: Bilal Khan (bilalsoft63@gmail.com)

ABSTRACT The increasing complexity of software projects makes it difficult to predict risks in software requirements, which is a crucial and essential part of the Software Development Life Cycle (SDLC). The failure of a software project may occur from an inability to appropriately anticipate such risks. Because it is the first stage of any software project, risk prediction has a greater significance in software requirements. Thus, ForExPlusPlus (FEPP), a novel model for risk prediction in software requirements, is proposed in this work. Standard models such as K-nearest Neighbor (KNN), Naïve Bayes (NB), Logistic Model Tree (LMT), Random Forest (RF), and Support Vector Machine (SVM) are used to benchmark the suggested model. The dataset from the Zenodo repository is used to train these models, and standard assessment criteria are used to evaluate the results. The accuracy analysis of the models is assessed critically using the precision, F-measure (FM), and Mathew's correlation coefficient (MCC), as well as the error rate using the Kappa Statistic (KS) and Mean Absolute Error (MAE). The suggested FEPP performs better overall, with an accuracy of 96.84%, whereas KNN performs the worst, with an accuracy of 50.99%.

INDEX TERMS Software risk prediction, ForExPlusPlus, requirements engineering, Zenodo datasets.

I. INTRODUCTION

In the ever-changing world of software development, precise prediction of risks connected with software requirements has become a vital part of project success. As technology progresses and software systems become more complex, the necessity for strong risk prediction models becomes more obvious. In recent years, the discipline of software engineering has seen remarkable expansion and complexity, needing a thorough awareness of possible hazards throughout the requirements phase [1]. Early risk assessment and mitigation are critical for effective software development, minimizing costly rework, and improving overall project efficiency. Previous research has offered significant insights; nevertheless, with the shifting nature of software projects, there is a need for fresh and adaptable ways to risk prediction [2].

Predicting risk in software requirements is critical throughout the Software Development Life Cycle (SDLC) since it allows for early detection and mitigation of possible difficulties. This proactive strategy reduces costly rework, improves project planning by permitting better resource allocation and realistic timetables, and improves overall product quality. Transparent communication with stakeholders is enhanced, which fosters trust and collaboration [3]. The capacity to identify and manage risks also allows teams to adapt to changing requirements, fostering a culture of continuous improvement. Early risk assessment ensures compliance with standards in regulated businesses, avoiding legal and financial consequences. Risk management success contributes

The associate editor coordinating the review of this manuscript and approving it for publication was Porfirio Tramontana.

considerably to project success by meeting or surpassing client expectations, improving overall results, and delivering a successful software product [4].

Effective risk prediction in software requirements is more than an academic pursuit; it is a real need with far-reaching consequences. Unexpected challenges during the requirements phase can cause project delays, budget overruns, and, in the worst-case scenario, project collapse. Identifying and managing possible risks on time allows project managers, developers, and stakeholders to make educated decisions, eventually contributing to the successful delivery of high-quality software products [5].

This study adds considerably to the advancement of risk prediction approaches by presenting novel models, ForEx-PlusPlus (FEPP). This model is intended not just to improve forecast accuracy, but also to provide the software engineering community a better understanding of the value of features in risk assessment. The primary goal of this work is to undertake a thorough comparison of these innovative risk prediction models to existing benchmarks such as K-nearest neighbor (KNN), logistic model tree (LMT), Naïve Bayes (NB), support vector machine (SVM), and Random Forest (RF). The evaluation of these models includes a full set of standard metrics, including the Kappa Statistic (KS) and mean absolute error (MAE) for a thorough examination of error rates. Furthermore, accuracy is assessed using a variety of performance measures, including the true positive rate (TPR), false positive rate (FPR), precision, f-measure (FM), accuracy, and Mathew's correlation coefficient (MCC). This analytical approach offers a full and nuanced examination, offering insights into the error dynamics as well as the accuracy levels attained by the models under consideration. We hope to establish the usefulness of the FEPP model in collecting and forecasting risks associated with software requirements through empirical analysis. Furthermore, this study aims to uncover the distinct contributions of FEPP by evaluating their adaptability across various project contexts.

Through the introduction of FEPP, a new risk prediction model, this research makes a substantial contribution to the field of software risk prediction. In addition to improving prediction accuracy, FEPP is intended to give software requirements risk assessment teams a more sophisticated grasp of feature value. The algorithmic improvements that set FEPP apart from other models are its prioritization of shorter rules for better interpretability and its consideration of individual class features during the rule extraction process. A variety of common measures are used in the study thorough comparison of FEPP with recognized benchmarks (KNN, LMT, NB, SVM, and RF). A thorough and technical study of FEPP's performance is provided by this extensive review, which takes into account KS, MAE, TPR, FPR, precision, FM, accuracy, and MCC.

The empirical study follows a methodical approach that ensures transparency and replicability. It is based on a carefully chosen dataset from the Zenodo repository. To prove robustness, the study process uses ten-fold cross-validation. The paper also presents the FEPP algorithmic framework, highlighting its systematic approach to rule extraction from decision forests. This architecture, which involves removing rules that are the same and combining rules from several classes, demonstrates how FEPP may be tailored to a variety of risk situations while still meeting software requirements. The technical results show that FEPP performs better than other models in several areas, which makes it a promising model for software development risk prediction. In general, this study makes a technical contribution by improving our knowledge of and ability to use risk prediction models in the intricate field of software engineering. Overall contribution if this study are:

- Introduction of FEPP, a novel risk prediction model tailored for software requirements.
- Comprehensive comparison of FEPP with established models, demonstrating its better performance.
- Empirical validation of FEPP's effectiveness using standardized evaluation metrics.
- Transparent methodology ensuring replicability, with rigorous dataset selection and cross-validation.
- Detailed algorithmic framework highlighting FEPP's unique features for rule extraction.

The rest of the paper is organized as follows: The literature study is presented in Section II. The experimental design and research technique are presented in Section III, the results are discussed in Section IV, and the study is finally concluded in Section V.

