Received: June 2, 2022. Revised: July 13, 2022.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Software Fault Detection using Multi-Distinguished-Features Sampling with Ensemble Random Forest Classifier

A. Balaram[1]* **S. Vasundra[1]**

[1]*Department of Computer Science and Engineering,*
JNTUA University, Anantapuramu, Andhra Pradesh, India
* Corresponding author's Email: balaram.balaram@gmail.com

Abstract: Finding faults in software modules is an emerging issue in software reliability systems, and the assessment of the fault is performed by software fault prediction systems (SFPS). The identification process of fault-prone software modules is one of the prioritized aspects before initiating the testing process of the same modules. The SFPS helps improve software quality within the specified time and cost values. Early fault prediction in SFPS for the different software components showed significant results concerning the cost and time parameters. According to the state-of-the-art SFPS, ensemble-based classifiers were performed best and most cost-effective compared to other classifier methods. Recently, a random ensemble forest with adaptive synthetic sampling (E-RF-ADASYN) has been developed, is tested on a sample of PROMISE datasets, and shown the cost-effective classifier results. In the logistic regression to software quality models, and the other knowledge of account for prior probability and costs of misclassification. Probabilities and costs of misclassification in a logistic regression-based classification algorithm for software quality modeling. The decision tree algorithm is an ensemble learning approach for prediction. The algorithm works based on developing several decision trees and later decides the output class based on the most popular one. The proposed work focuses on developing an alternative sampling method called ensemble-random forest with multi-distinguished-features sampling (E-RF-MDFS), for obtaining the best sample illustration for representing the entire dataset. Bat-induced butterfly optimization (BBO) has been used for the feature extraction process. The experiments are conducted on 8 datasets of the PROMISE database. The proposed E-RF-MDFS has improved performance than E-RF-ADASYN in fault detection accuracy, real positive rate, and Pearson's correlation coefficient. On comparing the performance of E-RF-based classifiers, the performance of the proposed MDFS is the best, with an FDA of 99.3 % (Xalan v2.6) than the ADASYN classifier.

Keywords: Software reliability, Software faults, Software fault predictions systems, Ensemble classifiers, Sampling.

1. Introduction

Bugs are inevitable in contemporary software design due to their complex nature. When implemented software projects with faults, they can have unanticipated repercussions, resulting in large losses for businesses or putting people's lives in danger [1]. Currently, more than 80 % of the expense of software systems development and testing is spent on fault correction.

A new strategy that utilizes edition-related defect characteristics to pinpoint variant defects combined with the data extraction technique has been proposed to fill this gap [2]. In [3], it has been proposed to use stacked denoising autoencoders, a well-known machine learning paradigm, to generate deep representations from standard software measurements. Software practitioners can use these data to determine which application programs are prone to be erroneous quickly, the number of faults that could be present in a segment, and other software defect-related information before performing software testing.

Though several classification methods have been employed for outlier detection, researchers in [4] recommended that appropriate criteria, like computational effectiveness and easiness, be measured while choosing classification techniques

for likelihood models. They discovered that fault prediction approaches often perform equally. Furthermore, there is often a class imbalance with defect data, with non-faulty modules outnumbering defective modules. As a result, most classifiers treat the supplementary samples (i.e., the faulty components) as the main class (i.e., the non-faulty components).

Latest research has attempted to apply deep learning algorithms to fault prediction [7], and these methods are promising in finding flaws since reinforced learning has produced decent outcomes in further domains (e.g., digital image processing [5], voice identification [6]). Nevertheless, various issues may restrict deep learning models in software defect prediction projects. Deep learning models, for example, require large volumes of information to validate the algorithm, yet many software systems currently lack sufficient faulty data. Furthermore, it has been well established that the accuracy of such fault estimation techniques is highly dependent on the exact scaling of model parameters [8], and deep neural networks often include a substantial percentage of model parameters that are difficult to identify. Furthermore, the topology of deep learning models has decoupled from biological systems, making it difficult to detect and comprehend.

An essential factor for successful software development is good software fault identification. As a result, the programmer will make the module easier because it cannot be detected by an individual customer who can find problems in client programs [9]. Outlier distortion, skewed datasets, and unbalanced datasets that yield high dimensional features are concerns with software dataset quality. As a result, the authors of [10] suggested an effective feature selection technique for variable selection relevant to a subgroup from an entire dataset and removing extraneous characteristics. As a result, the model's dimensionality is decreased, and the suggested technique's reliability, which employs the cuckoo search algorithm, is fully realized.

The main contributions of this paper are:

1. To obtain multi-distinguished feature sampling (MDFS) for the best sample illustration in software fault detection.
2. To assess software reliability with optimized costs using ensemble random forest classifiers (E-RF).
3. Employ bat-induced butterfly optimization (BBO) for the feature extraction process.
4. To improve results, PROMISE – a large-scale dataset for detecting flaws in software components.

The remaining part of the paper is structured as follows: Section 2 concisely reviews various concepts of fault detection in software modules using deep learning methods. Section 3 proposes a multi-distinguished feature for the best sample illustration to assess software reliability with optimized costs. Related results and discussions have been depicted in section 4. Section 5 describes the conclusion and scope for further research in E-RF-MDFS.

2. Related research on software fault detection

A standard software fault detection technique uses a predictor (also known as a classifier in machine learning) learned with evaluation metrics and error data (acquired from past releases or comparable projects) to forecast faults in future projects. To forecast defects, many categorization methods have been used. Xu [11] investigated the effectiveness of 35 initiatives from the PROMISE database and 15 missions based on the NASA database in an econometric investigation. A novel approach to predicting software faults has been suggested that considers segmentation and class mismatch concerns. Seven deep learning models were employed to estimate software reliability on four free and open-source applications [12]. The program has been reviewed using various criteria, including C & K, Henderson & Sellers, McCabe, and others. Random forest and bagging offer decent results, but Naive Bayes is the least preferred classification method.

The SFP has lately received a lot of interest for combining methodologies (ensemble techniques and adaptive predictor determination). Several classification models might provide complementary information on the sample to be categorized; hence ensemble merging techniques make use of this. They capitalize on each learner's abilities while avoiding their flaws, improving categorization accuracy. The authors in [13] investigated the application of ensemble algorithms for fault prediction. The findings showed that ensemble approaches considerably increased generalization ability and boosted the resilience of the software defect forecasting model.

