electronics

Article

Software Requirement Risk Prediction Using Enhanced Fuzzy Induction Models

Hussaini Mamman [1,2], Abdullateef Oluwagbemiga Balogun [1,*], Shuib Basri [1], Luiz Fernando Capretz [3,4], Victor Elijah Adeyemo [5], Abdullahi Abubakar Imam [6] and Ganesh Kumar [1]

1 Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Seri Iskandar 32610, Malaysia; hussaini_21000736@utp.edu.my (H.M.); shuib_basri@utp.edu.my (S.B.); ganesh_17005106@utp.edu.my (G.K.)
2 Department of Management and Information Technology, Abubakar Tafawa Balewa University, Bauchi 740272, Nigeria
3 Department of Electrical and Computer Engineering, Western University, London, ON N6A 5B9, Canada; lcapretz@uwo.ca
4 Division of Science, Yale-NUS College, Singapore 138533, Singapore
5 School of Built Environment, Engineering, and Computing, Leeds Beckett University, Headingley Campus, Leeds LS6 3QS, UK; v.adeyemo5225@student.leedsbeckett.ac.uk
6 School of Digital Sciences, Universiti Brunei Darussalam, Gadong BE1410, Brunei; abdullahi.imam@ubd.edu.bn
* Correspondence: abdullateef.ob@utp.edu.my

Abstract: The development of most modern software systems is accompanied by a significant level of uncertainty, which can be attributed to the unanticipated activities that may occur throughout the software development process. As these modern software systems become more complex and drawn out, escalating software project failure rates have become a critical concern. These unforeseeable uncertainties are known as software risks, and they emerge from many risk factors inherent to the numerous activities comprising the software development lifecycle (SDLC). Consequently, these software risks have resulted in massive revenue losses for software organizations. Hence, it is imperative to address these software risks, to curb future software system failures. The subjective risk assessment (SRM) method is regarded as a viable solution to software risk problems. However, it is inherently reliant on humans and, therefore, in certain situations, imprecise, due to its dependence on an expert's knowledge and experience. In addition, the SRM does not allow repeatability, as expertise is not easily exchanged across the different units working on a software project. Developing intelligent modelling methods that may offer more unbiased, reproducible, and explainable decision-making assistance in risk management is crucial. Hence, this research proposes enhanced fuzzy induction models for software requirement risk prediction. Specifically, the fuzzy unordered rule induction algorithm (FURIA), and its enhanced variants based on nested subset selection dichotomies, are developed for software requirement risk prediction. The suggested fuzzy induction models are based on the use of effective rule-stretching methods for the prediction process. Additionally, the proposed FURIA method is enhanced through the introduction of nested subset selection dichotomy concepts into its prediction process. The prediction performances of the proposed models are evaluated using a benchmark dataset, and are then compared with existing machine learning (ML)-based and rule-based software risk prediction models. From the experimental results, it was observed that the FURIA performed comparably, in most cases, to the rule-based and ML-based models. However, the FURIA nested dichotomy variants were superior in performance to the conventional FURIA method, and rule-based and ML-based methods, with the least accuracy, area under the curve (AUC), and Mathew's correlation coefficient (MCC), with values of approximately 98%.

Keywords: software requirement; software risk prediction; fuzzy induction model; nested dichotomy

Citation: Mamman, H.; Balogun, A.O.; Basri, S.; Capretz, L.F.; Adeyemo, V.E.; Imam, A.A.; Kumar, G. Software Requirement Risk Prediction Using Enhanced Fuzzy Induction Models. *Electronics* **2023**, *12*, 3805. https://doi.org/10.3390/electronics12183805

Academic Editor: Juan M. Corchado

Received: 30 June 2023
Revised: 22 August 2023
Accepted: 6 September 2023
Published: 8 September 2023

1. Introduction

The development of most software systems is associated with a high probability of uncertainty, and these uncertainties are ascribed to the unpredictable activities that may happen during the software development process. Increasing software project failure rates are now regarded as a pressing issue because modern software systems are now more complicated and interconnected [1,2]. These unpredictable uncertainties are known as software risks, and they result from multiple risk variables inherent to the many activities of the software development lifecycle (SDLC) [3,4]. Notably, these software risks have resulted in huge financial losses and effort wastage for software companies. Consequently, these software risks should be addressed as soon as possible, so that they do not become the reason for software system failures [5,6].

According to Standish Group research, 31.1% of software projects fail before successful delivery, and only 16.2% of software projects are finished on schedule and within budget. In global enterprises, the situation is much more dire: barely 9% of software projects are completed on schedule and within budget. Furthermore, often, finalized software projects do not cover all the originally specified requirements. Furthermore, approximately 42% of the elicited functional requirements of software projects are finalized by US-based tech corporations. Therefore, one feasible approach to software risk problems is to conduct risk assessment processes [1,7].

Risk assessment is the methodical process of assessing the possible risks associated with a proposed activity in the SDLC [8]. This assessment can be conducted either as a proactive or a reactive method. The reactive method is not a mature risk assessment approach, because it usually extends the scheduled project timeline, and uses more resources while lowering the quality and reliability of the software. On the other hand, the proactive assessment method, which is based on the use of machine learning (ML), aims to reduce the chances of software project failure [9]. Software risk prediction is useful at every phase of SDLC and, particularly, at the requirement elicitation phase, as it improves the software development process [1].

Subjective assessment or expert opinion is one of the techniques often employed in project risk management today [10,11]. However, because it relies on an expert's knowledge, subjective assessment is vulnerable to human error, and insights may be vague or ambiguous [12]. Furthermore, it does not allow repeatability, as expertise is not easily exchanged across the various units working on a software project [13]. As a result, it is critical that we develop intelligent modelling methods that offer more unbiased, reproducible, and explainable decision-making assistance in risk management [1,3].

The modelling of software risk prediction has been presented as consisting of four components, which are risk identification, risk analysis, risk prioritization, and risk measurement (dataset) [1]. Risk identification is the initial phase of the Software Risk Prediction Model, during which the Risk Manager or Project Manager identifies possible threats that have the potential to impact the project's duration, assets, or costs. A hazard is an adverse occurrence that, if it transpires, poses a disadvantage to the effective culmination of the project. The process is executed via a "checklist". The criteria described in the Software Requirements Specification (SRS) have been systematically evaluated and identified as potential risk concerns. These requirements have been marked as checked, to facilitate further analysis. The checklist is, thereafter, advanced to the subsequent stage [9]. Risk analysis involves the identification of potential risks, which are subsequently transformed into valuable information for decision-making. Risk analysis is utilized to evaluate the likelihood and impact of every threat. This process involves transforming the risks that are discovered into decision-making information [9]. Subsequently, each risk is assessed, and a determination is reached on both the likelihood and the severity of the risk. The risk dataset has an attribute called the "Risk Level", which serves the purpose of categorizing requirements into five distinct risk categories [1]. Risk prioritization is the final stage of the framework, in which the analyzed risks are assigned priority based on their significance. The criteria with a significant degree of risk are moved to the front of the list, while the

requirements with a low degree of risk are placed at the bottom. The software requirement risk dataset comprises a collection of data. Risk measures for software requirements can be found in the datasets available on Zenodo [2].

Several techniques have been proposed and developed for predicting software risks in the SDLC. For instance, [14] investigated the application of artificial neural networks (ANNs) and support vector machines (SVMs) for software risk prediction. The suggested methods were deployed on the dataset extracted from the questionnaire. Similarly, ensemble methods based on Bayesian network (BN) models have been proposed for software risk prediction [7]. The Influence Diagram (ID) model and Bayesian Belief Network (BBN) have also been implemented for software project risk prediction. However, ID and the BBN have two intrinsic disadvantages: both models require previous information, and both are difficult to extend [15,16]. Nonetheless, the ANN is commonly used as a successful risk prediction methodology. For example, [16] developed a successful ANN-based model for prediction. Furthermore, the ANN approach outperforms conventional methods, such as decision tree (DT) and multiple linear regression (MLR) in addressing complex software project risk prediction issues [16,17].

The objective of researchers is to provide sturdiness in the proper prediction of software project risk [1,3]. However, the efficacy in implementing an ML technique is contingent upon the proficiency of the selected ML technique, and the dataset utilized for its development [18,19]. Different ML approaches have been deployed for software risk prediction with comparably low prediction performances. This could be attributed to issues with data quality, such as disparities in the class labels, or the presence of irrelevant data attributes, which cause the performance of ML techniques to deteriorate [20].