II. LITERATURE REVIEW

Software risks are described by the US Department of Defense (DoD) as indices that gauge the incapacity to accomplish predetermined goals because of financial, schedule, or technological limitations [6]. Boehm established software risk management in 1989, setting the stage for further research [7]. The three essential elements of risk management in software projects are risk detection, analysis, and control [8].

Numerous risk analysis methods, including Bayesian Belief Network (BBN), Influence Diagram (ID), artificial neural networks (ANN), Monte Carlo analysis (MC), and classification and regression tree (CART), have been effectively used to anticipate the risk of software projects. However, there are limitations, especially concerning BBN and ID, wherein subjective correlations result from the need to understand the relationships between risk factors. As more nodes are added to the network, the expandability of ANN is restricted, requiring further study on a conditional probability table.

The inherent hazards in the software project development process were highlighted by Hu et al. [9], who also emphasized the high failure rate of this method. They created an intelligent model utilizing SVM and ANN on software risk datasets taken from surveys to solve this. Over 70% of software project failures, according to another research [10], are connected to risk. They used information from 332 software

TABLE 1. The overall summary of the literature to present the key findings of each study.

Literature	Methodology	Key Findings
Boehm [7]	Risk Management Framework	Established foundational elements: risk detection, analysis, and control.
Hu et al. [9]	SVM and ANN	Proposed intelligent model using SVM and ANN.
[10]	NB Classifier	Over 70% of software project failures connected to risk; NB classifier useful for risk prediction.
Christiansen et al. [11]	MLR	Identified and controlled inherent risk factors using MLR.
Shaukat et al. [12]	Data Challenges in Risk Prediction	Emphasized role of requirement elicitation phase; proposed dataset addressing data challenges.
Alharbi et al. [13]	LR and REPTree	Achieved high accuracy in assessing risk levels in multi-project situations using Logistic Regression and REPTree.
Xu, Yang, et al. [14]	GA and DT	Combined GA with DT for performance gains in risk prediction.
Xu, Zhang, et al. [15]	BNCC	Presented Bayesian Network-based framework outperforming existing machine learning techniques.
Akumba et al. [16]	NB Classifier	Utilized NB classifier during requirement elicitation; highlighted significance of probability and priority in risk forecasting.
Naseem et al. [4]	Various ML Methods	Stress on addressing class imbalance in risk datasets to avoid overfitting; empirical studies of machine learning methods.

projects to test an NB classifier for software risk prediction, and they concluded that risk prediction helps prioritize projects according to risk value. Christiansen et al. [11] underlined the significance of identifying and controlling inherent risk factors in software development processes by using MLR to assess software development risk based on survey data.

Shaukat et al. [12] highlighted the requirement elicitation phase of the SDLC as critical to addressing data challenges in software risk prediction. They also proposed a dataset for this phase that included needs from the software requirements specification (SRS) and their associated risk characteristics.

An analytical viewpoint on risk assessment across many software projects that are being performed concurrently was offered by Alharbi et al. [13]. In assessing risk levels in a multi-project situation, their research showed great accuracy, with logistic regression (LR) recording a 93% accuracy and REPTree recording a 98% accuracy. For software risk prediction, Xu et al. [14] combined a genetic algorithm (GA) with a decision tree (DT), demonstrating notable performance gains. A Bayesian network (BN)–based framework with causality constraints (BNCC) was presented by Hu et al. [15], outperforming existing machine learning techniques including LR, DT, NB, and traditional BN.

An NB classifier was used in risk prediction by Akumba et al. [16] during the SDLC's requirement elicitation stage. Based on the characteristics of the risk dataset, the NB model emphasized the significance of probability and priority in forecasting risk levels. In their empirical study of machine learning methods for software risk prediction, Naseem et al. [4] emphasized that to avoid overfitting, class imbalance in risk datasets must be addressed.

All things considered, these collaborative research projects greatly improve our knowledge of and capacity to deal with the complex issues raised by software project risks. The topic is advanced by the investigation of various approaches and strategies for risk analysis and prediction, which lays the groundwork for more reliable and efficient risk management in software development projects. Table 1 presents the overall summary of the literature for more better understanding the key findings of the literature.

A. HOW THIS RESEARCH STUDY IS DIFFERENT FROM THE PAST STUDIES MENTIONED IN THE LITERATURE?

This research study differs from previous studies in software risk prediction by introducing the FEPP model, which marks a significant development in the area. While addressing the traditional features of risk management in software projects, the study goes beyond existing approaches such as the Bayesian Belief Network and the Influence Diagram. FEPP combines a decision forest structure with novel algorithmic innovations, emphasizing the extraction of compact and interpretable rules. The study carefully compares FEPP's performance to benchmark models, resolving class imbalance in risk datasets and demonstrating its higher accuracy, precision, and recall. Furthermore, the study broadens the scope by emphasizing FEPP's flexibility to multiple risk situations through class-specific inspection and integration of rules from other classes, consistent with past literature's emphasis on the crucial significance of the requirement elicitation phase. In summary, the study article not only helps to overcome current obstacles but also establishes a new standard for reliable and interpretable software risk prediction.

III. EXPERIMENTAL SETUP AND RESEARCH METHODOLOGY

This study aims to present a cutting-edge risk prediction model that makes use of the capabilities of FEPP. The application of this approach to a rigorously selected dataset derived from the Zenodo repository serves as its base. To guarantee computational resilience, the experimental phase takes place on a powerful machine running Windows 10, with a 64-bit architecture, 12 GB of RAM, and a core i5 CPU. Figure 1 depicts the systematic flow of this investigation, which begins with the collection of the dataset and continues through a succession of painstakingly designed procedures. The first critical process is the rigorous cleansing of the dataset to improve its quality and integrity. next, the improved dataset serves as the foundation for training and testing the proposed model, following the standardized techniques mentioned in the next sections. The study's commitment to following rigorous and well-established methods throughout the testing phase is a noteworthy feature. Each used model's performance is evaluated with a critical eye, utilizing conventional assessment techniques to ensure a full examination of its predictive power. In summary, this study not only introduces a unique risk prediction model but also demonstrates a methodical and principled approach to its implementation, contributing to the larger conversation in the domain of software requirements and risk assessment.