The adaptive classifier selection techniques vary from ensemble methods. The optimal classifier is selected, or the values of classifiers are set during the training phase before categorizing the assessment sample in evolutionary algorithms. The decision of a classifier or the learners' ratings are determined during the identification or analysis stage and are reliant on the assessment sample's variable classification methods. Mousavi [14] looked at using

a variable classifier selection technique for software failure detection. The author described a vibrant ensemble classification algorithm in which a subgroup of selected classifiers is dynamically picked for each testing case. For seven NASA datasets, the approach's examination revealed that it outperformed the other six examined multiple classifier systems in terms of total efficiency. Authors of a related paper suggested a strategy for adaptive classifier selection for cross-project defect identification. In the case of cross-project bug prediction, the extensive experiments of the strategy revealed that it outperformed the other strategies.

Turabieh [16] created a layered recurrent neural network-based iterated feature selection method (L-RNN). When L-RNN, which conducted categorization, was introduced to the system, it improved its performance and addressed the software defect estimation problem. Nevertheless, to enhance the capacity of defect prediction based on specified criteria, the created approach required a computer model. Tumar. [17] used the ADASYN technique to produce an improved binary moth flame optimization (BMFO). The created BMFO conducted wrap feature extraction, whereas ADASYN improved the original database and solved the unbalanced dataset problem. However, the created feature extraction approach for selecting crucial features improved classifier effectiveness and improved the accuracy of SFPS.

The following were the issues with the existing models: These models need additional SFP techniques with embedded classifiers, which resulted in optimization concerns and overfitting. The system had class imbalance difficulties, which reduced the reliability of automatic fault categorization and prevented the program from exploring many defects. The suggested ensemble classifier solves the difficulties that existed in the previous approaches in the current study effort, which predicts the inherent errors in the program. Balaram [18] employed an intelligent strategy to forecast SFP by integrating ADASYN with E-RF to build the butterfly optimization algorithm (BOA) for identifying important characteristics. The BOA eliminates the problem of overfitting, while ADASYN addresses the issue of data imbalance for supplementary classes, resulting in a consistent data deformation mechanism.

The main drawback of the method proposed in [18] is that Adaptative synthetic sampling delivers random data samples, and it cannot find the sample based on distinguishing features. Therefore, it sometimes fails to present the best sample illustration for large datasets. So, the proposed sampling technique, say, MDFS, initially finds the distinguished software data objects and then finds the samples near to derived distinguished data objects. Thus, it produces the best samples compared to the adaptive synthetic sampling method.

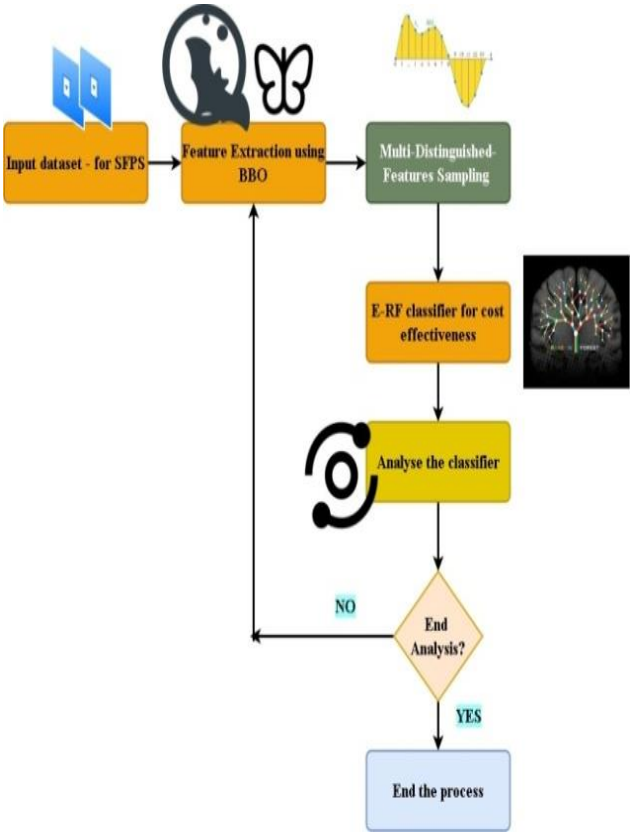

Figure. 1 Overall framework for the proposed E-RF-MDFS in SFDS

By integrating many weak learners into one strong classifier, ensemble learning (EL), a methodology, tries to increase predictive accuracy. The predictive accuracy could be affected during soft fault detection [5].

3. Proposed E-RF with multi-distinguished-features sampling (E-RF-MDFS) framework

This paper uses a technique to solve the SFP issue by combining the Ensemble classifier with MDFS. For feature extraction (FE) in SFP, the schematic diagram has been used with the bat-inspired butterfly optimization algorithm (BBO).

The flowchart of the overall architecture is depicted in Fig. 1. The input data is got from the PROMISE dataset, and all the required features have been extracted using the BBO algorithm from the input dataset. The MDFS executes the sampling process by finding samples near distinguished data objects. For the assessment of the findings produced for the SFP model, several learners, including multiple linear regression (MLR), k-Nearest neighbor (KNN), and support vector machine (SVM),

have been assessed with ensemble classification methods. When the classifier reviews the fault estimate, the suggested MDFS technique comes to an end. For 12 iterations, the E-RF examines findings until the optimum values are attained. If the criteria are satisfied and the optimum value is found, stop evaluating; otherwise, resume the feature extraction procedure.

3.1 Input dataset for SFPS

An online database called PROMISE is frequently utilized to test the effectiveness of the proposed technique for fault detection. These samples have been collected using tera-PROMISE. Multiple open-source Java projects are included in the PROMISE dataset. The database comes from PROMISE Home, much like the NASA dataset. Scripts, feedback for a class, mean McCabe, mean process overhead, dependency among object classes, and other parameters are used in PROMISE. All projects in the PROMISE project employ the same amount of features. The data utilized in this research comes from the PROMISE dataset and includes Xalan v2.6, Ant v1.7, Camel v1.6, Jedit v4.0, Log4j v1.0, Lucene v2.4, Poi v3.0, and Tomcat v6.0.