Consequently, this research proposes rule-based models for software requirement risk prediction. Specifically, the Fuzzy Unordered Rule Induction Algorithm (FURIA) and its nested dichotomy variants were developed for software requirement risk prediction. The FURIA is a rule learner that employs fuzzy rules and an effective rule-stretching method for its classification processes. Fuzzy rules are more generic than conventional rules, as they create dynamic boundaries for classification processes. For instance, the conventional rules work with fixed decision boundaries, with abrupt transitions between different classes, thereby generating questionable and non-intuitive models [21,22]. The understandability of the FURIA is, thus, a major advantage over the alternative (black box) conventional rule and ML-based methods, and a major reason for its adoption in this research. For improved performances, variants of the FURIA are based on nested dichotomy subset selection techniques, such as random (default), furthest centroid, balanced data, and random-pair subset selection methods.

In summary, the following are the scientific contributions made via this study:

(1) The implementation of the FURIA model for software risk prediction;
(2) The development of nested FURIA dichotomies for software risk prediction;
(3) The empirical comparison of proposed FURIA and nested FURIA dichotomy variants with existing software risk prediction models.

The remainder of the article is structured as follows. Related and relevant studies are presented and discussed in Section 2. An exhaustive description of the experimental procedure and methods used is provided in Section 3. Section 4 outlines the experimental results and the observed research findings. Section 5 concludes this study, and recommends further research.

2. Related Works

The United States Department of Defense (DoD) describes software risks as measuring indices when general pre-defined objectives cannot be met as a result of budgetary, time, or technological constraints [23]. In 1989, Boehm introduced the concept of software risk management into software engineering practices, laying the groundwork for future study [24]. Risk identification, risk analysis, and risk control are all aspects of risk management in software projects [25].

BBN, ID, ANN, Monte Carlo analysis (MC), classification and regression tree (CART), and other risk analysis techniques have been successfully deployed in software project risk prediction. Nonetheless, the drawback of the BBN and ID is that they require knowledge of risk factor connections and the impact of the variables on the project results is inferred from the correlations. The relations and the conditional probability table are very subjective, having been learned via professional experience. Regarding the ANN, as new nodes are added to the network, new research is required, to obtain a new conditional probability table, limiting network expandability.

Hu et al. [14] demonstrated that the software project development process is inherently risky, and has a high failure rate. To reduce the risks, they developed an intelligent model capable of predicting and managing the risks latent in software projects. Specifically, the SVM and ANN were used on software risk datasets extracted from questionnaires. In another study by [26], they reported that more than 70% of software project failures are related to risk. As a solution, they deployed a Naïve Bayes (NB) classifier for software risk prediction on software datasets gathered from 332 software projects, via an online poll. They concluded that software risk prediction can assist organizations in prioritizing software projects based on risk value. Similarly, Christiansen et al. [27] utilized MLR in their research, to estimate software development risk based on data collected through surveys from certain experts. They used statistical integration to show the risk variables that were predicted and controlled by reducing risk throughout the software development processes. The categorization of the likelihood of software project failure or success was predicted using a mix of factor analysis and MLR. Their research ended with an emphasis on the fact that there are inherent risk factors in software development processes that must be recognized and handled if software projects are to be delivered and finished on time. However, as these studies are conducted on data extracted from questionnaires, their respective findings cannot be generalized.

In a bid to address the data issues in software risk prediction, Shaukat et al. [2] identified the requirement elicitation phase of the SDLC as the most essential and difficult. They observed that there was no clear dataset from real-world software projects that had the characteristics of software needs and associated risks that could be used to predict risks in future software projects. They thus suggested a dataset for the SDLC's requirement-collecting phase that includes requirements derived from the software requirements specification (SRS) of certain software projects, as well as their risk characteristics.

Alharbi et al. [3] offered an analytical perspective on risk assessment in the context of numerous software projects running concurrently. This research demonstrated excellent levels of accuracy. For instance, logistic regression (LR) showed a risk prediction accuracy of 93%, while REPTree recorded a 98% risk accuracy in evaluating risk levels in a multi-project scenario that was running concurrently. Moreover, Xu, Yang et al. [28] hybridized the genetic algorithm (GA) and decision tree (DT) method, which automatically selects the optimum metric subset for software risk prediction. The experimental findings demonstrated the method's viability, while also showing substantial increases in the prediction performance. Xu, Zhang et al. [11] presented a BN-based framework with causality constraints (BNCC) for software project risk analysis. From their report, it was observed that the suggested model not only identifies causalities via expert knowledge, but also outperforms other ML methods, such as LR, DT, NB, and conventional BN.

Akumba et al. [29] used an NB classifier in risk prediction during the requirement elicitation phase of the SDLC in certain software projects. The NB model was built based on risk dataset features, such as size, effect, likelihood, priority, and risk dimension, to decide whether they were catastrophic, high, moderate, low, or inconsequential. From the analyses, probability and priority were reported to be the most important factors in predicting risk levels. That is, it is beneficial to recognize that the likelihood of the risk happening is high, and the priority is also recorded in advance. Furthermore, Naseem et al. [1] conducted an extensive empirical analysis of the applicability of ML techniques in software risk prediction. The findings from the experimental results indicated that

ML techniques can be used for software risk prediction. However, the preceding studies on software risk prediction failed to consider the inherent class imbalance characteristic of software risk dataset(s) used for developing models. For instance, Naseem et al. [1] reported that the occurrence of imbalanced classes (labels) in the software risk dataset may affect the prediction performance of ML techniques. Moreover, existing studies have shown that models developed with imbalanced datasets tend to overfit.

Consequently, the FURIA and its enhanced variants based on nested subset selection dichotomies are deployed for software risk prediction and are proposed in this research.

3. Methodology

In this section, the research methodology utilized in this research work is extensively discussed. Specifically, the FURIA and its nested dichotomy variants are presented. The software risk dataset we used, and the experimental procedure, are also discussed.

3.1. Fuzzy Unordered Rule Induction Algorithm (FURIA)

The Fuzzy Unordered Rule Induction Algorithm (FURIA) is an enhanced version of the Repeated Incremental Pruning to Produce Error Reduction (RIPPER) and Incremental Reduced Error Pruning (IREP) algorithms. It was created to address the latent problems in the RIPPER and IREP regarding the rule generation process from a decision sub-space, and using one class for the primary prediction. These problems are known to lead to a systemic bias for the default class label. However, the FURIA tends to distinguish class labels without considering the default rule and the order of the class labels. Typically, the FURIA introduces an additional rule-stretching function that allows it to extend the generated rules, to render it generic for all data instances [30].

For its pruning process, the FURIA applies pruning for the creation of the replacement and revision rule. In this case, the original pruning strategy is applied, except in cases where the pruning strategy tries to remove all antecedents from a rule, thereby generating a default rule. In this case, the pruning will be aborted, and the unpruned rule will be used for the Minimum Description Length (MDL) comparison in the optimization phase. These pruning strategies are still sufficient to avoid overfitting the generated model.

To generate fuzzy rules, the FURIA searches for the optimal fuzzy extension of each rule, where a fuzzy extension is deployed as a rule of the same structure, but with a fuzzy interval. For instance, for intervals A_i on the initial rules as the cores $\left[\partial_x^{a,B}, \partial_x^{a,C}\right]$ of the sought intervals A_i^F, the issues are to ascertain optimal bounds for the generated supports. For the fuzzification of a single antecedent ($H_j \in A_i$), it is critical to consider only the relevant training instances T_t^i.

$$T_t^i = \{y = (y_1 \ldots y_k) \in T_t | A_i^F(y_j) > 0 \; for \; all \; j \neq i\} \subseteq T_t \tag{1}$$

T_t^i is split into subsets of positive (T_{t+}^i) and negative (T_{t-}^i) instances.
To determine the degree of fuzzification, the rule purity (RP) is utilized:

$$RP = \frac{r_i}{r_i + m_i} \tag{2}$$

where

$$r_i = \sum_{y \in T_{t+}^i} \alpha_{H_j}(y) \tag{3}$$

$$m_i = \sum_{y \in T_{t-}^i} \alpha_{H_j}(y) \tag{4}$$

The fuzzification is then realized for the antecedent with the largest purity. This is repeated until all antecedents have been fuzzified. It is worth noting that a simple fuzzification is always discovered—notably, the one that sets the support bound to the first occurrence behind the core bound. Although this fuzzification does not affect the purity of the training data, it is useful in classifying new instances.

