AN EFFECTIVE SOFTWARE FAULT PREDICTION WITH HYBRID TECHNIQUE BASED ON CUCKOO SEARCH

*1Anju A.J, 2J.E. Judith

*1*Research Scholar, Computer Science, and Engineering, Noorul Islam, India.*

2*Professor, Computer Science and Engineering, Noorul Islam, India.*

*1*Email: ajanju184@gmail.com*

ABSTRACT

Software Defect Prediction (SDP) purposes to identify the fault-prone software modules to allocate limited resources for software reliability in a cost-effective manner. In this paper, the Hybrid Artificial Neural Network (HANN) is designed to predict software prediction. The training phase of the ANN is achieved with the assistance of Cuckoo Search Optimization (CSO). Initially, the pre-processing phase is enabled by the data normalization process. The extensive set of features are reduced using the dimension reduction process and is achieved by using Principal Component Analysis (PCA). Additionally, to enable an effective feature selection procedure, Quantum Theory-based Particle Swarm Optimization (QPSO) is used to improve the prediction performance of software defect analysis. With this technique, a better solution can be obtained for predicting the software defects; therefore, the existing problems related to accuracy metrics can be solved. To evaluate the performance analysis of the proposed technique, here; evaluation metrics such as accuracy, detection error, specificity, and sensitivity, etc. are measured and are compared to the regimes in force. The analytical analysis shows that the proposed approach offers more useful solutions for predicting defects.

***Keywords**: Artificial Neural Network, Cuckoo Search Optimization, Quantum Theory-based Particle Swarm Optimization, Principal Component Analysis, Software.*

1. INTRODUCTION

In recent years, the development of modern technological advancement makes software systems becoming more versatile and powerful. The software systems are highly incorporated with the reliability of the software, which fetches the attention of various researchers and developers are focusing on its development [1]. Also, software consistency typically has extensive code analysis and software unit testing, requiring more source and duration. [2]. Due to the necessity of maintaining software reliability, the software failure prediction technique is nowadays getting high consideration for automatically identifying the software defects from the software modules and also focusing on the software allocation process [3], [4]. Therefore, Software Deficiency Prediction (SDP) technique is highly utilized to identify the largest possible faults before software distribution, which plays a major part in ensuring the software quality [5], [6]. Besides, this technique is considered an essential step for enhancing the size and software complexity [7], [8]. In this area, the US defense department has spent $ 4 billion for a year to conduct research related to software bugs due to the reason for ensuring the quality of software. As a result of various researches based on SDP, the errors are diagnosed effectively with the accurate outcome by motivating the developers with limited time and manpower [9]. However, it has faced many difficulties in the lifecycle of software design and its development due to the failure of inaccurate predictions. Therefore, classifying the software modules is becoming an essential process to determine the faults to resolve. For the classification process, various technologies are adopted in this domain which including Support Vector Machine, Random Noise, Naive Bayesian Algorithm, Decision Trees (DT), and Neural Networks (NN) [10].

These machine learning technologies are considered very effective for the classification process [11]. In this work, we adopted the Principal Component Analysis (PCA) for the feature extraction process. It is an effective statistical way of changing the characteristics of a dataset to a fresh position of unrelated characteristics named principal components (PCs). Principal Component Regression (PCR) can be used to reduce the size of a dataset while keeping the variation of the dataset as low as possible. This purpose method is used to identify, whether PCA can be used to improve the performance of machine learning methods in the classification of these multivariate data. [12], [13]. This series of PCA experiments makes it easy to compare machine learning to the popular Chemistry Principal Component Regression (PCR) method, which combines PCA and linear regression [14]. Moreover, this article also focuses on utilizing the PSO technique for our work, which is belonging to the family of swarm intelligence optimization techniques. Swarm Intelligence Optimization technique incorporated with ant colony optimization, bee colony optimization, swarm particle optimization (PSO), etc. These technologies are imitating the behavior of a swarm of species, based on communication and preying approaches in an optimized way. The objective of this optimization is to obtain optimal results. When compared to other swarm intelligence algorithm, PSO is a very effective technique with fewer control parameters and possibly provide good performance [15]. Though, PSO has some drawbacks like low accumulation rate at which end of PSO phase, low accumulation effect, usually three parameters, and falling easily to the local minimum.

To increase the performance of the convolutional PSO, in this method, this technique adopts the Quantum theory-based PSO (QPSO) for achieving an effective optimization process. With this technique, we can derive the benefits of PSO, improved global search functions, velocity-free vector, and a single QPSO parameter, which specifies that QPSO is controlled easily. Hence, the article QPSO was applied to reduce the dimension and derive the matrix from the optimal QPSO solution used to extract the functionality. On the other hand, we used the cuckoo search algorithm to classify the pre-processed result to obtain results. The CS algorithm is an efficient swarm recognition algorithm toward solving a single goal for an optimization problem. It is generally recognized in science and Industry for its good performance. Nowadays, a lot of researchers have worked to increase the execution of the CS algorithm. Given the massive growth of difficulties in practical engineering problems, there is an urgent need to reduce the complex problem of optimizing the function of multiple criteria.

In the accompanying segment, this article provides definite data about the proposed work based on analyzing various literature that is explained in section 2. In section 3, this article discusses existing problems. Section 4 illustrates the proposed methodology where this article provides detailed information about the proposed methodology and, this article discusses PSO, QPSO, and Cuckoo Search Optimization algorithm for providing detailed information about the hybrid approach. Section 5 explains the result and discusses the part where this article discusses various results obtained by the performance analysis process.