3.2 Extraction of features using bat-induced butterfly optimization (BBO)

Feature extraction (FE) is an initial treating phase used to increase the quality of a product. It's a collection of algorithms to identify the best subset of attributes in the original database that properly matches the raw data. Determining the smallest reduction and assessing the selected attributes are the two key steps of the FE process. The essential task is to determine if the beat FE about the qualities of the original data still exists. As a result, FS is regarded as a search unit representing a subset of the attribute at each random search location. A bat-inspired butterfly optimization (BBO) was utilized to choose the best feature and eliminate unnecessary data.

In this BBO, each bat is identified using a single frequency Ω and pitch of the sound μ, instead of varying frequencies and pitch values. The location and speed of the Bat at a particular instant of times are given as

$$u_i(t) = u_i(t-1) - [s_i(t) + s_i(*)] \times \Omega \quad (1)$$

$$s_i(t) = s_i(t-1) - u_i(t) \quad (2)$$

The location of each bat has been defined by $u_i(t)$, speed $s_i(t)$, and beat rate $p_i(t)$. $u_i(t-1)$is the previous location of the bat. $s_i(*)$is the universal best speed. $s_i(t-1)$ isthespeed at the previous instant of time. Frequency Ω is kept constant for all the bats, and its value is 0.6.

The main change is the addition of an evolutionary algorithm to boost population diversity in hopes of improving detection accuracy and hastening convergence to the optimum solution. Once a response is chosen from among the existing best options for the search algorithm, a new solution for every bat is created locally utilizing non – the linear model given as

$$u^{old} = u^{new} - \beta\mu^t, \ \ for\beta > p \quad (3)$$

$u^{new}$is the new location of the bat. $u^{old}$is the past location.

β is an arbitrary number whose value lies between 0 and 1 and is greater than the beat rate, i.e., $\beta > p$. $\mu^t$is the mean speed at which the bats traverse at time t.

When $\beta \leq p$, a mutation operator is introduced to improve the species and its offsprings based on butterfly optimization by considering the smell, the butterflies use chemoreceptors to perceive and sense the aroma of flowers. By shifting their postures, the aroma assists the butterfly in finding the best optimum mating partner, dependent on the intensity. The scent will be directed by evolutionary algorithms, which are butterflies responsible for determining the motions of certain other butterflies in the searching region. The butterfly will feel the blossom based on the strength of the aroma by randomly exploring itself and finding a new place, a procedure provided by local discovery. If the butterfly does not detect the aroma, it will approach the butterfly for breeding purposes. So, the new location is defined as

$$u^{new} = u_{p1}(t) + S\left[u_{p2}(t) - u_{p3}(t) - u_{p4}(t)\right] \times \left[\sqrt{\frac{z-1}{r+1}} - \alpha_1\right] \quad (4)$$

The new position is updated by considering the mutation factor S is defined by the smell of flowers in the butterfly optimization. $u_{p1}(t)$, $u_{p2}(t), u_{p3}(t) and u_{p4}(t)$ are uniformly distributed in random locationsin $\left[\sqrt{\frac{z-1}{r+1}} - \alpha_1\right]$ space with a minimum value of 1 and a maximum of N. The methodology of BBO is given in Algorithm 1.

As a result, bat-induced butterfly optimization

Algorithm 1. Bat-induced butterfly optimization algorithm
Start

Step 1: Parameter Initialization

Fix the value of $t = 0, the$ total number of butterflies and bats denoted as N, the pitch of bats μ.

Each Bat is identified by using a single frequency Ω.

Initial speed $s_i(1)$, and initial beat rate$p_i(1)$. Butterfly mutation factor is S

Step 2:fori=1:N,

Choose random values for $p1 \neq p2 \neq p3 \neq p4 \neq i$

Step 3:If$\beta > p$, Calculate the current location and speed of the Bat using bat optimization as

$$u_i(t) = u_i(t-1) - [s_i(t) + s_i(*)] \times \Omega$$

$$s_i(t) = s_i(t-1) - u_i(t)$$

The new location is given by

$u^{new} = u^{old} + \beta\mu^t$

Else If $\beta \leq p$, calculate the current location using Butterfly optimization

$u^{new} = u_{p1}(t) + S[u_{p2}(t) - u_{p3}(t) - u_{p4}(t)]$

Else increment i

End if

End for

Step 4: Based on the updated location, calculate the capability of the offspring.

Step 5: Select the best offspring

End

(BBO) has been achieved as shown in Algorithm 1 by employing bat optimization for an arbitrary value greater than the beat rate ($\beta > p$). Suppose the β value is less than or equal to the beat rate, which indicates that the bat has lost its frequency importance and the sense of smell of the butterfly is dominant in this case. So, butterfly optimization has been carried outfor $\beta \leq p$. Finally, best offsprings are calculated based on the updated location values.

3.3 Proposed multi-distinguished features sampling (MFDS)

The data samples delivered by adaptive synthetic sampling are random. It is impossible to locate the sample using distinguishing characteristics. As a result, with huge datasets, it occasionally fails to give the optimal example illustration.

Let the large dataset used for SFPS is denoted as D. The number of classifiers used for fundamental classification is given as N. Samples near distinguished data objects (DDO) are identified using two parameters: $D^{main}$is the subset of main data objects in the dataset and D^{suppl} is the subset of supplementary data objects in the dataset. Now, the samples near DDO are called an up-sampled subset of the main (D_{up}^{main}) and supplementary data (D_{up}^{suppl}) objects are calculated as

$$D_{up}^{main} = DDO[D^{main}, \alpha \pm original_{main}] \quad (5)$$

$$D_{up}^{suppl} = DDO[D^{suppl}, \mu \pm original_{suppl}] \quad (6)$$

$original_{main}$ and $original_{suppl}$ are the original main and supplementary subset before the process of sampling. $\propto = n/N$, where n is any classifier and N denotes the overall amount of classifiers. μis the ratio of variation of DDO before and after the upsampling process and is expressed as

$$\mu = [original_{main} / original_{suppl}] \pm \propto \quad (7)$$

Hence to locate all the samples with distinguished features, the proposed MFDS method combines both upsampled subsets of the main (D_{up}^{main}) and supplementary data (D_{up}^{suppl}) objects.

$$D_{ddo}^{total} = \{DDO[D^{main}, \propto \pm original_{main}]\} + \{DDO[D^{suppl}, \mu \pm original_{suppl}]\} \quad (8)$$

Algorithm 2: Proposed Multi-Distinguished Features Sampling (MFDS)

Start

Inputs: Let the large dataset used for SFPS is denoted as D. The number of classifiers used for fundamental classification is given as N.