In addition, the FURIA uses minimal generalizations of all rules as appropriate replacements for any data instance. A generalization or "stretching" of a rule is obtained via deleting one or more of its antecedents, and a generalization is minimal if it does not delete more antecedents than is necessary to cover the query instance. Thus, the minimal generalization of a rule is simply obtained by deleting all antecedents that are not satisfied by the query. Further detail on the FURIA is available in [31–33]. Figure 1 depicts the flowchart for the working operation of the FURIA.

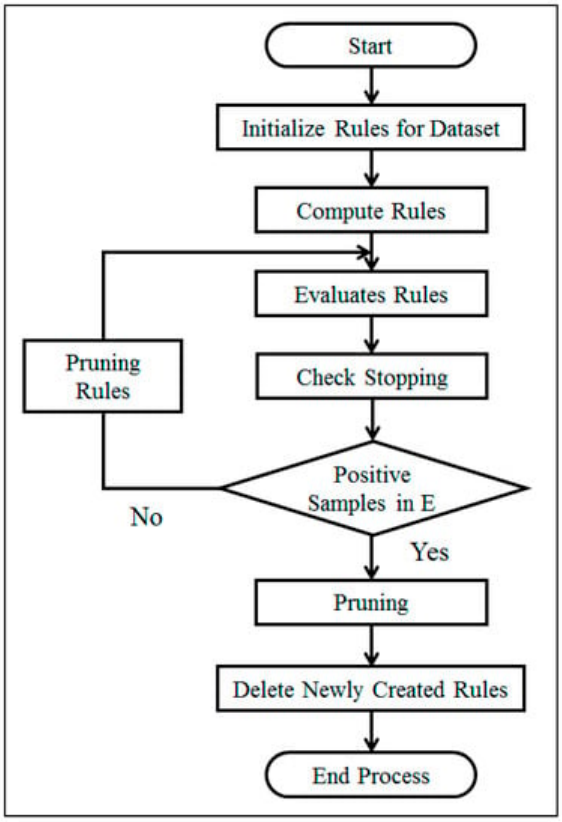

Figure 1. Flowchart for the working operation of the FURIA algorithm.

3.2. FURIA Nested Dichotomies Variants

A binary tree is utilized in a nested dichotomy approach to partition a multiclass problem into multiple binary classification problems, with the root node of the tree representing the complete set of problem classes. Subsequently, the classes are partitioned into two distinct subsets of classes, denoted as superclasses, and a model is formulated to discriminate between them. The procedure is iteratively executed, partitioning super classes until they comprise a solitary class from the initial set, signifying that each terminal node of the hierarchical structure exclusively encompasses a singular class. To categorize a novel entity, the tree that has been created is navigated through a binary framework, which selects the appropriate branch to follow at each level. Upon reaching a leaf, the object is assigned the corresponding class.

The rationale behind the Nested Dichotomy (ND) technique is that more dissimilar groups are better separated at the higher levels of a nested dichotomy because of the increased distance between them. The distance between classes is calculated by finding the centroid of each class, and then measuring the difference. After these distances are calculated, the classes are classified into two groups, say x_1 and x_2 as their centres, as they are the most distant from each other.

For simplicity, a nested dichotomy can be subdivided into two sequential steps: the construction step and the classification step.

1. Construction Step: This phase is primarily on the construction of the nested dichotomy. Assume we have a data collection of items that have been labelled with classes, and that every item has its own set of features.
    a. Select the class centroid that best represents the class means across all items in the data collection. When there are non-numeric features of a class, the item that is, on average, most like the others in that class is selected.
    b. The tree's starting point should consist of all the classes and their respective centres.
    c. Generate a dichotomy. This operation is repeatedly carried out on every node that has multiple classes until every tree leaf has exactly one class.
        i. Select the two classes whose centroids exhibit the highest degree of separation. Utilize the centroids that have been identified as the centres of the respective groups.
        ii. At the present node, each class is categorised based on its proximity to the nearest centre, as determined by the distance between the centroid of the class and the designated group centres. When the distance towards both centres is equivalent, the class is assigned to the initial group.
        iii. A new child node is generated for every group, and this procedure is iterated.
    d. Upon construction of the tree, a binary classifier is trained at every internal node to effectively distinguish between the groups of classes represented by their respective child nodes. In pursuit of this objective, every instance from the training set that matches the classes grouped at each child node is utilized.
2. Classification Step: For the classification process, the process of tree traversal commences at the root and proceeds by traversing the paths that correspond to each binary classifier until a leaf node is ultimately arrived at and the instance is assigned the class that is associated with the final tree leaf.

Nonetheless, as the nested dichotomy approach recursively applies binary splits to divide the set of classes into two subsets, and trains a binary classifier for each split, the split process can be deployed in several ways. In this research, we deployed four subset selection methods for the enhancement of the predictive performance of nested dichotomies based on random subset selection, furthest centroid subset selection, balanced subset selection, and random-pair subset selection.

3. Subset Selection Method

At every internal node i in a nested dichotomy, the collection of classes that exist at the node, denoted as N_1, is partitioned into two distinct and mutually exclusive subsets, namely N_{i1} and N_{i2}, both of which are non-empty. This section presents an overview of implemented class subset selection techniques for nested dichotomies. The techniques are primarily intended for employment in an ensemble context, wherein several ND decompositions are produced, each of which constitutes an ensemble member. The ensemble, in this context, is based on the randomness of the selection process.

a. Random Subset Selection

One of the fundamental approaches to selecting a subset of classes is to randomly divide the class set into two distinct subsets. This method offers various appealing characteristics. Firstly, the computational process is straightforward, and not dependent on the size of the training dataset, rendering it appropriate for extensive datasets. Moreover, in the context of a multi-class problem, the quantity of potentially nested dichotomies is exceedingly vast and can be expressed through a recurrence relation.

$$R(n) = (2n-3) \times R(n-1) \tag{5}$$

where $R(1) = 1$. This guarantees that in a nested dichotomy, there is a high degree of heterogeneity in the generated subsets. In this research work, the random subset selection method is deployed as the default subset selection method for the FURIA.

b. Furthest Centroid Subset Selection

The centroid-based techniques entail selecting each class split deterministically, using a distance metric. This implies that the arrangement of each embedded division within a collection will exhibit uniformity. The determination of the class radius involves the calculation of the spatial separation between the class centroid and the element situated at the maximum distance from the centroid, within the same class. Therefore, to quantify the distance between the two classes, C_1 and C_2, the centroid of each class (m_1, m_2) is utilized as a metric for the inter-class distance $d(m_1, m_2)$, and the radius of each class (r_1, r_2), using Equation (5), as follows.

$$D(C_1, C_2) = \frac{d(m_1, m_2)}{r_1 + r_2} \tag{6}$$

It should be noted that when $D = 1$, the classes are adjacent to each other, without any overlap. On the other hand, when $D > 1$, the classes are separate from each other. When $D < 1$, there is an overlap between the classes.

c. Balanced Data Subset Selection

One potential drawback associated with the utilization of random subset selection is the possibility of generating highly imbalanced tree structures. The quantity of internal nodes and, consequently, the quantity of binary models, is the determining factor. In any nest dichotomy with the same number of classes, an unbalanced tree frequently indicates that the training of internal binary models is conducted on extensive datasets located in the lower levels of the model hierarchy. As a result of this detrimental impact, the duration required to train the complete model is adversely affected. Additionally, subtrees that are further down in the hierarchical structure allow for a greater potential for the accumulation of estimation errors. [34] addressed this phenomenon, by partitioning N_i into two distinct subsets, N_{i1} and N_{i2}, with the constraint that the absolute difference between the cardinalities of N_{i1} and N_{i2} was no greater than one. Empirical evidence suggests that, in most cases, this approach has minimal impact on the accuracy, but it does result in a reduction in the training time required for nested dichotomies. The utilization of balanced selection is more advantageous in scenarios that involve many classes. The analysis indicates that, while the sample space of class-balanced nested dichotomy is smaller than that of random nested dichotomy, it remains sufficiently extensive to guarantee adequate random subset diversity.