2. LITERATURE REVIEW:

In this section, the number of articles that are related to the topic of software defects prediction is utilized based on analyzing its techniques and overall outcome. It will be explained in detail below.

The software defect prediction (SDP) is a class imbalance problem that is used in the dataset where are more non-defective instances than defective instances. Therefore, Shuo Feng *et al.,* [16] have introduced a Complexity-based Oversampling TEchnique (COSTE) to address the class imbalance problem in SDP. Here, the COSTE method incorporates the pairs of defective instances with comparable complexity to create the synthetic instance. As a result, this method is recommended as an efficient alternative to rectify the SDP’s class imbalance problem. However, Linchang Zhao *et.al.,* [17] have developed an efficient SDP model described as Siamese Parallel Fully Mesh Networks (SPFCNN) that combines benefits for deep learning and siamese networking in a single system. Training this model follows Adam's algorithm to find the best weights. Then the smallest value for the single formula is the training goal for the model of SPFCNN. Most importantly, this article carefully compared the SPFCNN to current SDP methods used for six open-source datasets with a repository of NASA. Similarly, Zhou Xu *et al.,* [18] have presented a framework of forecast defect called KPVE that combines two methods namely Weighted Extreme Learning Machine (WELM)and kernel Principal Component Analysis (KPCA). Here, the structure includes two main stages. At the initial level, KPVE tries to develop represent data characteristics. It uses the KPCA technique to project raw data into space with latent functionality via a non-linear equation. In the secondary stage, the KPWE tries to reduce class imbalances. Likewise, Yuanxun Shao *et al.,* [19] have suggested a class-weighted correlation rule-based (CVCAR) software failure prediction model, which uses a media-weighted framework instead of a traditional media reliability approach to manage class imbalances, and uses a heuristic based on correlation to assign function weights. Besides, it optimizes phase rankings, snippets, and predictions based on weighted support. Similarly, Alireza Souri *et al.,* [20] have suggested Feature selection which uses behavioral modeling and formal validation of the defect prediction model based on hybrid machine learning processes, Particle Group Optimization (PSO), and Multilayer Perceptron (MLP) Algorithms in PSO. In this case, forecasting failure is considered a form of behavior that is subject to official verification. Predictive failure behavior is classified into two types: predictable behavior with reduction behavior. A formal model has been developed for each behavior. The developed behavior models are presented as a system of marked transitions (LTS). The Process Evaluation Model (PAT) is used to evaluate behavioral patterns. The exactness of the disappointment forecast technique is gotten from some current details, for example, impeding Interdependent properties and accessibility as direct equations of fleeting rationale. Likewise, Qinbao Song, *et al.,* [21] have presented an imbalanced learning method for the software defect prediction. Here, the Matthews correlation coefficient (MCC) is used as an unbiased performance measure.

In software defect prediction, class imbalance and parameter selection are the two major problems. Therefore, Xingjuan Cai *et al.,* [22] have presented a hybrid multi-objective cuckoo search under-sampled software defect prediction model based on SVM (HMOCS-US-SVM) method to rectify the class imbalance problem. Here, hybrid multi-objective cuckoo search with dynamical local search (HMOCS) was utilized to select the non-defective sampling. It also utilized the three under-sampled methods to select the non-defective modules. Similarly, QIAO YU *et al.,* [23] have utilized feature subset selection and feature ranking methods to examine the effectiveness of the cross-project defect prediction (CPDP). Here, datasets like NASA and PROMISE were utilized to examine the CPDP's effectiveness. Likewise, Haitao He *et al.,* [24] have presented an Ensemble MultiBoost based on RIPPER classifier for prediction of imbalanced Software Defect data, called EMR_SD. The original feature from the dataset was identified through Principal Component Analysis (PCA) technique. Moreover, the combined sampling method of adaptive synthetic sampling (ADASYN) and random sampling without replacement was performed to solve the data class imbalance problem. In addition to that Thomas Shippey *et al.,* [25] have utilized Abstract Syntax Tree (AST) n-grams to identify features of defective Java code to enhance the defect prediction performance. here the relation between the AST n-grams and faults was determined through non-parametric testing. Similarly, Tianchi Zhou *et al.,* [26] have presented a deep forest model to build the defect prediction model (DPDF). This model transfers the random forest classifiers into a layer-by-layer structure to enhance the performance of the defect-predicted techniques.

The following section is provided with brief information about the problems of the existing technique. Therefore, the problems in the existing technology can be rectified with the novel technique.

3. PROBLEM DEFINITION

In this section, this article discusses the current problems of software defect prediction and its techniques.

- The existing technique is not providing better results in dealing with high and low dimension space while creating a mapping between them.
- The light-weighting nature of existing work in terms of compression and loosely connected network may lead to challenges.
- The existing technique is not as much as good for an effective feature selection process.
- There is a need for incorporating an effective technique to improve performance by utilizing adequate machine learning techniques.

The above-given problems should be resolved with a new effective technique; hence we provide a novel technique based on Hybrid-ANN for this work, the detailed information about the proposed technique and its processes are discussed below.