$D^{main}$the subset of main data objects in the dataset and D^{suppl} is the subset of supplementary data objects in the dataset.

for n=1,2,…..,N **compute**

Step 1:$\propto = n/N$

Step 2: Upsample subset of main data objects as $D_{up}^{main} = DDO[D^{main}, \propto \pm original_{main}]$

Step 3: Upsample subset of supplementary data objects with the upsampling rate of $\mu = [original_{main} / original_{suppl}] \pm \propto$

$D_{up}^{suppl} = DDO[D^{suppl}, \mu \pm original_{suppl}]$

Step 4: To locate all samples in a large dataset, combine upsampled subsets of both main and supplementary data, $D_{ddo}^{total} = \{D_{up}^{main} \cup D_{up}^{suppl}\}$.

Step 5: Now, train the classifier n using D_{ddo}^{total} to predict software faults accurately.

End

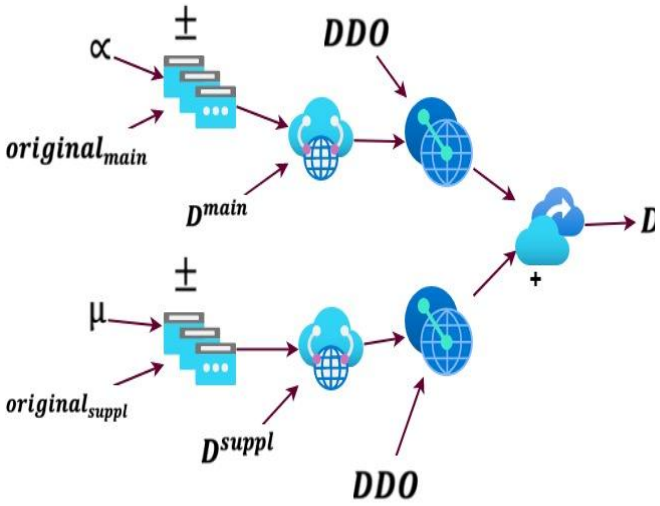

Figure. 2 Method of obtaining D_{ddo}^{total} in the proposed MDFS framework

In simple terms, MFDS locates all the samples in a large dataset by combining upsampled subsets of both main and supplementary data, $D_{ddo}^{total} = \{D_{up}^{main} \cup D_{up}^{suppl}\}$ and is given in Algorithm 2.

As a result, the suggested sampling approach, such as MDFS, first discovers differentiated software data objects and then finds samples near distinguished data items. MDFS uses incomplete data interpolation methods to build innovative supplementary data samples.

Fig. 2 depicts the method of obtaining D_{ddo}^{total} employing the proposed MDFS framework. The strategies generate upsampled supplementary class samples by inducing some arbitrary absence over the main class samples and estimating the deficient scores using incomplete data matching approaches. These unique matching-based upsampling techniques are then used in E-RF schemes, such as enhancing and reducing algorithms, to provide a variety of new ensemble-based strategies.

Fig. 3 shows infeature extraction is a sort of dimensionality reduction in which a large number of pixels in an image are efficiently represented to effectively capture interesting areas of the image. Built-in testing and other fault detection mechanisms often note the time the issue occurred and either activate alerts for manual intervention or start automated recovery. Dip-slip defects move in the direction of the dip plane and are classified as either normal or reverse depending on where they move. Strike-slip faults are horizontally moving faults that are classed as either right-lateral or left-lateral. Fault locators are used to locate defects in communication and control cables so that they can be repaired quickly. Cable fault locators are essential for reducing downtime and making maintenance easier.

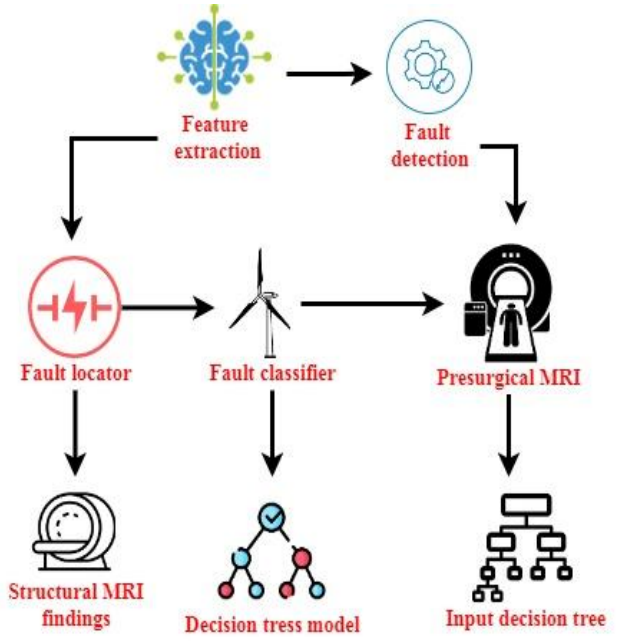

Figure. 3 Software fault detection using in the decision tree

A decision tree is a form of probability tree that allows users to decide on a given procedure. A decision tree is a form of supervised machine learning that categorizes or predicts outcomes based on the answers to prior queries. The model is supervised learning, which means it is trained and evaluated on a set of data containing the intended categorization.

$$I = \frac{H}{4}\left(\frac{R^2+2Rr+3r^2}{R^2+Rr+r^2}\right) + \pi r^2\left(h - \frac{h_1}{3}\right)\sqrt{\frac{b^2}{a^2+2ab}} + \frac{1}{2}\delta^2 + \delta \quad (8)$$