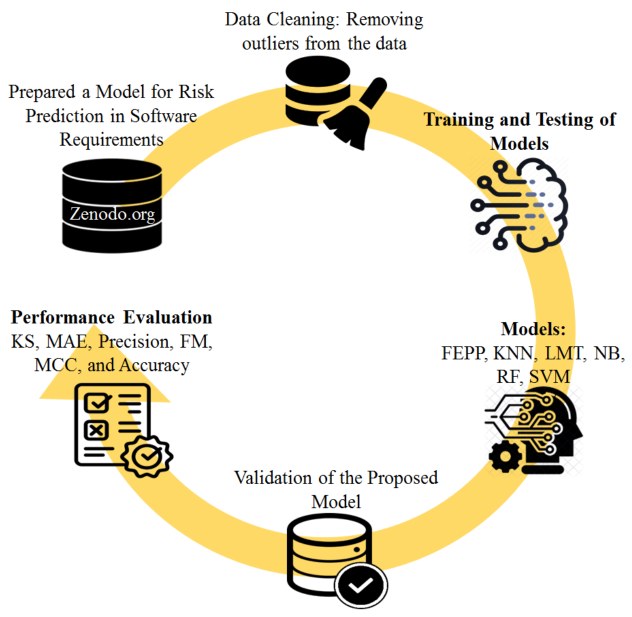

Data Cleaning: Removing outliers from the data
Prepared a Model for Risk Prediction in Software Requirements
Training and Testing of Models
Zenodo.org
Performance Evaluation
KS, MAE, Precision, FM, MCC, and Accuracy
Models:
FEPP, KNN, LMT, NB, RF, SVM
Validation of the Proposed Model

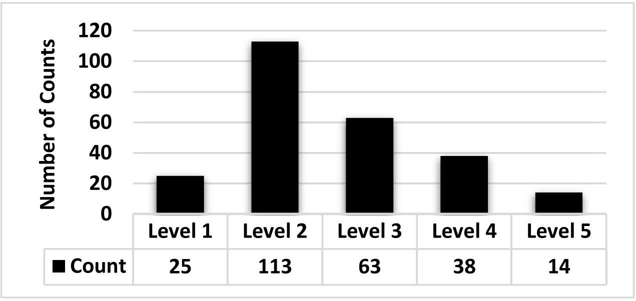
FIGURE 1. Systematic workflow for risk prediction model.

A. DATASET AND PREPROCESSING

The dataset used in this study originates from the Zenodo repository, which can be found at https://zenodo.org/records/1209601, and it has 13 characteristics and 253 occurrences. Table 2 lists these properties and their associated value types. The cases are divided into five categories marked by the numbers 1 through 5. The distribution of cases across various levels is seen in Figure 2. Data cleaning processes were used during the preprocessing step to correct anomalies and remove unneeded raw data from the dataset. The Isolation Forest approach was chosen expressly for outlier identification because of its capacity to find abnormalities via randomized partitioning inside a binary tree structure. The random selection of features and split values in this technique results in shorter pathways for anomalies. An ensemble of similar trees is built during training, and the average route length for each data point is determined. Shorter pathways suggest a greater chance of being an anomaly. Outlier scores are then assigned to data points, and a classification threshold is applied. Isolation Forest is renowned for its efficiency, particularly in high-dimensional datasets, and has proved useful in sectors such as fraud detection, network security, and quality control, where the discovery of rare abnormalities

TABLE 2. Attributes and characteristics of project risk management table, including data types and distinct values.

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



	Level 1	Level 2	Level 3	Level 4	Level 5
■ Count	25	113	63	38	14

FIGURE 2. Count of each level in the dataset.

among regular occurrences is critical. The steps of isolated forest are discussed in Algorithm 1.

Algorithm 1 Isolated Forest Algorithm

```
Input:
X – Dataset
T – number of Trees
S – Subsample Size
H – Maximum Tree Height
Output:
An ensemble of isolation trees
Function IsolationForest(X, T, S, H):
  Ensemble ← empty list
  For i in range(T):
  Subsample ← RandomlySelectedSubsample(X, S)
  Tree ← BuildIsolationTree(subsample, 0)
  ensemble.add(tree)
  return ensemble
Funstion BuildIsolationTree(subsample, current_height):
  If curret_height ≥ H
  Return CreatLeafNode(subsample)
  else:
  split_attribute, split_value ← RandomlySelectSplit(subsample)
  left_subsample, right_subsample ← SplitData(subsample, split_attribute, split_value)
  return
    CreateInternalNode(split_attribute, split_value,
    BuildIsolationTree(left_subsample, current_height + 1),
    BuildIsolationTree(right_subsample, current_height + 1)
Function RandomlySelectSubsample(data, sub_sample_size):
  reture data.sample(sub_sample_size)
Function RandomlySelectSplit(subsample):
  split_attribute ← RandomlyChooseAnAttribute(subsample)
  split_value ← RandomlyChooseValue(subsample, split_attribute)
  reture split_attribute, split_value
Function CreateLeafNode(subsample):
  return leafNode(subsample)
Function CreatInternalNode(split-attribute, split_value, left_child, right_child):
  return
  InternalNode(split_attribute, split_value, left_chiild, right_child)
Function RandomlyChooseAnAttribute(subsample):
  attribute ← sample(attribute(subsample))
  return attributes[0]
Function RandomlyChooseValue(subsample, attribute):
  value ← sample(value(subsample[attribute]))
  return values[0]
```

The Isolation Forest algorithm is a powerful outlier identification approach for detecting abnormalities in a dataset. It works by building an ensemble of isolation trees, each of which isolates anomalies in a binary tree structure via randomized partitioning. During training, the algorithm constructs these trees from random subsamples of the data, randomly picking features and splitting values. Anomalies in the trees are anticipated to have shorter pathways, and the method distributes outlier scores to data points based on the ensemble's average path length. The events are then classified as outliers or inliers using a threshold. Isolation Forest, known for its efficacy, particularly in high-dimensional data, has been effectively employed in a variety of fields such as fraud detection, network security, and quality control, where the discovery of rare abnormalities among regular cases is critical.