4. PROPOSED METHODOLOGY

This work is mainly focussing on providing effective software defects prediction with the utilization of novel technique HANN which incorporated with Artificial Neural Network and Cuckoo Search optimization of an enhanced prediction process. This technique is considered a tool to provide a cost-effective prediction process. For effective prediction, this work has experienced various research articles for analyzing the existing techniques and their problems. In the proposed work, the novel technique can provide an enhanced mechanism predicted to defects in the software system. The overall architecture of the proposed framework is displayed as follows,

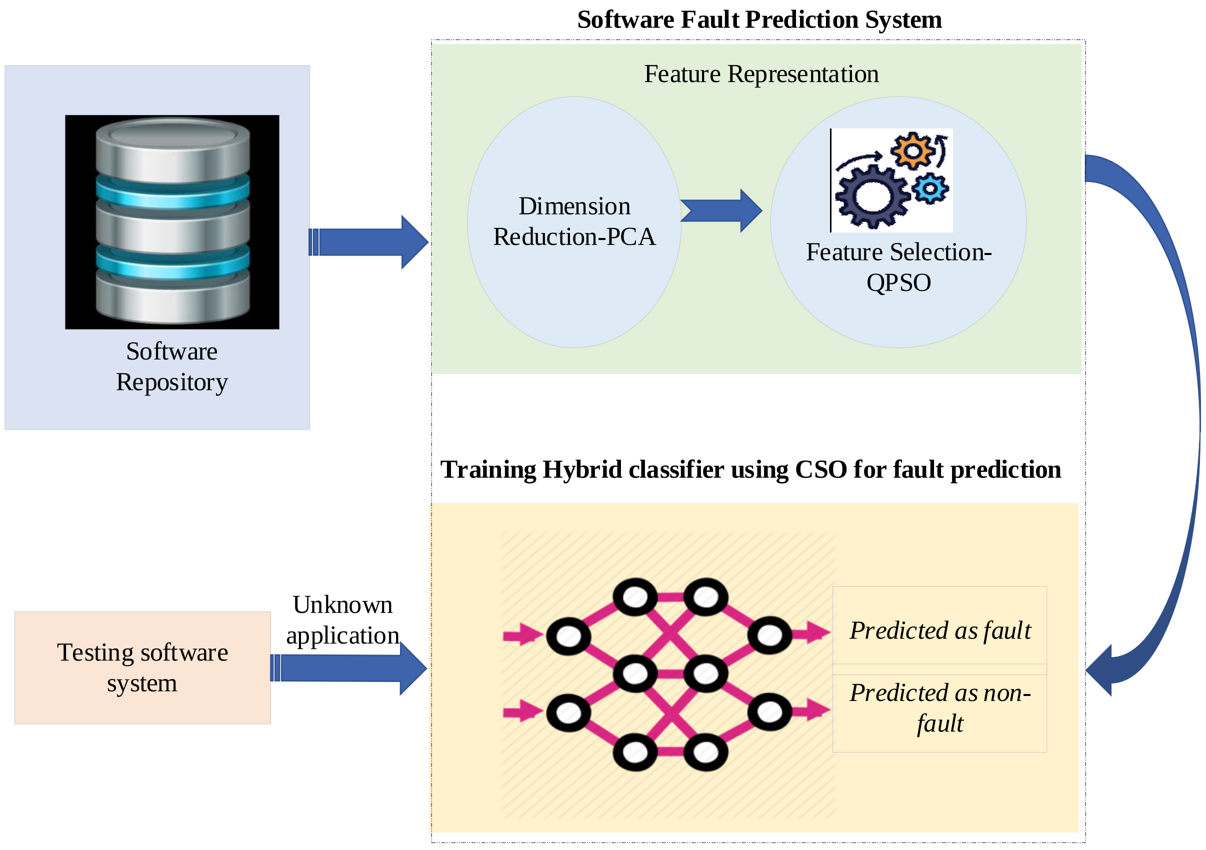


Figure 1: The step-by-step procedure of the proposed framework

The execution process for this framework is based on four steps that include dataset collection, pre-processing, feature representation, and prediction process, etc. In the following section, each step in the software prediction process is explained in detail.

4.1 Data Collection Module

In this module, the input data is utilized from the internet-sourced dataset is defects-predictor-master. This model is associated with massive data that cannot be directly utilized for the prediction process. Then the dataset is prepared for further improvement by executing the data normalization process.

4.2 Pre-Processing-Data Normalization

The data normalization process is processed for reducing the data for ignoring the redundant data of the database. In this approach, the highly enabled process for executing the normalization processes through insertion, deletion, and updating process. Therefore, the dataset is becoming simple to execute the upcoming steps.

4.3 Dimension Reduction and Feature selection

In this phase, the two significant processes are dimension reduction and feature selection process which are utilized for fetching only the essential data from the dataset. The initial process is executed by adopting the dimension reduction process with the utilization of Principle Component Analysis (PCA). For extracting an effective set of features from a high dimensional feature vector, the dimensionality reduction approach is utilized with the help of PCA. With this technique, the low dimensional feature component from the dataset is extracted from the massive dataset for further process. Thus, errors accomplished while inducing higher dimensional feature space to the classifier are reduced.

Then the feature selection process is seen as a combinatorial enhancement issue that points to finding optimal functions for a subset of the original data set that still reliably describes the original data [27]. The whole feature representation process consists of two main stages: (i) determining the minimum reductions and (ii) evaluating the selection of traits. This makes them quicker than machine learning techniques that assess subsets of functions through a learning algorithm validation system. This article has utilized an improved PSO procedure to improve the presentation of traditional PSO. For an effective capacity choice cycle, this article utilized a PSO calculation dependent on the quantum hypothesis. Here this technique incorporated with the itemized data about PSO and QPSO cognizance calculations.