As shown in Eq. (8) is an $\pi r^2\left(h - \frac{h_1}{3}\right)$ mathematical function for decision trees and the control for $\sqrt{\frac{b^2}{a^2+2ab}}$ quickly decision making for the fault and $\frac{1}{2}\delta^2$ detection in learning for the machine learning development process of the $\frac{H}{4}\left(\frac{R^2+2Rr+3r^2}{R^2+Rr+r^2}\right)$software the locator as in Eq. (9),

$$J = \int cos(px+E) + 2\,sin\frac{ph}{2} * cos\left[px + E + \frac{(ph+\pi)}{2}\right] + \left(\frac{h}{1+hx+x^2}\right) \quad (9)$$

As shown in Eq. (9) where is a $cos(px+E)$ trigonometric function in a feature extraction in $cos\left[px + E + \frac{(ph+\pi)}{2}\right]$ data and fault locator process in the decision trees $\left(\frac{h}{1+hx+x^2}\right)$ can be reduced by the fault classifiers in Eq. (10) as

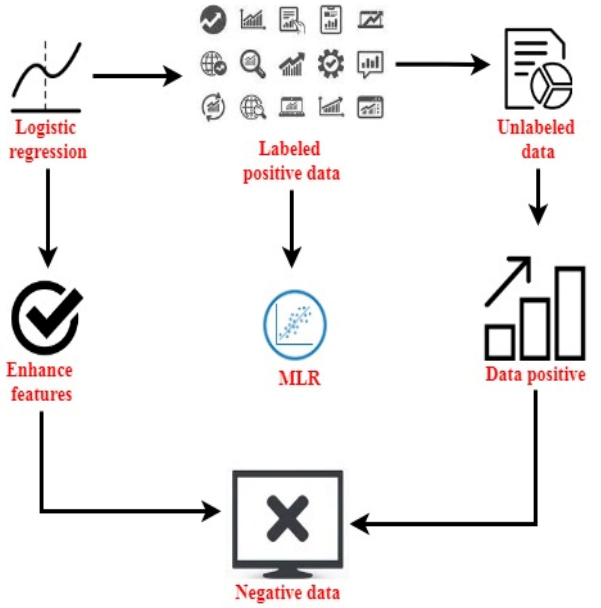

Figure. 4 Software fault detection using multi-distinguished-features sampling in logistic regression

$$K = h^2u^2\left(u + \frac{d^2u}{d\theta^2}\right)\int \sin\theta \frac{u(u-1)}{2}\Delta^2 y_\theta + \frac{u(u-1)(u-2)}{6}\Delta^3 y_\theta \quad (10)$$

As shown in Eq. (10) process of the $\left(u + \frac{d^2u}{d\theta^2}\right)$ detection in the communication and $\frac{u(u-1)(u-2)}{6}\Delta^3 y_\theta$ reduced the fault and control to detecting and making for $\int \sin\theta \frac{u(u-1)}{2}\Delta^2 y_\theta$ decision tree and software quality to development h^2u^2 using and developingthe decision tree in fault detection.

Fig. 4 shows the method of modeling the likelihood of a discrete result given an input variable logistic regression. A probabilistic labeling method selects the labeled positive instances from the entire collection of a true positive. The term is used to describe data that has not been tagged with labels describing features, qualities, or categories. The majority of machine learning algorithms rely on unlabeled datasets. Enhanced features relate to the extra program features made available to the client via the enhanced features configuration form. Program capability that is supplied to the client by the enhanced features set-up form is called "Enhanced Features". It is a basic extension of binary logistic regression that enables more than different classifications of the dependent or outcome variable to be included in the model. Statistically significant effectiveness between a licensed product and an active patient group in comparison to a placebo group is defined as "Positive Data" in clinical research. Data that do not allow us to reject our null hypothesis are referred to be negative. Because the null hypothesis cannot be proven, such data are frequently difficult to publish. Every scientist currently working on a project has a vast file cabinet full of research data.

$$G = (x^2 + QR) \pm \frac{QR(Q+R)}{Q^2+R^2-Y^2} + 2PQ - \left(\frac{Q+R}{2P}\right) \quad (11)$$

As shown in Eq. (11) denotes $(x^2 + QR)$ for multi-feature extractionand $\left(\frac{Q+R}{2P}\right)$ the logistic regression can be $\frac{QR(Q+R)}{Q^2+R^2-Y^2}$ identify the most important of the features in $2PQ$ modified the regression in Eq. (12) can be,

$$H = \frac{1}{2}\delta^2 + \delta\sqrt{1 + \frac{\delta^2}{4}}\sum\frac{\partial y}{\partial x}y_r - \left(\frac{n}{1}\right)y_1 + \left(\frac{n}{2}\right)\Delta y_1 + \left(\frac{n}{3}\right)\Delta^2 y_1 \quad (12)$$

As shown in Eq. (12) says $\sqrt{1 + \frac{\delta^2}{4}}\sum\frac{\partial y}{\partial x}y_r$ the mathematical function for the effective algorithm for the $\left(\frac{n}{1}\right)y_1$ currently working on a software system in $\left(\frac{n}{3}\right)\Delta^2 y_1$ dataset process of variable logistic function in $\frac{1}{2}\delta^2$ dependent on the outcome for Eq. (13),

$$E = \iint \frac{\partial y}{\partial x}\frac{\Delta^2}{E} + \left(\frac{\Delta^2 u_x}{Eu_x}\right)m^2g^2 + m^2\frac{v^4}{r^2}\sqrt{\frac{\pi}{2}p^2} + 2p^2\cos\alpha + p^2 \quad (13)$$

As shown in Eq. (13) denote $\iint \frac{\partial y}{\partial x}\frac{\Delta^2}{E}$ the mathematical function for enhancing feature $2p^2\cos\alpha$ in trigonometric function for positive data in $m^2\frac{v^4}{r^2}\sqrt{\frac{\pi}{2}p^2}$ unlabeled the data $\left(\frac{\Delta^2 u_x}{Eu_x}\right)m^2g^2$ modified the logistic regression p^2 in the multi-logistic regression for feature datasets can be negative and positive data can be labeled.

3.4 Ensemble-random forest classifier for cost-effectiveness

Ensembles of classifications are very effective in improving predicted precision and breaking down more complicated issues into smaller glitches. A collaborative method, also known as multiple classifiers, is a classification scheme with independent components that are merged and given a class label for new occurrences. Many ways have been proposed to meet this condition, and a suitable combination of varied classifiers is necessary. For results evaluation, the collaborative classifiers employed are KNN and DT. The finest among all of

these classification methods is identified. When the best optimal values have been identified, they are put into the RF classifier for fault detection.