$$R_{CB}(n) = \begin{cases} \frac{1}{2}\binom{n}{n/2} R_{CB}(n/2) R_{CB}(n/2) & if\ n\ is\ even \\ \binom{n}{(n+1)/2} R_{CB}\left(\frac{n+1}{2}\right) R_{CB}\left(\frac{n-1}{2}\right), & if\ n\ is\ odd \end{cases} \tag{7}$$

where $R_{CB}(2) = R_{CB}(1) = 1$.

d. Random-Pair Subset Selection

The utilization of random-pair selection offers a non-deterministic approach to the formation of N_{i1} and N_{i2}, which effectively clusters comparable classes, according to reference [35]. The random-pair selection method involves utilizing the base classifier to directly detect analogous categories within the set of N_i. Initially, a pair of classes, N_1 and N_2, belonging to the set N_i, is randomly chosen, and then a binary classifier is trained exclusively on these two classes. Subsequently, the classifier is applied to the remaining

classes, and the resulting predictions are recorded in the form of a confusion matrix, M. The matrices N_{i1} and N_{i2} are then generated, based on the information contained in M.

$$N_{i1} = \left\{N \in N_i \backslash \{N_1, N_2\} : M_{N,N_1} \leq M_{N,N_2}\right\} \cup \{N_1\}$$
$$N_{i2} = \left\{N \in N_i \backslash \{N_1, N_2\} : M_{N,N_1} > M_{N,N_2}\right\} \cup \{N_2\} \quad (8)$$

where $M_{j,\,i}$ denotes the number of instances belonging to class j that were categorized as class i by the binary classifier. Stated differently, a class is designated as N_{i1} if it exhibits a lower occurrence of confusion with N_1 in comparison to N_2, and as N_{i2} if the opposite is true. Subsequently, the binary classifier undergoes re-training concerning the newly introduced meta-classes N_{i1} and N_{i2}. The suggested approach facilitates the separation of each split by the base learner, as opposed to a fully random split. Additionally, it introduces a level of randomness that results in the creation of varied subsets.

3.3. Software Requirement Risk Prediction Dataset

The dataset used in the experimentation phase of this research is based on risks in software projects and is composed of attributes that are associated with both risks and requirements, as stated in [2]. The dataset is constructed by incorporating risk attributes into the SRS data. Specifically, the project's risk attributes, including the project category, requirement category, risk target category, probability, impact, dimension of risk, and priority of risk, were obtained from sources [9,31,36]. The dataset has 299 instances, and 13 features across five different class labels. Appendix A (Table A1) tabulates the components of the dataset, and further details of the dataset are provided in [1,2,29].

3.4. Experimental Procedure

This section presents and explains the experimental procedure conducted in this research. Specifically, Figure 2 depicts a schematic illustration of the experimental procedures carried out in our research, and highlights the significance of the methodology, which aims to offer empirical justification for the efficacy of the suggested and implemented fuzzy induction models. This research involved the development and analysis of a two-stage experimental design, with a subsequent evaluation of the predictive capabilities of the resulting FURIA-based models, in a fair and unbiased manner.

Initially, the proposed FURIA model and baseline rule-based and ML-based models are implemented on the software risk dataset (See Section 3.3), using the parameter setting as depicted in Table 1. The primary objective of this assessment is to ascertain and authenticate the effectiveness of the FURIA model in software risk prediction in contrast to the baseline rule-based and ML-based models. The investigated rule-based (rough set (RS), partial decision tree (PART), RIPPER) and ML-based (SVM, k nearest neighbour (KNN), DT, average one dependency estimator (A1DE), NB, random forest (RF)) models are selected based on their reported predictive performances in existing software risk prediction studies and other ML processes [7,11,28,29,37,38]. In addition, these models have diverse computational characteristics and are aimed at introducing heterogeneity to empirical experimentation.

Finally, enhanced fuzzy induction methods based on nested subset selection dichotomies are designed and deployed on the experimental dataset. Specifically, the FURIA is further improved via the introduction of nested dichotomy subset selection techniques (further centroid subset selection (FURIA-FCS), balanced subset selection (FURIA-BSS), and random-pair subset selection (FURIA-RPS)), which iteratively deploy binary splits, and function by separating the class label and generating optimal subsets for effective classification. Consequently, the FURIA variant models (FURIA-FCS, FURIA-BSS, and FURIA-RPS) are compared with the baseline FURIA and current rule-based and ML-based models.

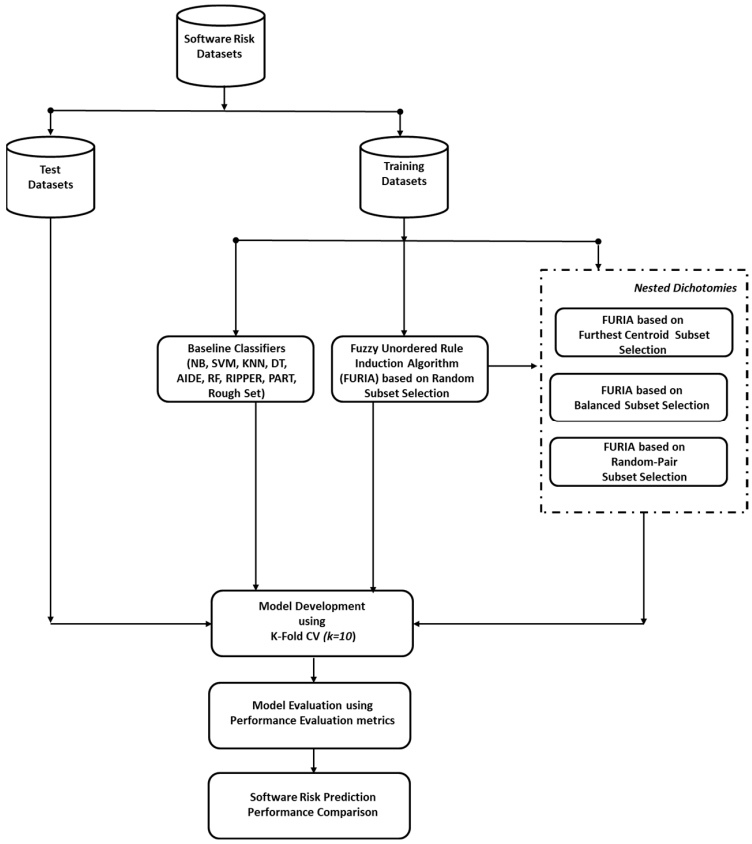

Figure 2. Experimental framework.

Table 1. Parameter settings of the implemented models.

Models	Model Type	Parameter Setting
RS	Rule-based	alphaForPartialReducts = 0.5; discernabilityMethod = OrdinaryDecisionAndInconsistenciesOmitted; discrConfidenceLevelForIntervalDifference = 0.9; reducts = AllLocals; discretization = MaximalDiscernabiliyHeuristicLocal
PART		confidenceFactor = 0.25; useMDLCorrection = True; unpruned = False; NumFolds = 3
RIPPER		optimizations = 2; checkErrorRate = True; usePruning = True; NumFolds = 3
SVM	ML-based	SVMType = C-SVC; KernelType = RadialBasisFunction; loss = 0.1; eps = 0.001; cost = 0.1; nu = 0.5; shrinking = True; gamma = 0.0
A1DE		subsumptionResolution = False; weight = 1.0; weightAODE = False; frequencyLimit = 1
KNN		K = 1; distanceWeighting = False; NearestNeighbourSearchAlgorithm = LinearNNSearch; DistanceFunction = EuclideanDistance
DT		useMDLCorrection = True; collapseTree = True; confidenceFactor = 0.25; subtreeRaising = True;
RF		bagSizePercent = 100; calcOutOfBag = False; numIterations = 100
NB		UseKernelEstimator = False; UseSupervisedDiscretization = False
FURIA	Fuzzy-rule-based	T-Norm = ProductT-Norm; checkErrorRate = True; optimization = 2; uncovAction = ApplyRuleStreching; minFolds = 3;

Analysis of the empirical results obtained from experiments, along with the inferences drawn from these observations, is conducted, to address the research inquiries outlined in Section 1. The cross-validation (CV) technique was utilized in the development of the

software risk prediction models we investigated on the model development technique. The present research employs the *k*-fold cross-validation technique, where *k* is equal to 10, to construct the models used for experimentation. The rationale behind utilizing the CV technique is rooted in its ability to withstand data quality issues that may result in model overfitting, as shown in previous studies [4,32,34]. The training and test datasets were randomly partitioned to mitigate the presence of duplicate values or recurring patterns. To ensure consistency in the performance of the models under investigation, each experiment was conducted ten (10) times, in adherence with the principle of fairness. Ultimately, the mean values of the generated performance metric are utilized to assess the implemented models [39,40]. The experimentation involved the utilization of the WEKA machine learning library [41], and the R programming language [42], on a computer equipped with an Intel(R) Core^{TM} processor, operating at 3.4 gigahertz, and 16 gigabytes of random-access memory. Samples of the experimental dataset, generated models, and solution buffers are available at the GitHub repository (https://github.com/bharlow058/SoftwareRiskPrediction/tree/main, accessed on 29 June 2023).