4.3.1 Mathematical modelling of PSO algorithm

PSO technique is an effective tool for managing optimization problems [28] and is an appropriate model for the feature selection approach. The calculation is an arbitrary pursuit and equal optimization calculation, which is straightforward, simple to actualize, and exceptionally proficient, and requires less boundary change. This calculation is reasonable for logical exploration as well as for the designed application. It begins from the stochastic arrangement with looking for ideal arrangement repeatedly; what's more, it assesses the nature of arrangement thinking about wellness.

The calculation begins from an arbitrary arrangement, looks through the ideal arrangement iteratively, and assesses the nature of the arrangement by thinking about wellness. Assume that a measurement space comprises a populace for particles. The area of particle I in an n −dimensional space is X. Particle speed V. In the functional cycle, particle speed and position can be represented as follows:

$$\begin{cases} V_{id}^{k} = wV_{id}^{k} + c_1 r_1 \left(P_{id}^{k} - X_{id}^{k}\right) + c_2 r_2 (P_{gd}^{k} - X_{id}^{k}) \\ X_{id}^{k+1} = X_{id}^{k} + V_{id}^{k+1} \end{cases} \tag{1}$$

w is inertial weight, $d = 1, 2, \cdots$; k is referred to as a number of emphases; c term defines quickening elements, r are irregular numbers going on [0,1]. Equation (1) demonstrates the particle speed was refreshed the three sections: initially, the segment is the memory of the previous movement of the particles, which is called the "inertial" part; the latter segment differs from the present state of the particle to ideal state through which it has passed, it may be considered "self-accepting"; then the third part is separate the current state of a particle and best position in the assembly, it can be considered as "social experience". The cycle stream of PSO calculation as per the following statements:

1) Induce particle swarm which includes speed of each particle and irregular position.

2) To calculate the fitness value for every particle in each cycle.

3) Update pi and PG as indicated by a fitness to update the position and speed of the particles.

4) Decide if the most extreme number of cycles or worldwide ideal position was reached, whether fulfilled the minimum limit.

If satisfied, complete the iteration; otherwise, repeat the steps of 2–4.

4.3.2 Quantum theory-based PSO

To select the optimal set of feature space, a novel Quantum Theory-based PSO technique is utilized which is based on the perspective of quantum mechanics. The quantum particle swarm optimization algorithm is a global convergence guarantee algorithm. The QPSO assumes that the particle swarm optimization system is a particle system that satisfies the basic hypothesis of quantum mechanics. In quantum space, particles do not have a specific trajectory, which allows particles to search for the global optimal solution in the entire feasible solution space. The lack of a defined trajectory implies that in quantum space, the position and speed of the particles cannot be measured at the same time [29]. The reason behind improving the default PSO is that to improve the performance of a low convergence rate. In addition, the advancement of PSO is done by improving performance analysis of traditional PSOs is the application of quantum theory to detect particle behavior. This is because of the way that particle collection in quantum space is depicted by a bound state made by a potential fascination field in particle sports focus. The particle's quantum states can some likelihood be situated anytime in the arrangement space. Particles with an aggregated state can look for the solution in all possible space, but do not deviate endlessly. In the quantum PSO model, the condition of a particle can be portrayed through the wave work Ψ (k, t) rather than speed and position. The dynamic behavior of particles is different from the behavior of prototype PSO particles. In this specific circumstance, the likelihood of a particle showing up at position X is dictated by the likelihood thickness work Ψ (k, t) ^ 2, the state in which relies upon the expected field in the particle $\Psi(x,t)^2$,

The particles travel according to the subsequent iterative equations

$$X_i(t+1) = g_i(t) + \propto . |m_{i.best} - X_i(t)| . ln \frac{1}{u_i(t)}, \ ifs \ \geq 0.5 \qquad (2)$$

$$X_i(t+1) = g_i(t) + \propto . |m_{i.best} - X_i(t)| . ln \frac{1}{u_i(t)}, \ ifs \ < 0.5 \quad (3)$$

where u_i, s is random numbers evenly distributed in the interval [0,1]. The parameter ∝ is known as the compression-expansion ratio. Also

$$g_i(t) = \varphi_i(t). G_{i.best}(t) + \left(1 - \varphi_i(t)\right). G_{global}(t) \quad (4)$$

$$m_{i.best} = \frac{1}{Q} \sum_{i=1}^{Q} G_{i.best}(t) \quad (5)$$

Where φ_i is an equally shared random number in [0, 1] interval, Q is a number of particles, and m_ (i.best) was determined by the center of best locations for all particles in the population.

The achievement QPSO model is when absolute deviation among C (t + 1) and C (t) is 10 times slower when the algorithm is closed; Otherwise, G_mak reaches the highest amount of iterations. Where δ is the learning threshold.