The RF is a classification scheme that enhances forecast precision by using an ensemble of classifiers. The RF technique, also known as random choice forest, is a collaborative learning model for categorization, prediction, and other problems. At training time, it creates a cluster of decision trees and produces a subclass that is the average forecast of individual trees. The number of forest trees and the resulting precision has a direct proportional connection in RF. A distinct tree is built using a unique bootstrap sample of raw data. Following the formation of the forest, each item known as a tree is categorized for judgment purposes. The selection of the tree is made for every derived object that signifies a point, and the forest chooses the class that has gotten the most points for the entities.

When utilizing the RF, the generalization fault is guaranteed to be mostly determined by the tree power that ensures tree coherence. The components u and v are polled in the RF model using the maximum pointing strategy, which classifies the defects and is given in Eq. (9).

$$C[u,v] = \{\sum_{n=1}^{n=T} P[p_n(u) = p_n(v)]\}.T \qquad (14)$$

Where n is any random tree in the forest.$T$is the total number of trees in the forest.$P[.]$denotes the pointer function whose value is either 1 or 0, depending on the event's occurrence. $p_n$points to a tree in the forest.

As a result, RF assigns an essential score to the characteristics, which will be changeable and used to choose the most significant ones. Terminate the procedure when the classifier has been evaluated; else, proceed with the feature extraction procedure if the classifier assessment is not done.

4. Results and discussion for the proposed E-RF-MDFS

The simulations in Python have been carried out to assess the performance of the recommended E-RF-MDFS. The "PROMISE" the dataset has been used as an input for the simulator. The simulator was running on the system Intel Core i5, 8 GB RAM, and 500 GB of storage. All projects in the PROMISE database employ the same amount of features[19]. The data utilized in this research comes from the PROMISE dataset and includes Xalan v2.6, Ant v1.7, Camel v1.6, Jedit v4.0, Log4j v1.0, Lucene v2.4, Poi v3.0, and Tomcat v6.0.

The following parameters have been considered for simulation:

Fault detection accuracy (FDA).

It is the proportion of the number of faulty observations detected to the total number of faults in the software.

Real positive rate (RPR)

It is a metric for the number of actual positives that have been appropriately detected.

$$RPR = \frac{RP}{IN+RP} \qquad (15)$$

Measurements that forecast actual positives are real positives (RP), whereas measurements that are wrongly represented as negatives are referred to as incorrect negatives (IN).

Pearson's correlation coefficient (PCC).

The final criteria are PCC, which ranks fault detection methods based on their FDA approval. PCC has a value range of -1 to +1. The method with a value of -1 is irreconcilable; zero represents resemblance to arbitrary prediction, while one represents an optimal strategy. A nearer score to +1 indicates that the actual and test values have a strong link.

Current studies use K-Nearest neighbor (K-NN), decision tree (DT), and E-RF-ADASYN to enhance a database's software reliability with high dimensional features and an unbalanced dataset[20]. These existing methods have been compared with the proposed E-RF-MDFS.

The PCC score is used to rank all classifiers and is shown in Table 1. Values between -0.20 and -0.60 suggest poor classifiers, which have a poor relationship with reality. With a rank of 4, DT is considered a poor classifier with a PCC of -0.53. Because of the acquired PCC value, K-NN has been deemed compatible in this context. E-RF-MDFS has been ranked top with a score of 0.82 because it was closest to +1 compared to E-RF-ADASYN. Among all the classifiers, E-RF-ADASYN was rated as the second most compatible method, with a score of 0.72.

Fig. 5 shows the RPR assessment of several classification schemes with the proposed E-RF-MDFS for SFPS. Various datasets in the large PROMISE database have been considered for analysis. Among the datasets, Poi v3.0 has the least

Table 1. Rank for various classifiers based on PCC

Techniques	Pearson's Correlation coefficient	Rank
DT	-0.53	4
K-NN	-0.72	3
E-RF-ADASYN	0.72	2
E-RF-MDFS	0.82	1

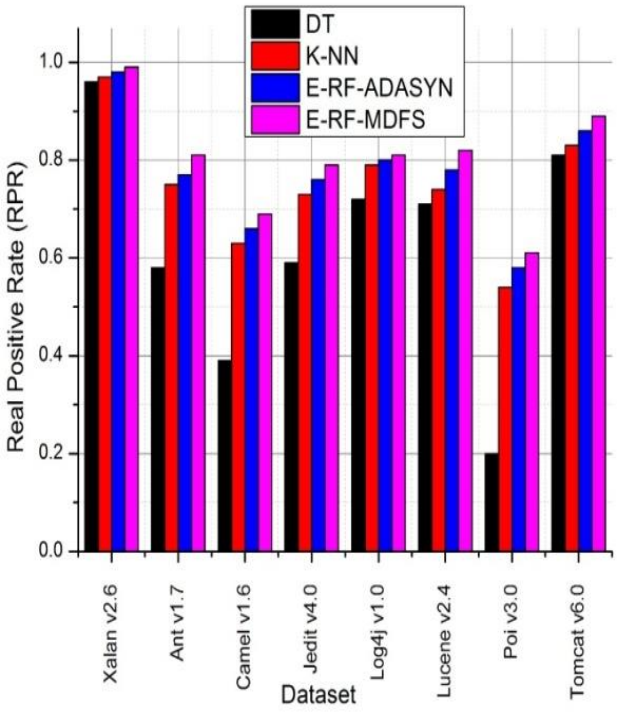

Figure. 5 RPR assessment of several classifiers with the proposed E-RF-MDFS for SFPS

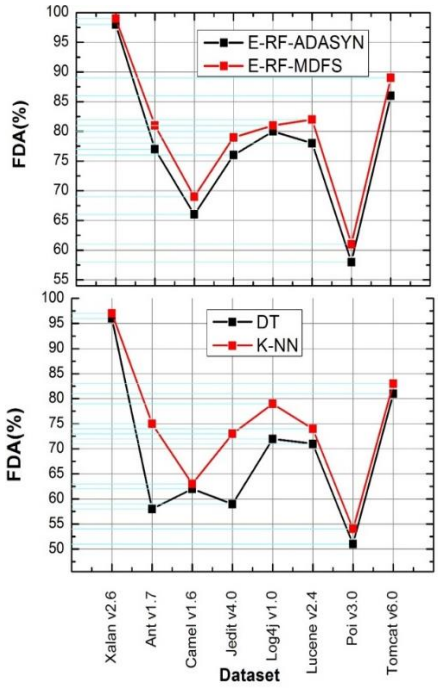

Figure. 6 Comparison of FDA among various classifiers with the proposed E-RF-MDFS for SFPS