3.5. Experimental Procedure

This research employed well-established evaluation metrics, including accuracy, f-measure, the area under the curve (AUC), and Matthew's correlation coefficient (MCC), to assess and contrast the predictive capabilities of the various investigated models. The selection of these performance indicators was based on their frequent utilization in prior studies in the assessment of rule-based and ML-based software risk prediction models [43–45]. Furthermore, these metrics are reported to be dependable collectively, as they consider all areas of the confusion matrix produced for each developed model [46,47].

4. Experimental Results and Discussion

This section presents a comparative analysis of the predictive performances of the suggested fuzzy induction models against the corresponding rule-based (RS, PART, and RIPPER) and ML-based (SVM, KNN, NB, DT, RF, and A1DE) classifiers. The analysis is conducted on a benchmark software risk dataset. The primary objective of the comparison is to verify and authenticate the efficacy of the fuzzy induction models in comparison to the established baseline rule, and the ML-based classifiers and current models utilized in prior research.

To enhance the clarity, the analysis of the results is presented and discussed through the utilization of three distinct scenarios. The initial scenario showcases the prediction performances of the FURIA model against the rule- and ML-based models. In the second scenario, a comparative analysis is conducted between the FURIA and its variants (FURIA-FCS, FURIA-BSS, and FURIA-RPS). Lastly, the predictive performance of the suggested fuzzy induction models is compared with the current rule and ML models utilized on the same experimental benchmark dataset.

4.1. Scenario 1: Performance Analysis of the FURIA against the Rule- and ML-Based Models

As shown in Table 2, the predictive performance of the FURIA was compared with individual baseline classifiers such as NB, AIDE, SVM, KNN, DT, RF, RS, PART, and RIPPER, on the experimental dataset. As observed, the FURIA showed the highest accuracy (97.99%), f-measure (0.980), and MCC (0.975) values. Specifically, the FURIA showed +7.71%, +8.11%, +6.16%, +15.03%, +1.73%, and +17.66% increments over the respective accuracy values of NB (90.97%), A1DE (90.64%), SVM (92.30%), KNN (85.19%), DT (96.32%), RF (83.28%) and RIPPER (93.29%), which are ML-based models. A similar occurrence can be observed with other evaluation metrics, such as the f-measure and MCC. The best-performing ML model was DT, and the lowest-performing ML model was KNN. The poor predictive performances of KNN can be attributed to its operational mechanism of instance (lazy) learning, and its dependence on parameter tuning. Nonetheless, the FURIA outperformed the ML-based models on the utilized evaluation metrics.

Table 2. The performance comparison of the FURIA against the rule- and ML-based models.

Models	Accuracy (%)	F-Measure	MCC
NB	90.97	0.911	0.877
A1DE	90.64	0.907	0.872
SVM	92.30	0.923	0.897
KNN	85.19	0.579	0.406
DT	96.32	0.963	0.952
RF	83.28	0.805	0.766
RS	93.31	0.929	0.910
PART	96.65	0.967	0.956
RIPPER	97.32	0.973	0.964
FURIA	97.99	0.980	0.975

Concerning rule-based models, such as PART, RS, and RIPPER, the FURIA showed a comparable predictive performance across the evaluation metrics. Based on accuracy values, the FURIA showed a +5.31%, +1.38%, and +0.7% increment over RS (93.31%), PART (96.65%), and RIPPER (97.32%). Regarding the f-measure and MCC values, similar trends were observed in the predictive performance of the FURIA over the rule-based models.

Although RIPPER's performance was quite comparable to the FURIA, the FURIA still showed higher predictive performances. The relatively high predictive performance of RIPPER may be due to its divide-and-conquer operational mechanism, which makes it easy to use for the classification of data instances. However, the FURIA's use of fuzzy rules over conventional rules, as in the case of RIPPER and RS, makes it better. Moreover, the FURIA can consider an unordered rule set and deploy an effective rule-stretching function. Figure 3 displays a graphical representation of the predictive performances of the FURIA against the investigated rule- and ML-based models on the investigated software risk dataset.

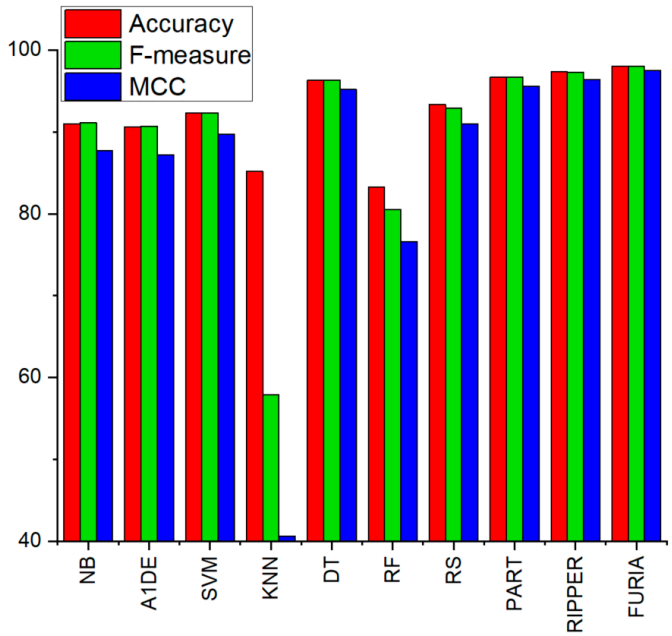

Figure 3. Graphical representation of the FURIA and the baseline rule and ML models.

To improve the predictive performance of the FURIA, nested dichotomy functions were introduced into its working operations. Hence, the following subsection presents a detailed

comparison of the predictive performances of the FURIA, and its nested subset selection dichotomy variants (FURIA-FCS, FURIA-BSS, and FURIA-RPS), on the dataset used.

4.2. Scenario 2: Performance Analysis of the FURIA against Its Nested Subset Selection Dichotomy Variants

Table 3 presents the predictive performance comparison of the FURIA and its nested subset selection dichotomy variants (FURIA-FCS, FURIA-BSS, and FURIA-RPS) on the dataset studied. As indicated, the proposed FURIA variants outperformed the conventional FURIA model. Specifically, FURIA-FCS showed the highest accuracy (98.62%), f-measure (0.987), and MCC (0.983) values at +0.64%, +0.71%, and +0.82% increments over the predictive performance of the FURIA on the same dataset. The duo of FURIA-BSS and FURIA-RPS showed a similar predictive performance, with a 98.33% accuracy value, a 0.983 f-measure value, and a 0.979 MCC value. The differences in the predictive performance of the suggested fuzzy induction models may be insignificant; however, the cost of misprediction or misclassification is significant.

Table 3. The performance comparison of the FURIA against its nested subset selection dichotomy variants.

Model	Accuracy (%)	F-Measure	MCC
FURIA	97.99	0.980	0.975
FURIA-FCS	98.62	0.987	0.983
FURIA-BSS	98.33	0.983	0.979
FURIA-RPS	98.33	0.983	0.979

The superior predictive performances of the nested subset selection dichotomy variants may be attributed to their respective ability to manage hierarchical class relationships. The variants can, for example, handle overlapping class labels, which can be detrimental to the predictive performance of rule- or ML-based models. In addition, the proposed nested subset selection dichotomy variants exhibit incremental learning, where new classes or instances can be added to the existing model without the requirement to completely retrain the entire system. Further, incremental learning in nested dichotomy methods can save computational resources, and allow for efficient adaptation to changing environments. Figure 4 presents the graphical representation of the predictive performances of the FURIA and its nested subset selection dichotomy variants on the investigated software risk dataset.