B. MODEL TRAINING AND PERFORMANCE ASSESSMENT

Model training and testing are critical phases that are at the core of each ML research project. Standardized criteria, namely 10-fold cross-validation, are utilized in this work to achieve these goals. Every model is tested using defined evaluation measures, including KS [17] and MAE [18] for error rate evaluation and TPR, FPR, precision, accuracy, FM, and MCC [19], [20], [21] for accuracy analysis. Here is an outline of how these metrics are calculated:

$$KS = \frac{P_0 - P_e}{1 - P_e} \tag{1}$$

where, P_0 is the observed agreement and it can be calculated as:

$$P_0 = \frac{a+d}{n} \tag{2}$$

P_e is the expected agreement, it can be calculated as:

$$P_e = \frac{(a+b)\,(a+c) + (c+d)(b+d)}{n^2} \tag{3}$$

where b is the number of observations where the second model allocates a different category from the first rater or model, and a is the number of observations where both models agree. The quantity c denotes the number of observations in which a distinct category is assigned by the first model, and a different category is assigned by the second model. n is the total number of data, and d is the number of observations for which the model assigns a distinct category.

$$MAE = \frac{1}{n}\sum\nolimits_{i=1}^{n} \left|y_i - \hat{y}_i\right| \tag{4}$$

where n is the total number of data points or observations. The i-th data point's actual (observed) value is denoted by y_i. The expected value for the i-th data point is $\hat{y}_i$. The absolute value is shown by $|\cdot|$.

$$TPR\ or\ Recall = \frac{TP}{TP+FN} \tag{5}$$

$$FPR = \frac{FP}{FP+TN} \tag{6}$$

$$Precision = \frac{TP}{TP+FP} \tag{7}$$

$$FM = \frac{2 * Precision * Recall}{Precision + Recall} \tag{8}$$

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

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{10}$$

where TP stands for the number of True Positives. The count of True Negatives is denoted by TN. FP stands for False Positives. FN represents the number of False Negatives.

C. PROPOSED AND BENCHMARKED MODELS

This study proposes a new model for risk prediction in software requirements which is FEPP. It is an algorithmic enhancement designed to enhance the rule extraction from decision forests for binary or multi-class classification applications [22]. By incorporating significant improvements, the method resolves some of the shortcomings found in ForEx. A noteworthy feature of FEPP is that it recognizes that shorter rules are often considered easier to understand. It makes sure that the presence of rules from minority classes is not overwhelmed by the majority class by evaluating correctness, coverage, and rule length individually for each class. By overcoming prior shortcomings and providing a more nuanced assessment of rule properties, the FEPP method provides a comprehensive and enhanced approach to rule extraction from decision forests [23].

The FEPP algorithm is a framework designed for knowledge discovery from decision forests. I can be described as:

1) DECISION TREE STRUCTURE

A two-dimensional data collection with a variety of numerical and category properties is used by FEPP. Decision trees are hierarchical structures used in the decision-making process. The leaves of the tree indicate ultimate outcomes or class labels, while the nodes contain attribute values for data segmentation. The pathways that connect the root and leaves are used to create rules that represent the connections between class labels and attributes. This tree-based method, which is the foundation of FEPP's knowledge discovery methodology, makes predictions for unlabeled records in testing data sets easier.

2) DECISION FOREST

FEPP uses a Random Forest algorithm that combines the Random Subspace and Bagging techniques. Decision trees are built on each of the bootstrap samples produced by bagging. Furthermore, FEPP presents "Forest by Penalizing Attributes (Forest PA)," a decision forest algorithm that methodically penalizes attributes while accounting for their effect at different stages. By deliberately penalizing qualities according to their testing levels, this strategy increases the decision forest's variety and improves the overall efficacy of FEPP.

3) RULE EXTRACTION

Rules in FEPP are decision tree routes that include conditions (antecedent) and a consequent (class label) that lead from the root node to the leaf node. To score rules according to their correctness, coverage, and rule length across the decision forest, the system uses equations. FEPP removes duplicate rules before rule extraction, which speeds up the knowledge discovery process that follows.

4) FEPP FRAMEWORK

Each class is evaluated separately for correctness, coverage, and rule length throughout the FEPP rule extraction process. Within each class, the algorithm finds rules that have higher accuracy and coverage than average. Moreover, FEPP emphasizes conciseness by taking rule length into account during the extraction procedure. Lastly, the method provides a complete set of extracted rules by integrating rules from various classes that satisfy particular requirements for coverage, correctness, and rule length.

Algorithm 2 described the overall steps for the FEPP algorithm framework.

Algorithm 2 ForExPlusPlus (FEPP)

```
Input: Decision forest rules R
Output: FEPP rules RFEPP
Begin
  $RFEPP \leftarrow Empty\ set$
  $k \leftarrow Number\ of\ distinct\ classes$
  // Step 1: Remove identical rules
  $R \leftarrow RemoveIdenticalRules(R)$
  // Step 2: Rule extraction for each class
  for i = 1 to k do
    $RAvgAcc \leftarrow CalculateAverageAccuracy(Ri)$
    $RAccSet \leftarrow SelectRulesByAccuracy(Ri,\ RAvgAcc)$
    $RAvgCov \leftarrow CalculateAverageCoverage(Ri)$
    $RCovSet \leftarrow SelectRulesByCoverage(Ri,\ RAvgCov)$
    $RAvgLen \leftarrow CalculateAverageRuleLength(Ri)$
    $RLenSet \leftarrow SelectRulesByLength(Ri,\ RAvgLen)$
    // Intersection of accuracy, coverage, and rule length sets
    $RClassSet \leftarrow RAccSet \cap RCovSet \cap RLenSet$
    // Add class-specific rules to the overall set
    $RFEPP \leftarrow RFEPP \cup RClassSet$
  end for
    // Step 3: Integration of rules from different classes
    $RFEPP \leftarrow IntegrateClassRules(RFEPP)$
    Return RFEPP
End
```

The following pseudocode summarizes the main FEPP framework steps:

a: ELIMINATE IDENTICAL RULES

To prevent duplication, this stage eliminates identical rules from the decision forest.

b: RULE EXTRACTION FOR EACH CLASS

It determines the average rule length, coverage, and correctness for each class. Next, a set of rules for each class is obtained by taking the intersection of the rules selected based on criteria such as correctness, coverage, and rule length.

c: INTEGRATION OF RULES FROM DIFFERENT CLASSES

In this stage, rules from various classes are combined to create the final set of FEPP rules.