Pseudocode of the proposed Feature selection process

```
Initialize the positions and the $\boldsymbol{G_{t.best}}$ positions of all the particles
Do Calculate $\boldsymbol{m_{t.best}}$ using Equation (5) for particle i, where i = 1, 2. . . Q
Select a suitable value ∝
For particles i = 1 to Q
Calculate the cost value of particle $\boldsymbol{i}$ according to Equation (3)
Update $\boldsymbol{G_{t.best}}$ using Equation (4)
Update $\boldsymbol{G_{global}}$ as follows
If$\boldsymbol{C(G_{i.best}(t) < C(G_{global}(t-1))}$then$\boldsymbol{G_{global}(t) = G_{i.best}(t)}$ else$\boldsymbol{G_{global}(t) = G_{i.best}(t-1)}$
For dimension 1 to d
$\boldsymbol{\varphi}$= rand (0,1)
u = rand (0,1)
If s = rand (0,1) ≥ 0.5
Update particle positions using Equation (2)
else
Update particle positions using Equation (3)
Until the terminal condition is met
```

4.4 Proposed Hybrid Machine Learning Classifier for fault prediction

The proposed hybrid model incorporates an ANN framework for predicting software fault rate, whereas the network is trained with the help of a pre-defined cuckoo optimization strategy. Typically, ANN has the capability to learn the basic process structure based on previous data and generalize the complex mathematical relationships between acquired knowledge of input and output data. This model can predict a more complex system based on its architecture. During, the complexity of the adsorption process, the computational intelligence-based ANN model offers greater flexibility than a model for statistical, it can complex model data sets by non-linearities. ANN consists of three parts such as a hidden layer, input layer, and output layer. These layers are used to characterize neurons in the brain, and nodes are connected. Specific mapping by ANN is related to this structure, then the weight values among neurons. As a very broad representation and transformation, ANNs work in parallel and widely on many interconnected nodes.

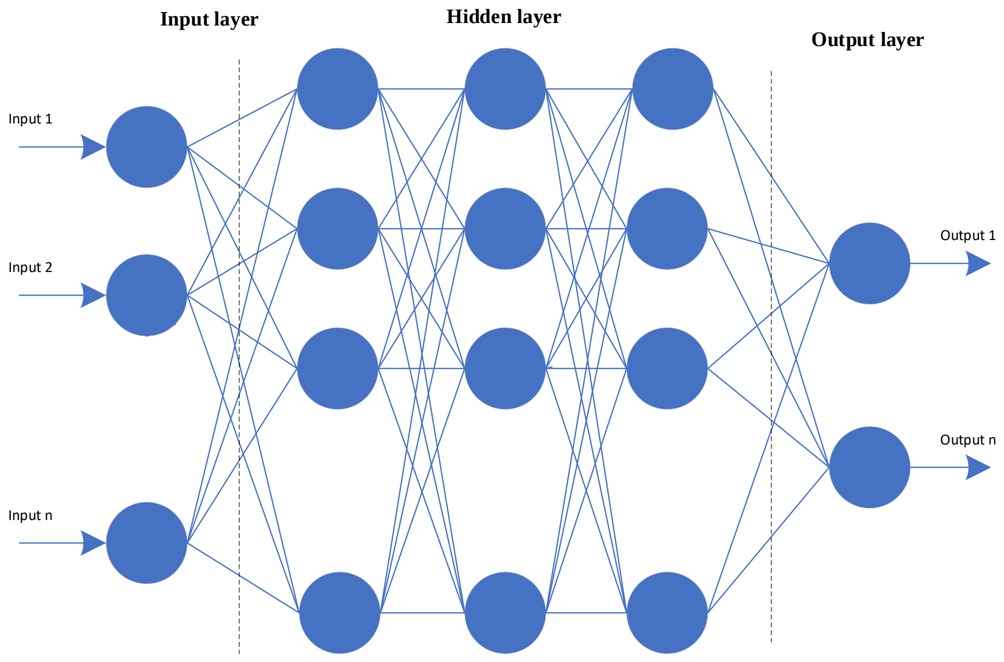


Figure 2: illustrates the execution process of ANN

An ANN can be developed by using the black-box theory. Similar to examining the human brain by repeating a similar stimulus, ANN controls a pair of inputs and outputs that function normally without a priory assumption showing or limiting the connection between the input-output. Using the ANN model involves constructing a response by transferring weights to denote the exact relationship that exists among inputs and outputs. All data were randomly divided into two types: training (70%) and testing (30%). The training dataset can be used to train the ANN model. In the Testing dataset, which allows assessment of predictive capabilities for the ANN model. During the training test, ANN is effectively announcing the destination between the input and output layers. In figure1, the process of execution of ANN is shown clearly.

The ANN does not provide clear details on this objective. The ideal application of the ANN is the recognition of structural patterns. In such work, ANN has a group of functions available with which the input function model can be divided into one or more types. In such cases, the more relevant information is made available to the network at the same time. The efficiency of the ANN model depends upon variables like the number of neurons in hidden layers, output layers, and also the type of transfer function. In a research report, an ANN model calculation is performed with a version of MATLAB R2020a tool, the size of Input is 94×1, the target size is 94×1, and the validation ratio is 15%. Here the training process is carried out with the help of a cuckoo search optimization algorithm. This means that the weight values adopted in the network are trained optimally to achieve an error-free mechanism with a better prediction rate.

4.4.1 Hyperparameter tuning

This section describes the hyperparameter tuning of the proposed network using the optimization algorithm. The hyper-parameters play a crucial role in obtaining an optimized learning model for the dataset. Here the standard Cuckoo Search (CS) algorithm is utilized to find suitable heuristics for tuning the hyper-parameters of an ANN. It uses Levy Flights for randomization and uses accuracy evaluation of dynamically constructed ANN architecture as its objective function. The created networks are trained, verified, and tested on the given dataset for a fixed number of epochs before being returned to the modified cuckoo search algorithm for final testing accuracy.