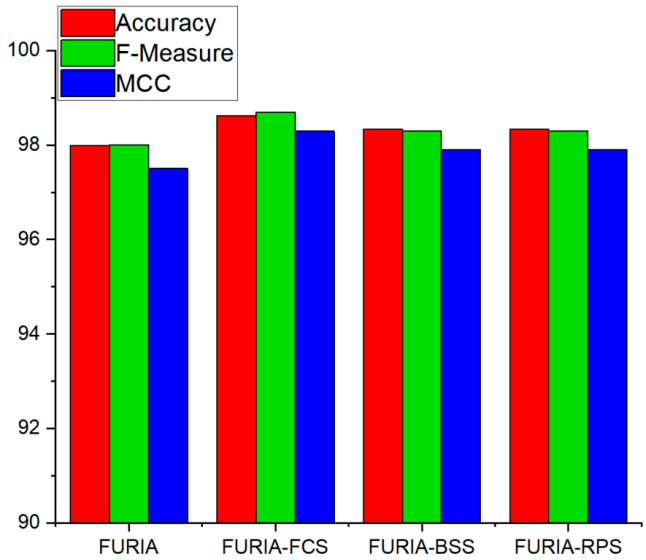

Figure 4. Graphical representation of the FURIA and its nested subset selection dichotomy variants.

For a generalizable predictive performance evaluation, the predictive performances of the suggested fuzzy induction models are contrasted with those of the current software risk prediction models with varied computational characteristics on the same dataset. That is, in the following subsection, the predictive performances of the FURIA, FURIA-FCS, FURIA-BSS, and FURIA-RPS are compared with the current advanced rule- and ML-based models, on the same dataset.

4.3. Scenario 3: Performance Analysis of the FURIA and Its Enhanced Nested Dichotomy Variants against the Existing Rule- and ML-Based Models

The predictive performance of the proposed FURIA, FURIA-FCS, FURIA-BSS, and FURIA-RPS models, with the current advanced rule- and ML-based models, is shown in Table 4.

Table 4. The performance comparison of the FURIA and its enhanced nested dichotomy variants, against the existing rule- and ML-based models.

Model	Accuracy (%)	F-Measure	MCC
FURIA *	97.99	0.980	0.975
FURIA-FCS *	98.62	0.987	0.983
FURIA-BSS *	98.33	0.983	0.979
FURIA-RPS *	98.33	0.983	0.979
[1], DTNB	97.99	0.980	0.959
[1], CDT	97.99	0.980	0.968
[29] NB	98.00	-	-
[11] BNCC	75.15	-	-
[14] ANN	75.00	-	-
[14] SVM	85.00	-	-

The sign (*) indicates the proposed models.

As shown in Table 4, the predictive performances of the suggested fuzzy induction models are compared to current advanced rule- and ML-based models, such as [1,11,14,29], on the same dataset. For instance, Naseem et al. [1] implemented a hybrid method for software risk prediction, based on Decision Tables and Naïve Bayes (DTNB), which recorded an accuracy value of 97.99%, an f-measure value of 0.980, and an MCC value of 0.959. Although the DTNB showed a prediction performance competitive with that of the FURIA model, the nested subset selection dichotomy variants outperformed the DTNB. In addition, Naseem et al. [1] deployed a credal decision tree (CDT) with a comparable predictive performance; still, the suggested fuzzy induction models are better. Hu, Zhang et al. [11] developed an enhanced Bayesian network model with causality constraints (BNCC). The BNCC had a prediction accuracy value of 75.15%, which is still inferior to the accuracy values of the suggested fuzzy induction models. Xu, Zhang, Sun et al. [14] and Akumba et al. [29] used a hyper-parameterized ANN, SVM, and NB, respectively. However, their respective performances are not as good as those of the FURIA model or any of its nested dichotomy variants. Figure 5 presents the graphical representation of the FURIA and current software risk prediction models.

In conclusion, the suggested fuzzy induction models outperformed the current rule- and ML-based models that were investigated using various computational methods on the same experimental dataset. Moreover, the suggested fuzzy induction models can handle datasets with imprecise values and inherent fuzziness. In addition, the suggested fuzzy induction models employed rule-stretching and subset selection methods that helped to reduce rule redundancy in the generated models, which, in turn, improved the efficiency of the fuzzy induction models, by eliminating overlapping or duplicate rules, while preserving the classification accuracy.

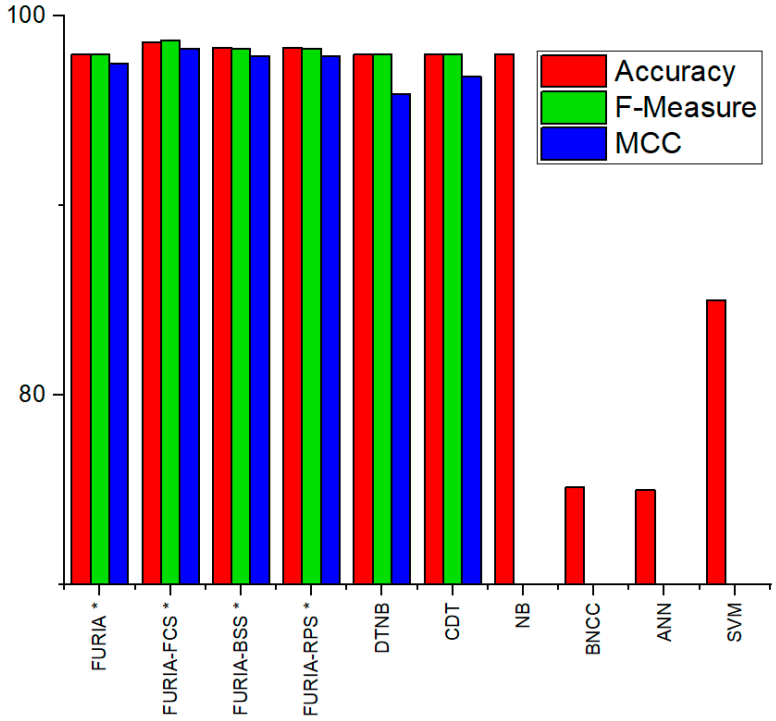

Figure 5. Graphical representation of FURIA and its nested subset selection dichotomy variants against existing current software risk prediction models [1,11,14,29]. The sign (*) indicates the proposed models.

5. Threat to Validity

Assessing and managing potential risks that may affect the dependability of empirical results is an essential aspect of any experimental investigation [48–50]. Through conducting the present research, we have identified several potential threats to its validity, which are outlined below:

External validity: The transferability of scientific investigations to real-world settings is a crucial factor in determining their reliability. The generalizability of experimental insights to other contexts may be limited, due to various factors, such as the characteristics and scope of the datasets utilized. The present research has incorporated a benchmark dataset that exhibits a diverse range of attributes. The public dataset is widely utilized in the advancement and assessment of software risk prediction models, and it is readily accessible to the public without charge. Furthermore, the present research provided a thorough assessment of the experimental methodology, potentially enhancing the consistency and accuracy of its methodological approaches across various software risk datasets.

Internal validity: This paradigm embodies the significance and consistency of the data, tested models, and empirical research. Consequently, this research utilizes established machine learning methodologies that have been previously implemented and utilized in prior research endeavours. The selection of these ML methods was based on their ability to encompass a diverse array of approaches, and their demonstrated efficacy in accomplishing ML objectives. Furthermore, the cross-validation (CV) methodology was employed to train the investigated rule-based and ML-based models on the designated dataset, with great attention to detail. Each trial was repeated 10 times, to ensure its validity. The implementation of this approach has proven effective in mitigating the probability of unforeseen disparities in empirical findings. Conversely, further investigations could explore alternative approaches and techniques for appraising models, in subsequent studies.

Construct validity: This concept pertains to the criteria employed to evaluate the efficacy of the software risk prediction models that were examined. The present research utilized various statistical performance metrics, including the accuracy, f-measure, and

Matthew's correlation coefficient (MCC). These measures facilitated a thorough empirical evaluation of the investigated models utilized in the research. The models investigated were designed with careful consideration of the software risk prediction and its corresponding features.