The purpose of this algorithm is to produce refined rules, also known as FEPP rules (RFEPP), by decision forest rule processing. The procedure begins by generating an empty set RFEPP and figuring out how many different classes there are (k). The first stage is to extract the same rules from the input decision forest rules (R). The algorithm then loops through each class from 1 to k, extracting rules for each one. It determines the average accuracy (RAvgAcc) of the rules belonging to each class (i) and chooses a subset of rules based on accuracy (RAccSet). Rule subsets RCovSet and RLenSet, respectively, are produced by applying comparable processes to average coverage (RAvgCov) and average rule length (RAvgLen). Subsequently, the algorithm determines the point at which the sets of accuracy, coverage, and rule length cross, resulting in the creation of the class-specific rule set (RClassSet). The whole set of RFEPP is comprised of these class-specific rules. To produce the output FEPP rules (RFEPP), the last stage is integrating rules from several classes. By taking into account rule length, accuracy, and coverage across several decision forest types, this all-encompassing method guarantees the extraction of high-quality rules. Enhancing interpretability and performance, the technique provides a methodical and efficient approach for rule refining in decision forests.

The proposed model undergoes a technical comparison with established models, as detailed in Table 3.

TABLE 3. List of benchmarked models used in this study for risk prediction in software requirements.

Benchmarked Models	References
K-nearest Neighbour	[24]–[26]
Logistic Model Tree	[20], [27]
Naïve Bayes	[19], [28]
Support Vector Machine	[19], [21], [29]
Random Forest	[20], [21], [25]

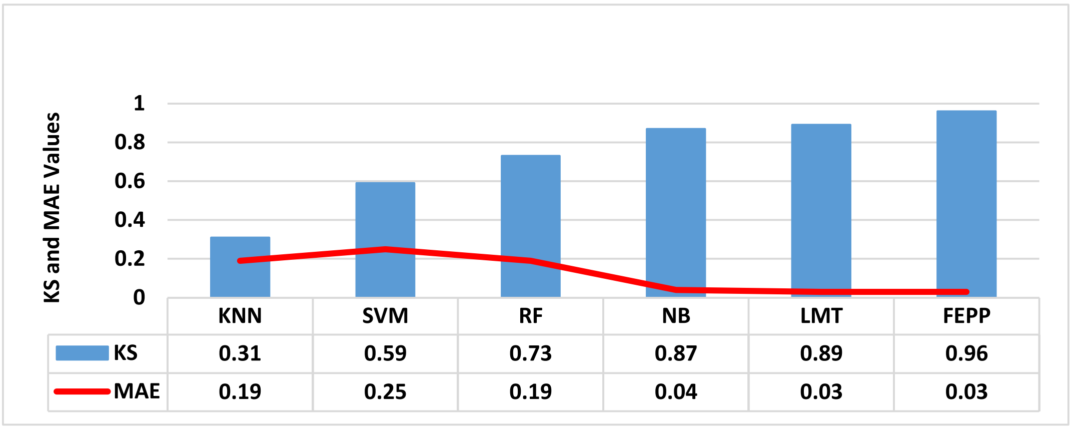
IV. RESULTS ANALYSIS AND DISCUSSION

This section discusses the experimental findings obtained from implementing both the proposed and benchmark models for risk prediction in software requirements. The analyses are divided into two parts: the first is an analysis of error rates, and the second is an evaluation of accuracy. In classification or prediction tasks, error rate analysis is critical for evaluating model performance, simplifying model comparison, giving interpretability, directing model fine-tuning, and providing insights into areas for development. KS evaluates agreement while accounting for chance, whereas MAE calculates the average absolute difference between predicted and actual values. These metrics help in the selection, refinement, and comprehension of predictive models in a variety of applications.

Figure 3 shows the KS and MAE values for different models used in software risk prediction. The KS represents the degree of agreement between expected and actual results, whereas the MAE is the average absolute difference between projected and actual values. The investigation reveals that the FEPP outperforms other models with a KS of 0.96 and an incredibly low MAE of 0.03 when compared to KNN, SVM, RF, NB, and LMT. The high KS shows that there is significant agreement between expected and actual outcomes, whilst the low MAE indicates that there are few mistakes in forecasting risk levels. The reasons for FEPP's superior performance might be ascribed to its intrinsic algorithmic characteristics, robust feature selection, and optimization methodologies, which lead to more accurate risk predictions in software requirements when compared to other models studied.