4.4.2 Cuckoo search algorithm

The weight values adopted in the ANN framework are trained with the help of the optimization algorithm used in this research. The Cuckoo Search algorithm is characterized by its ability to simulate cuckoo breeding tactics in the wild. The coward searches at random or similarly for a nest that is best suited to spawning. The main system of the CS algorithm is to simulate the parasitic role of a coward and use Levi's flight to simulate a random walk with a coward. In nature, the movements of many birds are reminiscent of Levi's flight. Levi's flights include frequent short and long flights. Phases of flight follow Levi's distribution, and regular long-haul flights can increase the search area and prevent a decrease in local optimization. To clarify the determination of this algorithm, three simple and idealized rules for a particular research algorithm are specified [30].

- Specific cuckoo spawns a cuckoo egg, and then randomly picks a place to hatch.
- The best egg from the nest is stored and used in this monotony.
- The nests are useful for the number is fixed and the probability that nests are found in the nest is $p_a \epsilon$ [0, 1].

When the egg is found, the nest owner either builds a new nest in a different location or throws a loose egg. According to the CS Code of Conduct above, if x_i^{t+1} designs the next formation solution, x_i^tdenotes the modern solution and represents a brittle solution in the solution.

The system for refreshing your browser location is as follows:

$$x_i^{t+1} = x_i^t + \rho L\acute{e}vys(\delta) \tag{6}$$

In the equation. (6) ρ represents the scale factor of the size of the displacement step and ρ> 0, typically ρ = 1. The fit of ρ generally depends on the volume of the difficulty area. Therefore, it is important to find the exact benefit of a more efficient search without jumping out of space. Decide. In (5), the above formula is a stochastic equation using random walks. Mark's chain is typically used as a random trait, whose next position depends only on the current position (the first element in the above formula) and the probability of intersection (the second element). Any Levi flight score is generated using the Levi probability distribution.

$$Lévys(\delta) \sim u = t^{-1-\delta}, (0 < \delta \leq 2) \quad (7)$$

In addition to equations (1) and (2), the algorithm of cuckoo also includes a local search. The algorithm of CS uses a switching parameter to balance between local random research and global search. They described the process of

$$x_i^{t+1} = x_i^t + \rho \times s \otimes \mathrm{H(Pa - c)} \otimes (x_j^t - x_k^t) \quad (8)$$

In the equation. (7) s denotes the step size , x_i^t and x_k^t are two solutions chosen at random from the whole group. c is a random number generated by a similar model going from 0 to 1. H (u) shows the Heaviside function. After the introduction, the pseudo-code of the CS algorithm was displayed in algorithm 1.

Pseudocode for the Training process of ANN using an optimization model

```
Fitness function $\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{x} = (\boldsymbol{x_1}, \dots \boldsymbol{x_d})^{\boldsymbol{T}}$;
Initializing a population of n nests, $\boldsymbol{x_i}(\boldsymbol{i} \leq \boldsymbol{n})$;
while (t< Max Generation) or (stop criterion) do
Update cuckoo xi by Lévy flights via Eq. (6);
Get its quality/fitness Fi;
Choose a nest xj randomly;
if (Fi > Fj) then
```

```
replace xj by the new solution;
Use fraction (pa) to reset worse nests via Eq. (8);
Keep the best nest and rank the solutions, find the current best;
Final result output and presentation.
```

5. RESULT AND DISCUSSION

5.1 Dataset Description and evaluation metrics

There are several datasets available for the software fault prediction, namely PROMISE, AEEEM, NASA, etc., In this work, we utilize the publicly available PROMISE dataset. The dataset is available in the link [31]. It stores a collection of datasets that are commonly used by the software engineering research community to construct predictive software models. These datasets are noise-free and have no missing values. This paper employs the five most widely used datasets from this repository (CM1, JM1, KC1, KC2, and PC1). Each of the considered datasets possesses several software modules with input as the quality metrics. The output of each of the modules includes a defective or non-defective case, which identifies the presence of faults in any of the respective modules. whereas, the details about the selected dataset are given in table 1.

Table 1: Details of PROMISE dataset

Dataset	Number of instances	Number of attributes	Defect	Non-defect
CM1	498	22	49	449
JM1	10885	22	8779	2106
KC1	2109	22	326	1783
KC2	522	22	105	415
PC1	1109	22	1032	77

Furthermore, the effectiveness of the system is validated by comparing the evaluation metrics of the proposed approach with the existing approaches. As result, the proposed technique was incorporated with various performance results evaluation process in terms of analyzing the accuracy, sensitivity, specificity, and precision. Therefore, the proposed technique is proven as a better technique. This technique is implemented into the platform of MATLAB 2020a. The results obtained from various techniques such as firefly optimization, PSO, and Genetic Algorithm are compared with the proposed technique's results. The result of the proposed technique is shown better results than the existing one.

5.2 Comparative analysis of proposed and existing feature selection model

Table 2: Comparative analysis of proposed and existing feature selection model

Feature selection using QPSO				
S.NO	Accuracy	Specificity	Sensitivity	Precision
1	98.627	95.37	96.457	96.567
2	96.456	96.246	97.476	97.46
3	97.568	97.827	97.35	98.847
4	98.371	97.352	98.356	99.256
Fire Fly Optimization				
5	94.322	91.5	92.22	92.1
6	94.5	92	93.5	92.2
7	95.5	93.7	94	94
8	96	95	95	95.7
PSO				
9	87.82	85.4	89.82	80.4
10	89.9	90	90.7	85
11	90.9	92	92	90
12	92	94	94	91
Genetic Algorithm				
13	82.22	75.2	78.22	76.2
14	86.5	78.3	79.5	79
15	88.5	85.3	83.5	84
16	90.5	90	91	87