6. Conclusions and Future Works

This research proposed enhanced Fuzzy Induction Methods for software risk prediction. Specifically, the FURIA and its variants, based on nested subset selection dichotomies, were developed for the software risk prediction process. The enhancement of the predictive capabilities of the FURIA with the nested dichotomies was performed via iteratively deploying binary split functions, to separate the class label, and to generate optimal subsets for effective classification or the prediction process. The viability and effectiveness of the proposed enhanced fuzzy induction models were tested via experiments. The experimental findings observed on the software risk dataset studied indicate the superiority of the fuzzy induction models over the prominent baseline rule- and ML-based models. These findings validate the applicability and efficacy of fuzzy induction models for multi-class software risk prediction processes. In an extended evaluation, the prediction performances of enhanced fuzzy induction models in most cases outperformed the existing rule- and ML-based models from the current research on publicly available software risk prediction. Consequently, this research recommends the use of fuzzy induction-based models in software risk prediction. Aside from the empirically validated predictive performance of the suggested models, fuzzy models allow explainability and are highly scalable and, in most cases, stable, compared to other conventional ML models.

In the next phase of this research, ensemble-based nested dichotomies for software risk prediction and multi-class ML processes will be explored, as ensemble methods are often able to generate more accurate classifiers than individual classifiers. Hence, it is worth investigating the application of ensemble methods to nested dichotomies with varying configurations. Furthermore, the optimization process of the nested dichotomies can be studied more effectively. Regardless, these highlighted areas will be explored in future research studies.

Author Contributions: Conceptualization, H.M., A.O.B. and L.F.C.; Methodology, H.M., A.O.B. and L.F.C.; Software, A.O.B., V.E.A. and G.K.; Validation, V.E.A.; Formal analysis, H.M., A.A.I. and G.K.; Investigation, A.O.B.; Resources, V.E.A. and G.K.; Data curation, V.E.A., A.A.I. and G.K.; Writing—original draft, H.M. and A.O.B.; Writing—review & editing, S.B., L.F.C. and A.A.I.; Visualization, V.E.A., A.A.I. and G.K.; Supervision, S.B. and L.F.C.; Project administration, S.B. and A.A.I. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are openly available in [https://github.com/bharlow058/SoftwareRiskPrediction/tree/main] (accessed on 29 June 2023).

Acknowledgments: This research/paper was fully supported by Universiti Teknologi PETRONAS, under the STIRF Grant Scheme (015LA0-049).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Description of the software requirement risk dataset [1,2].

Features	Datatype	Description
Requirements	String	Text
Project Category	Nominal	Transaction Processing System, Safety Critical System, Enterprise System, Management Information System

Table A1. *Cont.*

Features	Datatype	Description
Requirement Category	Nominal	Functional, Usability, Reliability & Availability, Performance, Security, Supportability, Constraints, Interfaces, Standards, Safety
Risk Target Category	Nominal	Budget, Quality, Schedule, Personal, Performance, Functional Validity, People, Project complexity, Planning & Control, Team, Resource availability, User, Requirement, Time Dimension, Organizational Environment, Cost, Design, Business, Unrealistic Requirements, Overdrawn Budget, Software, Process
Probability	Numeric	0–100%
Magnitude of Risk	Nominal	Negligible, Very Low, Low, Medium, High, Very High, Extreme
Impact	Nominal	high, catastrophic, moderate, Low, insignificant
Dimension of Risk	Numeric	Requirements, User, Project complexity, planning and control, Team, Organizational Environment, Estimations, Software Requirement, Planning and Control, Schedule, Complexity, Project cost, Organizational Requirements
Affecting No Modules	Numeric	Numbers (Whole Numbers)
Fixing Duration	Numeric	Days (Digits)
Priority	Numeric	0–100%
Risk Level	Nominal	1, 2, 3, 4, 5

References

1. Naseem, R.; Shaukat, Z.; Irfan, M.; Shah, M.A.; Ahmad, A.; Muhammad, F.; Glowacz, A.; Dunai, L.; Antonino-Daviu, J.; Sulaiman, A. Empirical assessment of machine learning techniques for software requirements risk prediction. *Electronics* **2021**, *10*, 168. [CrossRef]
2. Shaukat, Z.S.; Naseem, R.; Zubair, M. A dataset for software requirements risk prediction. In Proceedings of the 2018 IEEE International Conference on Computational Science and Engineering (CSE), Bucharest, Romania, 29–31 October 2018; pp. 112–118.
3. Alharbi, I.M.; Alyoubi, A.A.; Altuwairiqi, M.; Ellatif, M.A. Analysis of Risks Assessment in Multi Software Projects Development Environment Using Classification Techniques. In Proceedings of the International Conference on Advanced Machine Learning Technologies and Applications, Cairo, Egypt, 20–22 March 2021; pp. 845–854.
4. Mohamed, H.A.M. Model-Based Prediction of Resource Utilization and Performance Risks. Ph.D. Thesis, Sudan University of Science & Technology, Khartoum, Sudan, 2018.
5. Balogun, A.O.; Almomani, M.; Basri, S.; Almomani, O.; Capretz, L.F.; Khan, A.A.; Gilal, A.R.; Baashar, Y. Towards the sustainability of small and medium software enterprises through the implementation of software process improvement: Empirical investigation. *J. Softw. Evol. Process* **2022**, *34*, e2466. [CrossRef]
6. Salih, H.A.; Ammar, H.H. Model-based resource utilization and performance risk prediction using machine learning Techniques. *JOIV Int. J. Inform. Vis.* **2017**, *1*, 101–109.
7. Hu, Y.; Feng, B.; Mo, X.; Zhang, X.; Ngai, E.; Fan, M.; Liu, M. Cost-sensitive and ensemble-based prediction model for outsourced software project risk prediction. *Decis. Support Syst.* **2015**, *72*, 11–23. [CrossRef]
8. Aslam, A.; Ahmad, N.; Saba, T.; Almazyad, A.S.; Rehman, A.; Anjum, A.; Khan, A. Decision support system for risk assessment and management strategies in distributed software development. *IEEE Access* **2017**, *5*, 20349–20373. [CrossRef]
9. Williams, L. Project risks product-specific risks. *J. Secur. NCSU* **2004**, *1*, 1–22.
10. Du, S.; Keil, M.; Mathiassen, L.; Shen, Y.; Tiwana, A. Attention-shaping tools, expertise, and perceived control in IT project risk assessment. *Decis. Support Syst.* **2007**, *43*, 269–283. [CrossRef]
11. Hu, Y.; Zhang, X.; Ngai, E.; Cai, R.; Liu, M. Software project risk analysis using Bayesian networks with causality constraints. *Decis. Support Syst.* **2013**, *56*, 439–449. [CrossRef]
12. Fan, C.-F.; Yu, Y.-C. BBN-based software project risk management. *J. Syst. Softw.* **2004**, *73*, 193–203. [CrossRef]
13. Neumann, D.E. An enhanced neural network technique for software risk analysis. *IEEE Trans. Softw. Eng.* **2002**, *28*, 904–912. [CrossRef]
14. Hu, Y.; Zhang, X.; Sun, X.; Liu, M.; Du, J. An intelligent model for software project risk prediction. In Proceedings of the 2009 International Conference on Information Management, Innovation Management and Industrial Engineering, Xi'an, China, 26–27 December 2009; pp. 629–632.
15. Bai, C.-G. Bayesian network based software reliability prediction with an operational profile. *J. Syst. Softw.* **2005**, *77*, 103–112. [CrossRef]