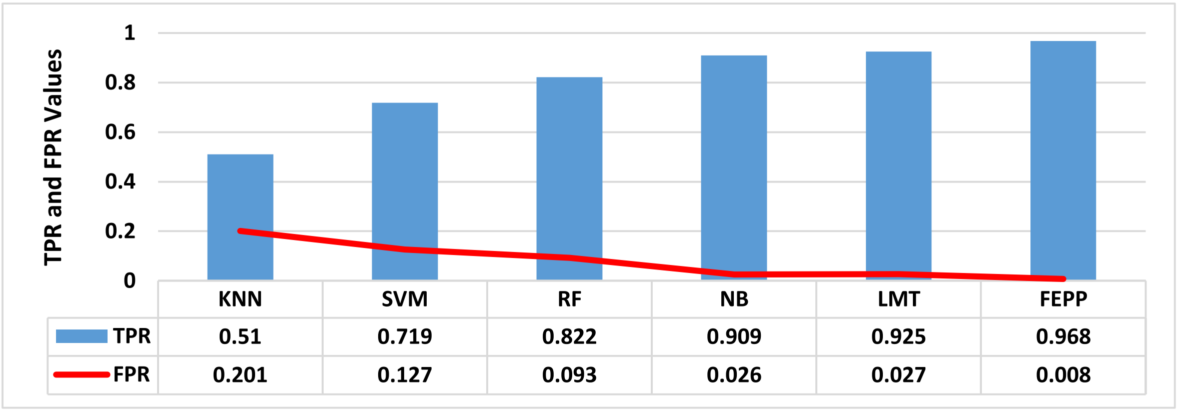
Figure 4 provides a critical assessment of the TPR and FPR for models that are used in risk prediction for software requirements, such as KNN, SVM, RF, NB, LMT, and the suggested FEPP model. The TPR denotes the percentage of real positive cases that the model properly detected, whereas the FPR shows the percentage of real negative cases that were mistakenly categorized as positives. After analyzing the data, it can be shown that the FEPP model performs better than the other models. It obtains a maximum TPR of 0.968, meaning that a significant proportion of real positive events are accurately identified. Concurrently, the FEPP model shows a remarkably low FPR of 0.008, indicating that it reduces the possibility of misclassifying real negatives as positives. This strong performance is important for risk prediction scenarios because it shows that the FEPP model can accurately identify and divide software requirements, which leads to more accurate risk assessments overall compared to other models that have been studied. The FEPP model's effectiveness in preventing false alarms is shown in the decreased FPR, which is especially significant for real-world software risk management applications.

Figure 5 presents a thorough analysis of MCC, FM, and precision. The percentage of actual positive predictions among all the model's positive predictions is known as precision. The FM is a balanced indicator of a model's overall performance; it is the harmonic mean of accuracy and recall. True positives, true negatives, false positives, and false negatives are all included in the MCC metric, which produces a number between -1 and $+1$, where $+1$ denotes flawless prediction. The FEPP model emerges as the best performer across all measures once the data is analyzed. It shows a remarkable F-measure of 0.969 and an MCC of 0.959, and reaches a precision of 0.969, suggesting a high accuracy in positive predictions. Together, these findings demonstrate how well the FEPP model performs in striking a compromise between precision and recall, producing precise and trustworthy risk forecasts for software needs. The model's capacity to handle unbalanced datasets and overall efficacy in capturing the genuine link between anticipated and actual outcomes are further highlighted by the high MCC. These results show that, in comparison to other models examined in this work, the FEPP model is especially well-suited for applications in software risk prediction.



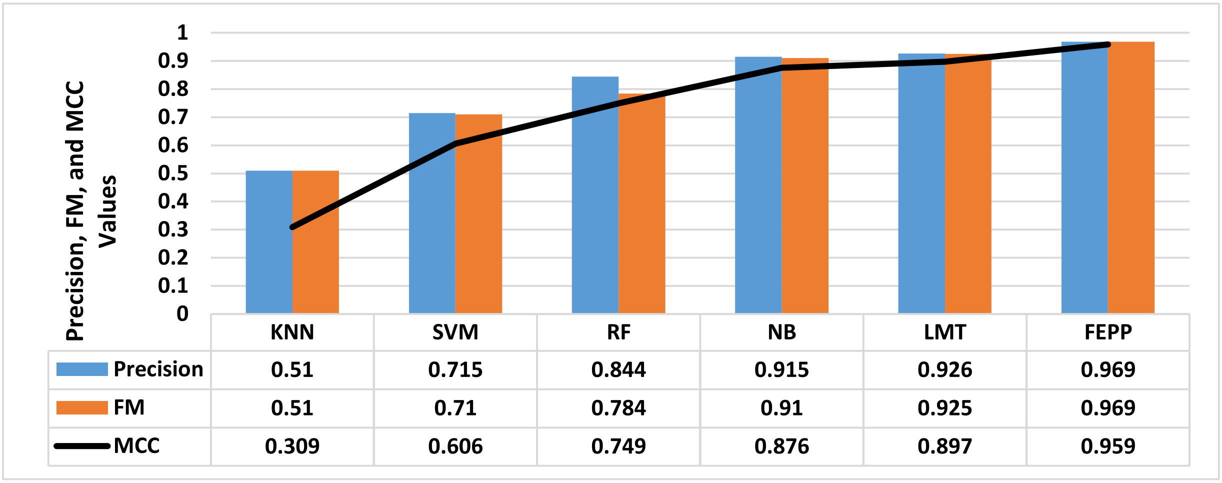
	KNN	SVM	RF	NB	LMT	FEPP
🟦 KS	0.31	0.59	0.73	0.87	0.89	0.96
▬ MAE	0.19	0.25	0.19	0.04	0.03	0.03

FIGURE 3. Analysis of kappa statistic (KS) and mean absolute error (MAE) across different models in software risk prediction.