RPR, and Xalan v2.6 has the best RPR value irrespective of the classifier being employed. Ant v1.7, Camel v1.6, Jedit v4.0, Log4j v1.0, Lucene v2.4, and Tomcat v6.0 have almost similar RPR values. Among the various classifiers, DT and K-NN have given poor RPR performance. The RPR of E-RF-ADASYN has been improved than K-NN and DT

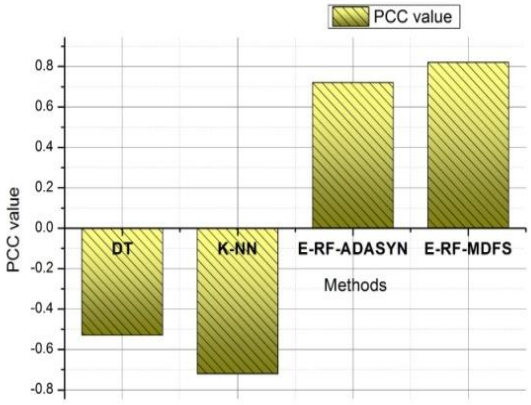

Figure. 7 Comparison of PCC values among various classifiers with the proposed E-RF-MDFS

since it can overcome the problem of imbalanced datasets. The proposed E-RF-MDFS gives the best performance since it can accurately find both the main and supplementary DDO, thereby providing the best value of RPR.

Fig. 6 represents the FDA (%) assessment of several classification schemes with the proposed E-RF-MDFS for SFPS. Even here, 8 datasets have been compared. It has been observed that Xalan v2.6 and Camel v1.6 have a constant FDA of around 97.5 % and 63 % for both DT and K-NN classifiers, respectively. For other datasets, the performance of K-NN is superior to DT. On comparing the performance of E-RF-based classifiers, the performance of the proposed MDFS is the best, with an FDA of 99.3 % (Xalan v2.6) than the ADASYN classifier.

A comparison of PCC values among various classifiers with the proposed E-RF-MDFS has been given in Fig. 7. The lines represent the algorithm's PCC score in the classification process. DT has been ranked last among all classifiers, whereas E-RF-MDFS remained top. DT is considered a poor classifier with a PCC of -0.53. E-RF-MDFS has been ranked top with a score of 0.82 because it was closest to +1 compared to E-RF-ADASYN.

Fig. 8 depicts the comparison of E-RF with other ensemble methods in terms of FDA and RPR for SFPS. During the instability of DT, a little variation in the layout of the effective decision tree resulted in often erroneous findings. Similarly, the K-NN does not learn or generate any discriminating function throughout the training phase. Hence, DT and K-NN methods gave poor performance in both FDA and RPR. Utilizing a representative sample of data and characteristics, the suggested E-RF decreases the correlation between trees and provides the best value of FDA and RPR.

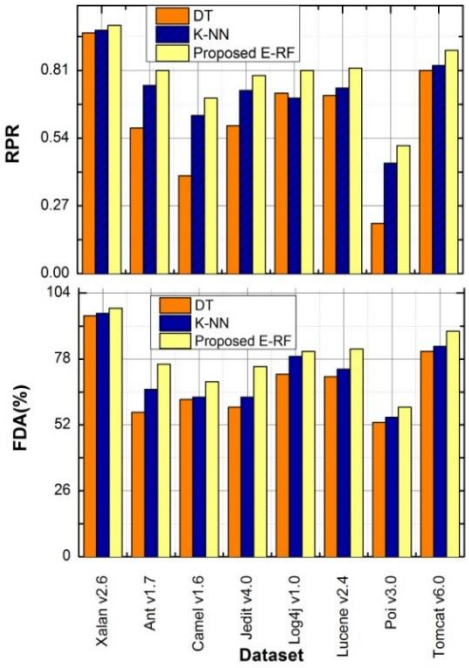

Figure. 8 Comparison of E-RF with other ensemble methods of FDA and RPR for SFPS

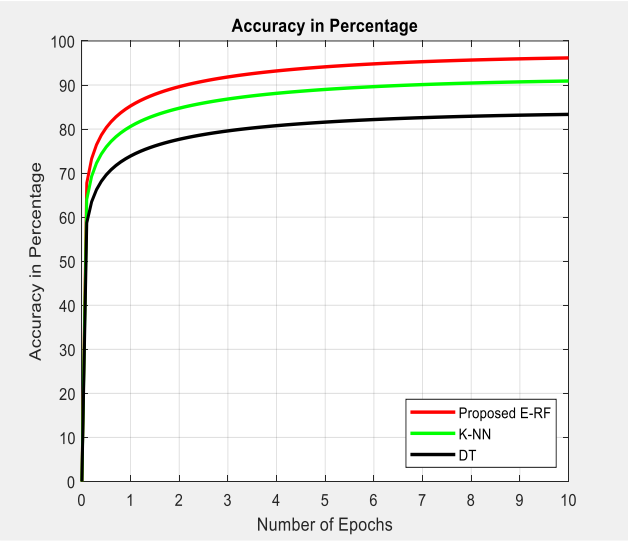

Figure. 9 Comparison graph between proposed E-RF, K-NN, and DT

Fig.9 shows the comparison graph between the proposed methodology and other methodologies. The proposed E-RF performance is better than others. The accuracy is calculated in terms of percentage. The E-RF accuracy is 97 %. The performance of the proposed E-RF-MDFS has improved for all the performance metrics. It selects the most relevant features through appropriate integration of bat and butterfly optimization through the BBO feature extraction algorithm. Also, MDFS first discovers differentiated software data objects and then finds samples near derived distinguished data items through evaluation of D_{ddo}^{total}.