16. Lee, E.; Park, Y.; Shin, J.G. Large engineering project risk management using a Bayesian belief network. *Expert Syst. Appl.* **2009**, *36*, 5880–5887. [CrossRef]
17. Khoshgoftaar, T.M.; Allen, E.B.; Hudepohl, J.P.; Aud, S.J. Application of neural networks to software quality modeling of a very large telecommunications system. *IEEE Trans. Neural Netw.* **1997**, *8*, 902–909. [CrossRef]
18. Balogun, A.O.; Basri, S.; Abdulkadir, S.J.; Hashim, A.S. Performance analysis of feature selection methods in software defect prediction: A search method approach. *Appl. Sci.* **2019**, *9*, 2764. [CrossRef]
19. Balogun, A.O.; Basri, S.; Mahamad, S.; Abdulkadir, S.J.; Almomani, M.A.; Adeyemo, V.E.; Al-Tashi, Q.; Mojeed, H.A.; Imam, A.A.; Bajeh, A.O. Impact of feature selection methods on the predictive performance of software defect prediction models: An extensive empirical study. *Symmetry* **2020**, *12*, 1147. [CrossRef]
20. Balogun, A.O.; Basri, S.; Said, J.A.; Adeyemo, V.E.; Imam, A.A.; Bajeh, A.O. Software defect prediction: Analysis of class imbalance and performance stability. *J. Eng. Sci. Technol.* **2019**, *14*, 3294–3308.
21. Kamarudin, N.K.; Firdaus, A.; Zabidi, A.; Ernawan, F.; Hisham, S.I.; Ab Razak, M.F. Android malware detection using PMCC heatmap and Fuzzy Unordered Rule Induction Algorithm (FURIA). *J. Intell. Fuzzy Syst.* **2023**, *44*, 5601–5615. [CrossRef]
22. Průcha, P.; Skrbek, J. Use of FURIA for Improving Task Mining. *Acta Inform. Pragensia* **2022**, *11*, 241–253. [CrossRef]
23. McConnell, S. *Software Project Survival Guide*; Microsoft Press: Redmond, WA, USA, 1998.
24. Boehm, B. Software risk management. In Proceedings of the European Software Engineering Conference, Coventry, UK, 11–15 September 1989; pp. 1–19.
25. Yong, H.; Juhua, C.; Zhenbang, R.; Liu, M.; Kang, X. A neural networks approach for software risk analysis. In Proceedings of the Sixth IEEE International Conference on Data Mining-Workshops (ICDMW'06), Hong Kong, China; 2006; pp. 722–725.
26. Kawamura, T.; Toma, T.; Takano, K.I. Outcome prediction of software projects for information technology vendors. In Proceedings of the 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Singapore, 10–13 December 2017; pp. 1733–1737.
27. Christiansen, T.; Wuttidittachotti, P.; Prakancharoen, S.; Vallipakorn, S.A.-o. Prediction of risk factors of software development project by using multiple logistic regression. *ARPN J. Eng. Appl. Sci.* **2015**, *10*, 1324–1331.
28. Xu, Z.; Yang, B.; Guo, P. Software risk prediction based on the hybrid algorithm of genetic algorithm and decision tree. In Proceedings of the International Conference on Intelligent Computing, Qingdao, China, 21–24 August 2007; pp. 266–274.
29. Akumba, O.B.; Otor, S.U.; Agaji, I.; Akumba, B.T. A Predictive Risk Model for Software Projects' Requirement Gathering Phase. *Int. J. Innov. Sci. Res. Technol.* **2020**, *5*, 231–236. [CrossRef]
30. Akter, S.; Shahriar, H.; Cuzzocrea, A. Autism Disease Detection Using Transfer Learning Techniques: Performance Comparison Between Central Processing Unit vs Graphics Processing Unit Functions for Neural Networks. In Proceedings of the 2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC), Torino, Italy, 26–30 June 2023; pp. 1084–1092.
31. Hühn, J.; Hüllermeier, E. FURIA: An algorithm for unordered fuzzy rule induction. *Data Min. Knowl. Discov.* **2009**, *19*, 293–319. [CrossRef]
32. Mejjaouli, S.; Guizani, S. PDF Malware Detection Based on Fuzzy Unordered Rule Induction Algorithm (FURIA). *Appl. Sci.* **2023**, *13*, 3980. [CrossRef]
33. Zhang, T.; Fu, Q.; Li, C.; Liu, F.; Wang, H.; Han, L.; Quevedo, R.P.; Chen, T.; Lei, N. Modeling landslide susceptibility using data mining techniques of kernel logistic regression, fuzzy unordered rule induction algorithm, SysFor and random forest. *Nat. Hazards* **2022**, *114*, 3327–3358. [CrossRef]
34. Dong, L.; Frank, E.; Kramer, S. Ensembles of balanced nested dichotomies for multi-class problems. In Proceedings of the Knowledge Discovery in Databases: PKDD 2005: 9th European Conference on Principles and Practice of Knowledge Discovery in Databases, Porto, Portugal, 3–7 October 2005; pp. 84–95.
35. Leathart, T.; Pfahringer, B.; Frank, E. Building ensembles of adaptive nested dichotomies with random-pair selection. In Proceedings of the Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, 19–23 September 2016; pp. 179–194.
36. Boehm, B.W. Software risk management: Principles and practices. *IEEE Softw.* **1991**, *8*, 32–41. [CrossRef]
37. Rana, A.; Dumka, A.; Singh, R.; Panda, M.K.; Priyadarshi, N.; Twala, B. Imperative role of machine learning algorithm for detection of Parkinson's disease: Review, challenges and recommendations. *Diagnostics* **2022**, *12*, 2003. [CrossRef] [PubMed]
38. Alarfaj, F.K.; Malik, I.; Khan, H.U.; Almusallam, N.; Ramzan, M.; Ahmed, M. Credit card fraud detection using state-of-the-art machine learning and deep learning algorithms. *IEEE Access* **2022**, *10*, 39700–39715. [CrossRef]
39. Bhatt, C.M.; Patel, P.; Ghetia, T.; Mazzeo, P.L. Effective heart disease prediction using machine learning techniques. *Algorithms* **2023**, *16*, 88. [CrossRef]
40. Shafiezadeh, S.; Duma, G.M.; Mento, G.; Danieli, A.; Antoniazzi, L.; Del Popolo Cristaldi, F.; Bonanni, P.; Testolin, A. Methodological issues in evaluating machine learning models for EEG seizure prediction: Good cross-validation accuracy does not guarantee generalization to new patients. *Appl. Sci.* **2023**, *13*, 4262. [CrossRef]
41. Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten, I.H. The WEKA data mining software: An update. *ACM SIGKDD Explor. Newsl.* **2009**, *11*, 10–18. [CrossRef]
42. Crawley, M.J. *The R Book*; John Wiley & Sons: Hoboken, NJ, USA, 2012.
43. Alkhatib, R.; Sahwan, W.; Alkhatieb, A.; Schütt, B. A Brief Review of Machine Learning Algorithms in Forest Fires Science. *Appl. Sci.* **2023**, *13*, 8275. [CrossRef]

44. Joshi, K.; Bhatt, C.; Shah, K.; Parmar, D.; Corchado, J.M.; Bruno, A.; Mazzeo, P.L. Machine-learning techniques for predicting phishing attacks in blockchain networks: A comparative study. *Algorithms* **2023**, *16*, 366. [CrossRef]
45. Li, S.; Oshnoei, A.; Blaabjerg, F.; Anvari-Moghaddam, A. Hierarchical Control for Microgrids: A Survey on Classical and Machine Learning-Based Methods. *Sustainability* **2023**, *15*, 8952. [CrossRef]
46. Akintola, A.G.; Balogun, A.O.; Capretz, L.F.; Mojeed, H.A.; Basri, S.; Salihu, S.A.; Usman-Hamza, F.E.; Sadiku, P.O.; Balogun, G.B.; Alanamu, Z.O. Empirical Analysis of Forest Penalizing Attribute and Its Enhanced Variations for Android Malware Detection. *Appl. Sci.* **2022**, *12*, 4664. [CrossRef]
47. Balogun, A.O.; Odejide, B.J.; Bajeh, A.O.; Alanamu, Z.O.; Usman-Hamza, F.E.; Adeleke, H.O.; Mabayoje, M.A.; Yusuff, S.R. Empirical Analysis of Data Sampling-Based Ensemble Methods in Software Defect Prediction. In Proceedings of the International Conference on Computational Science and Its Applications, Malaga, Spain, 4–7 July 2022; pp. 363–379.
48. Coleman, P. Validity and reliability within qualitative research for the caring sciences. *Int. J. Caring Sci.* **2022**, *14*, 2041–2045.
49. Flake, J.K.; Davidson, I.J.; Wong, O.; Pek, J. Construct validity and the validity of replication studies: A systematic review. *Am. Psychol.* **2022**, *77*, 576. [CrossRef]
50. Slocum, T.A.; Pinkelman, S.E.; Joslyn, P.R.; Nichols, B. Threats to internal validity in multiple-baseline design variations. *Perspect. Behav. Sci.* **2022**, *45*, 619–638. [CrossRef]