	KNN	SVM	RF	NB	LMT	FEPP
🟦 TPR	0.51	0.719	0.822	0.909	0.925	0.968
▬ FPR	0.201	0.127	0.093	0.026	0.027	0.008

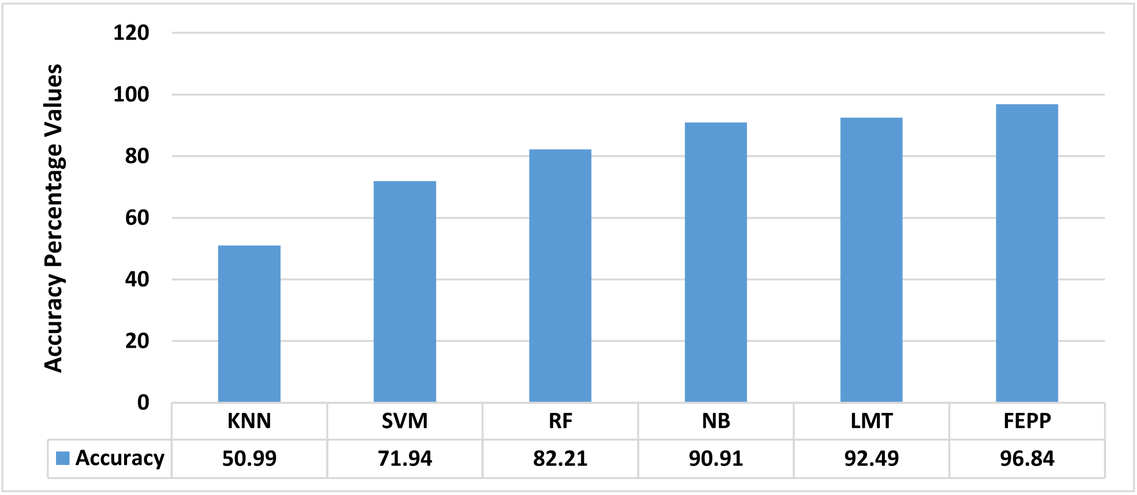
FIGURE 4. Evaluation of true positive rate (TPR) and false positive rate (FPR) across the employed models in software risk prediction.



	KNN	SVM	RF	NB	LMT	FEPP
🟦 Precision	0.51	0.715	0.844	0.915	0.926	0.969
🟧 FM	0.51	0.71	0.784	0.91	0.925	0.969
▬ MCC	0.309	0.606	0.749	0.876	0.897	0.959

FIGURE 5. Evaluation of precision, F-measure (FM), and matthews correlation coefficient (MCC) across the employed models in software risk prediction.

This creative method of fine-tuning rules using decision forests is what makes the FEPP model perform so much better. By recognizing that shorter rules are frequently simpler to grasp, FEPP, in contrast to standard models, focuses on extracting precise and succinct rules. A greater capacity to understand the underlying patterns in software



	KNN	SVM	RF	NB	LMT	FEPP
Accuracy	50.99	71.94	82.21	90.91	92.49	96.84

FIGURE 6. **Accuracy percentages of various models in this study for risk prediction in software requirements.**

requirements is made possible by this emphasis on rule length, which also makes the model more interpretable. Furthermore, FEPP protects against the dominance of rules from the majority class by taking into account each class separately during the rule extraction process, ensuring a balanced review. To further strengthen its prediction powers, FEPP demonstrates a strong feature selection process by systematic penalization of characteristics and optimization of decision forest structures. More accurate risk forecasts result from these algorithmic improvements, particularly when it comes to software requirements where interpretability and nuanced understanding are critical.

Furthermore, FEPP's ability to respond to a variety of risk scenarios within software requirements is demonstrated by its class-specific examination and integration of rules from various classes. Notable in particular is the ability of the model to handle imbalanced datasets well, which guarantees that the model's predictive capability spans a range of risk levels. By carefully integrating rules from several classes, the integration process produces a complete collection of improved rules that encapsulate the complexities of software risk prediction. The high MCC, TPR, and accuracy of FEPP show that it is effective in finding a compromise between minimizing false positives and correctly recognizing positive situations. Overall, FEPP performs better than the benchmarked models in this study because of its creative algorithmic design, emphasis on the conciseness of rules, and flexibility in various risk situations.

V. CONCLUSION

The study presents and assesses the FEPP risk prediction model against accepted benchmarks to handle the changing problems in software development. Compact and comprehensible rule extraction is prioritized by FEPP's novel algorithmic design, which also exhibits better performance in measures like KS, MAE, TPR, FPR, precision, FM, accuracy, and MCC. It excels in real-world software development with unbalanced datasets because of its flexibility to various risk scenarios, as shown by class-specific analysis and rule integration. The model's innovative ''Forest by Penalizing Attributes'' technique and complex decision forest structure are credited with its performance. In addition to minimizing risks, FEPP improves project planning, lowers rework expenses, and guarantees the delivery of high-caliber software solutions. FEPP's proven flexibility and efficacy make it a potential step for resilient and successful software development methods as software projects continue to change, encouraging proactive risk management and a continuous improvement culture.

Future study endeavors might include enhancing and perfecting the FEPP model, investigating its amalgamation with nascent technologies such as artificial intelligence, and executing pragmatic deployments and case analyses to authenticate its efficaciousness throughout multifarious software undertakings. Collaborations between academic institutions and businesses might make it easier for FEPP to be widely adopted and guarantee that it will remain relevant in the face of changing software development issues.

REFERENCES

[1] B. Khan, R. Naseem, I. Alam, I. Khan, H. Alasmary, and T. Rahman, ''Analysis of tree-family machine learning techniques for risk prediction in software requirements,'' *IEEE Access*, vol. 10, pp. 98220–98231, 2022.

[2] M. S. G. Qureshi, B. Khan, and M. Arshad, ''ML-based model for risk prediction in software requirements,'' *Int. J. Technol. Diffusion*, vol. 13, no. 1, pp. 1–17, Nov. 2022.

[3] H. Mamman, A. O. Balogun, S. Basri, L. F. Capretz, V. E. Adeyemo, A. A. Imam, and G. Kumar, ''Software requirement risk prediction using enhanced fuzzy induction models,'' *Electronics*, vol. 12, no. 18, p. 3805, Sep. 2023.

[4] R. Naseem, Z. Shaukat, M. Irfan, M. A. Shah, A. Ahmad, F. Muhammad, A. Glowacz, L. Dunai, J. Antonino-Daviu, and A. Sulaiman, ''Empirical assessment of machine learning techniques for software requirements risk prediction,'' *Electronics*, vol. 10, no. 2, p. 168, Jan. 2021.