5. Conclusion

The suggested research focuses on creating a new sampling approach known as ensemble-random forest with multi-distinguished-features sampling (E-RF-MDFS) for finding the best sample illustration for portraying the full dataset. The feature extraction technique has been carried out using bat-induced butterfly optimization (BBO). The tests are carried out on 8 datasets from the PROMISE repository to show that the suggested E-RF-MDFS is more efficient than other established approaches such as DT and K-NN.

The suggested technique has been evaluated using fault detection accuracy, actual positive rate, and Pearson's correlation coefficient. The suggested E-RF-MDFS achieves the greatest results because it can properly locate the main and supplementary DDO, resulting in the best RPR value. In addition, E-RF-MDFS has been given the highest score of 0.82 since it was the closest to +1 compared to E-RF-ADASYN. When assessing E-RF-based classifiers' effectiveness, the suggested MDFS outperforms the ADASYN classifier with an FDA of 99.3 percent (Xalan v2.6).

Conflicts of interest

"The authors declare no conflict of interest."

Author contributions

Conceptualization, A. Balaram and S.Vasundra; methodology, A. Balaram; software, A. Balaram; validation, A. Balaram and S. Vasundra; formal analysis, A. Balaram; investigation, A. Balaram; resources, A. Balaram and S. Vasundra; data curation, A. Balaram; writing—original draft preparation, A. Balaram; writing—review and editing, S. Vasundra; visualization. A. Balaram; supervision, S. Vasundra;

References

[1] X. Sun, X. Peng, K. Zhang, Y. Liu, and Y. Cai, "How security bugs are fixed and what can be improved: an empirical study with Mozilla", *Science China Information Sciences*, Vol. 62, No. 1, pp. 1-3, 2019.

[2] X. Sun, W. Zhou, B. Li, Z. Ni, and J. Lu, "Bug localization for version issues with defect patterns", *IEEE Access*, Vol. 7, pp. 18811-18820, 2019.

[3] H. Tong, B. Liu, and S. Wang, "Software defect prediction using stacked denoising autoencoders and two-stageensemble learning", *Information and Software Technology*, Vol. 96, pp. 94-111, 2018.

[4] L. Qiao, X. Li, Q. Umer, and P. Guo, "Deep learning-based software defect prediction", *Neurocomputing*, Vol. 385, pp. 100-110, 2020.

[5] S. Law, C. I. Seresinhe, Y. Shen, and M. G. Roig, "Street-Frontage-Net: urban image classification using deep convolutional neural networks", *International Journal of Geographical Information Science*, Vol. 34, No. 4, pp. 681-707, 2020.

[6] D. Palaz, M. M. Doss, and R. Collobert, "End-to-end acoustic modeling using convolutional neural networks for HMM-based automatic speech recognition", *Speech Communication*, Vol. 108, pp. 15-32, 2019.

[7] C. Manjula and L. Florence, "A deep neural network-based hybrid approach for software defect prediction using software metrics", *Cluster Computing*, Vol. 22, No. 4, pp. 9847-9863, 2019.

[8] T. Zhou, X. Sun, X. Xia, B. Li, and X. Chen, "Improving defect prediction with deep forest", *Information and Software Technology*, Vol. 114, pp. 204-216, 2019.

[9] D. Sharma and P. Chandra, "Linear regression with factor analysis in fault prediction of software", *Journal of* Interdisciplinary *Mathematics*, Vol. 23, No. 1, pp. 11-19, 2020.

[10] Y. Niu, Z. Tian, M. Zhang, X. Cai, and J. Li, "The adaptive two-SVM multi-objective cuckoo search algorithm for software defect prediction", *International Journal of Computing Scienceand Mathematics*, Vol. 9, No. 6, pp. 547-554, 2018.

[11] Z. Xu, J. Liu, X. Luo, Z. Yang, Y. Zhang, P. Yuan, and T. Zhang, "Software defect prediction based on kernel PCA and weighted extremelearning machine", *Information and Software Technology*, Vol. 106, pp. 182-200, 2019.

[12] A. Kaur and I. Kaur, "An empirical evaluation of classification algorithms for fault prediction in open source projects", *Journal of King Saud University-Computer and Information Sciences*, Vol. 30, No. 1, pp. 2-17, 2018.

[13] F. Yucalar, A. Ozcift, E. Borandag, and D. Kilinc, "Multiple-classifiers in software quality engineering: Combining predictors to improve software fault prediction ability", *Engineering Science and Technology, an International Journal*, Vol. 23, No. 4, pp. 938-950, 2020.

[14] R. Mousavi, M. Eftekhari, and F. Rahdari, "Omni-ensemble learning (OEL): utilizing over-bagging, static and dynamic ensemble selection approaches for software defect prediction", *International Journal on Artificial Intelligence Tools*, Vol. 27, No. 06, p. 1850024, 2018.

[15] F. Pecorelli, and D. D. Nucci, "Adaptive selection of classifiers for bug prediction: A large-scale empirical analysis of its performances and a benchmark study", *Science of Computer Programming*, Vol. 205, p. 102611, 2021.

[16] H. Turabieh, M. Mafarja, and X. Li, "Iterated feature selection algorithms with layered recurrent neural networks for software fault prediction", *Expert systems with applications*, Vol. 122, pp. 27-42, 2019.

[17] I. Tumar, Y. Hassouneh, H. Turabieh, and T. Thaher, "Enhanced binary moth flame optimization as a feature selection algorithm to predict software fault prediction", *IEEE Access*, Vol. 8, pp. 8041-8055, 2020.

[18] A. Balaram and S. Vasundra, "Prediction of software fault-prone classes using ensemble random forest with adaptive synthetic sampling algorithm", *Automated Software Engineering*, Vol. 29, No. 1, pp. 1-21, 2022.

[19] A. Mateen, S. Y. Nam, M. A. Haider, and A. Hanan, "A Dynamic Decision Support System for Selection of Cloud Storage Provider", Appl. Sci. Vol. 11, No. 23, p. 11296, 2021.

[20] O. Gültekin, E. Cinar, K. Özkan, and A. Yazıcı, "Real-Time Fault Detection and Condition Monitoring for Industrial Autonomous Transfer Vehicles Utilizing Edge", *Artificial Intelligence Sensors*, Vol. 22, No. 9, 3208, 2022.