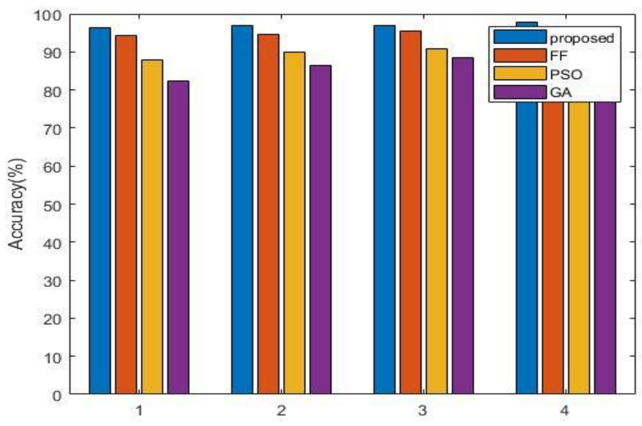


Figure 3: Results of accuracy based on other existing techniques

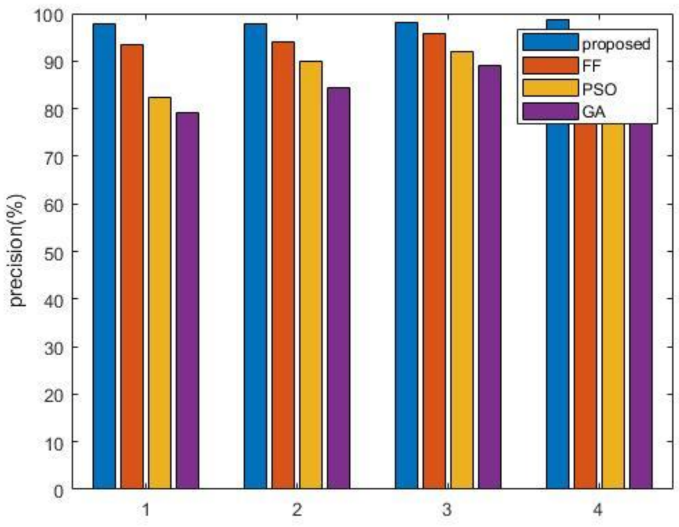


Figure 4: Results of precision based on other existing techniques

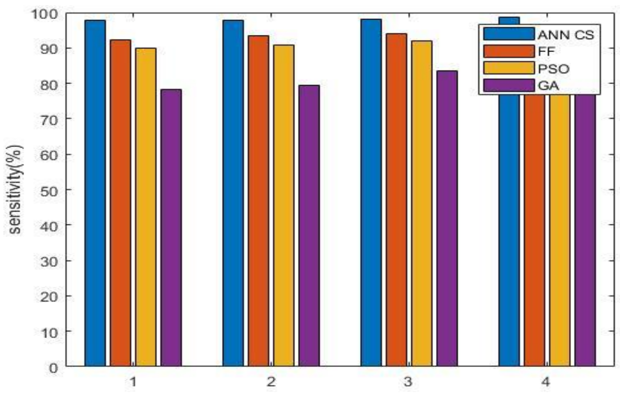


Figure 5: Results of sensitivity based on other existing techniques

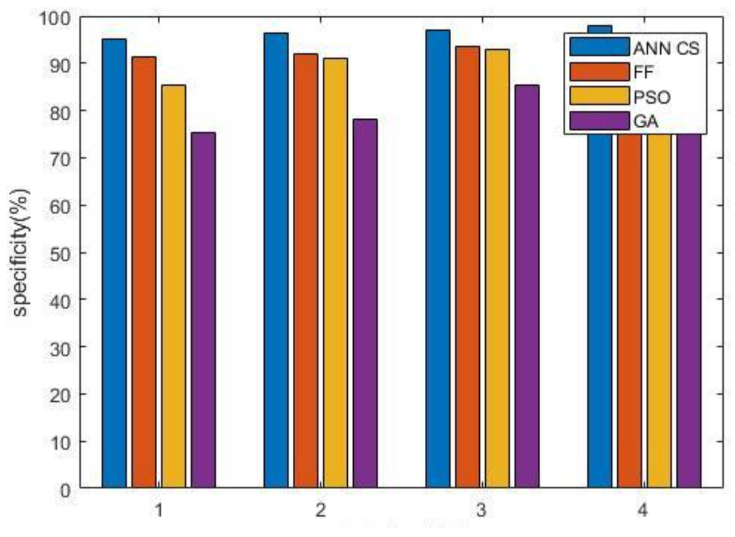


Figure 6: Results of specificity based on other existing techniques

The above graphical image illustrates the result obtained based on analyzing the performance metrics. Figure 2 illustrates the accuracy-based results, Figure3 is based on the comparative results of precision, figure5 shows the results of sensitivity-based results, and figure 6 is based on specificity-based results. All these images are represented in the graphical image. This has proven that the proposed is better than all other existing techniques.

5.3 Performance analysis of software defect prediction model

The following tables are provided based on different results obtained while the software defect prediction system.