[5] A. Iftikhar, M. Alam, R. Ahmed, S. Musa, and M. M. Su'ud, "Risk prediction by using artificial neural network in global software development," *Comput. Intell. Neurosci.*, vol. 2021, pp. 1–25, Dec. 2021.

[6] P. Pråcha and J. Skrbek, "Use of FURIA for improving task mining," *Acta Inf. Pragensia*, vol. 11, no. 2, pp. 241–253, Aug. 2022.

[7] B. Boehm, "Software risk management," in *Proc. Eur. Softw. Eng. Conf.*, 1989, pp. 1–19.

[8] H. Yong, C. Juhua, R. Zhenbang, M. Liu, and X. Kang, "A neural networks approach for software risk analysis," in *Proc. 6th IEEE Int. Conf. Data Mining Workshops (ICDMW)*, 2006, pp. 722–725.

[9] Y. Hu, X. Zhang, X. Sun, M. Liu, and J. Du, "An intelligent model for software project risk prediction," in *Proc. Int. Conf. Inf. Manag., Innov. Manag. Ind. Eng.*, 2009, pp. 629–632.

[10] T. Kawamura, T. Toma, and K. Takano, "Outcome prediction of software projects for information technology vendors," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manage. (IEEM)*, Dec. 2017, pp. 1733–1737.

[11] T. Christiansen, P. Wuttidittachotti, S. Prakancharoen, and S. A. Vallipakorn, "Prediction of risk factors of software development project by using multiple logistic regression," *ARPN J. Eng. Appl. Sci.*, vol. 10, no. 3, pp. 1324–1331, 2015.

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

[13] I. M. Alharbi, A. A. Alyoubi, M. Altuwairiqi, and M. A. Ellatif, "Analysis of risks assessment in multi software projects development environment using classification techniques," in *Proc. Adv. Mach. Learn. Technol. Appl. AMLTA*, 2021, pp. 845–854.

[14] Z. Xu, B. Yang, and P. Guo, "Software risk prediction based on the hybrid algorithm of genetic algorithm and decision tree," in *Proc. Int. Conf. Intell. Comput.*, 2007, pp. 266–274.

[15] Y. Hu, X. Zhang, E. W. T. Ngai, R. Cai, and M. Liu, "Software project risk analysis using Bayesian networks with causality constraints," *Decis. Support Syst.*, vol. 56, pp. 439–449, Dec. 2013.

[16] B. O. Akumba, S. U. Otor, I. Agaji, and B. T. Akumba, "A predictive risk model for software Projects' requirement gathering phase," *Int. J. Innov. Sci. Res. Technol.*, vol. 5, no. 6, pp. 231–236, Jun. 2020.

[17] T. M. Deist, "Machine learning algorithms for outcome prediction in (chemo)radiotherapy: An empirical comparison of classifiers," *Med. Phys.*, vol. 45, no. 7, pp. 3449–3459, Jul. 2018, doi: 10.1002/mp.12967.

[18] C. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," *Climate Res.*, vol. 30, pp. 79–82, 2005, doi: 10.3354/cr030079.

[19] R. Naseem, B. Khan, M. A. Shah, K. Wakil, A. Khan, W. Alosaimi, M. I. Uddin, and B. Alouffi, "Performance assessment of classification algorithms on early detection of liver syndrome," *J. Healthcare Eng.*, vol. 2020, pp. 1–13, Dec. 2020, doi:10.1155/2020/6680002.

[20] 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.

[21] 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.

[22] M. N. Adnan and M. Z. Islam, "ForEx++: A new framework for knowledge discovery from decision forests," *Australas. J. Inf. Syst.*, vol. 21, pp. 1–20, Nov. 2017, doi:10.3127/ajis.v21i0.1539.

[23] M. Mashayekhi and R. Gras, "Rule extraction from random forest: the RF+HC methods," in *Proc. 28th Can. Conf. Artif. Intell., Canadian AI*, Halifax, NS, Canada, Jun. 2015, pp. 223–237.

[24] 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.

[25] A. Zamir, H. U. Khan, T. Iqbal, N. Yousaf, F. Aslam, A. Anjum, and M. Hamdani, "Phishing web site detection using diverse machine learning algorithms," *Electron. Library*, vol. 38, no. 1, pp. 65–80, Mar. 2020, doi: 10.1108/el-05-2019-0118.

[26] P. Dini and S. Saponara, "Analysis, design, and comparison of machine-learning techniques for networking intrusion detection," *Designs*, vol. 5, no. 1, p. 9, Feb. 2021, doi: 10.3390/designs5010009.

[27] P. Motarwar, A. Duraphe, G. Suganya, and M. Premalatha, "Cognitive approach for heart disease prediction using machine learning," in *Proc. Int. Conf. Emerg. Trends Inf. Technol. Eng. (ic-ETITE)*, Feb. 2020, pp. 1–5, doi: 10.1109/ic-ETITE47903.2020.242.

[28] 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.

[29] E.-H.-A. Rady and A. S. Anwar, "Prediction of kidney disease stages using data mining algorithms," *Informat. Med. Unlocked*, vol. 15, 2019, Art. no. 100178, doi:10.1016/j.imu.2019.100178.

MUHAMMAD BINSAWAD received the master's degree in applied information technology and the post-baccalaureate certificate in information systems management from Towson University, U.S., and the Ph.D. degree in information systems from the University of Technology Sydney (UTS), in 2019. He has professional experiences in Information Systems Design, Digital Transformation, Business Analysis, IT Project Management, and Solid knowledge in the SDLC approach. He has taught several subjects in the area of information systems and software engineering with King Abdulaziz University and the University of Technology Sydney. He is currently an Associate Professor with the Department of Information Systems, Faculty of Computing and Information Technology, King Abdulaziz University. His research interests include and not limited to information systems modeling-services, digital transformation human–computer interaction (HCI), and empirical studies. Also, he actively involved in international research and events activities, and contributing to international conferences and journals.

BILAL KHAN is currently pursuing the Ph.D. degree in computer software engineering from the University of Engineering and Technology, Mardan, Pakistan. He is also a Lecturer with 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.