Table 3: Shows the results of the proposed technique with other existing techniques

Proposed HANN				
S.NO	**Accuracy**	**Specificity**	**Sensitivity**	**Precision**
1	96.266	95.1705	97.7391	76.2
2	96.8	96.5	97.8	79
3	97	97	98	84
4	97.8	97.8	98.7	87
Existing ANN				
5	87.58	86.457	91.627	67.38
6	90.142	88.236	89.082	71.672
7	91.73	89.154	90.193	78.48
8	89.26	90.134	89.564	77.673

Existing SVM				
9	79.639	75.948	74.309	66.243
10	76.28	78.635	73.268	63.487
11	77.32	73.325	77.249	62.985
12	75.3489	77.256	79.764	63.89

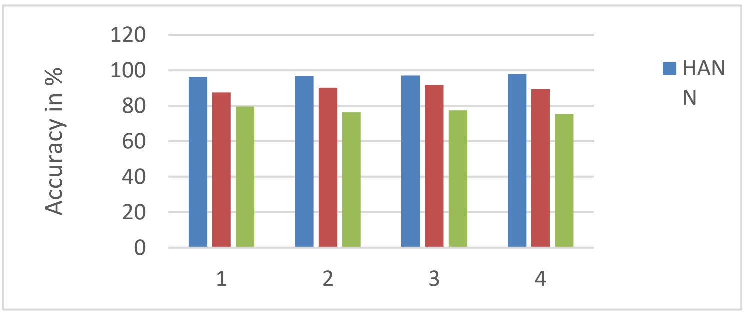


Figure 7: Shows the results of accuracy based on other existing techniques

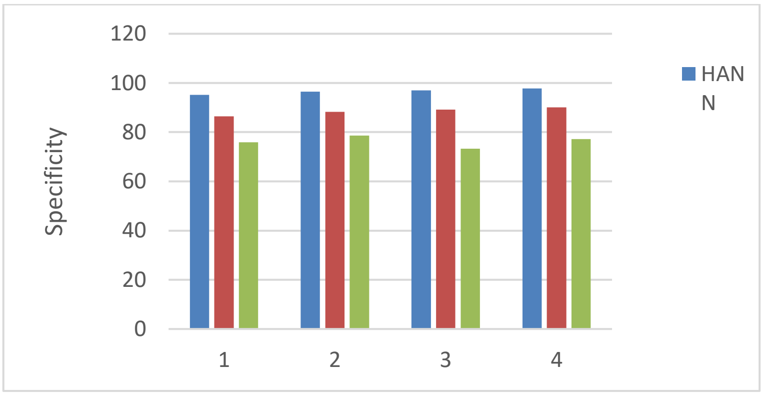


Figure 8: Shows the results of specificity based on other existing techniques

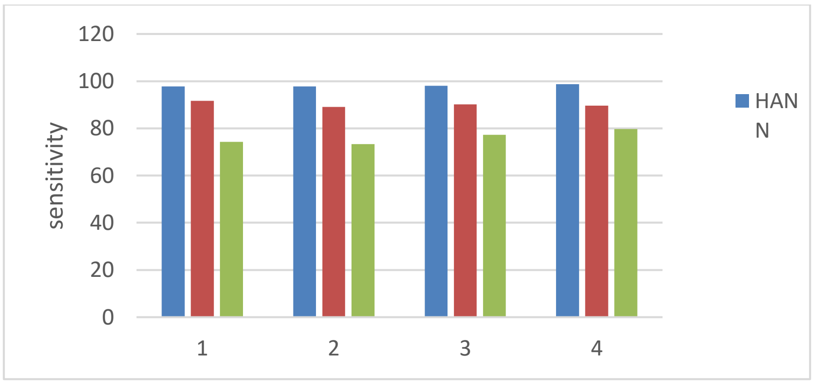


Figure 9: Shows the results of sensitivity based on other existing techniques

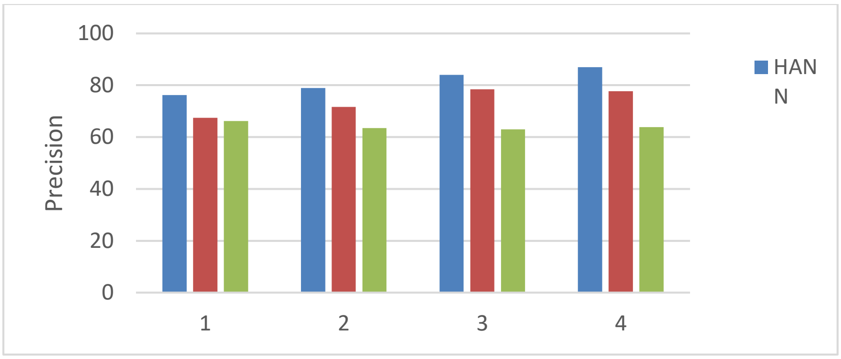


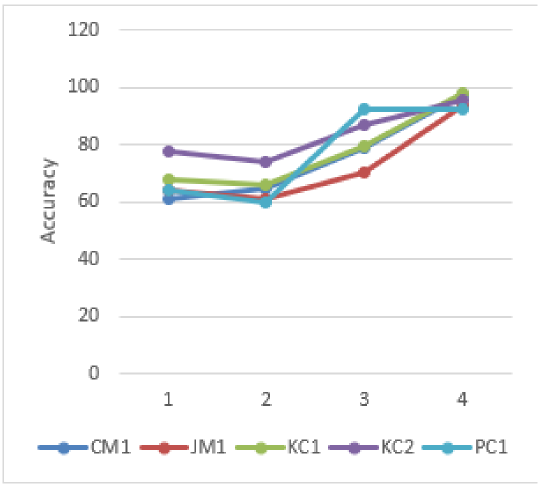
Figure 10: Shows the results of accuracy based on other existing techniques

The above graphical image illustrates the result obtained based on analyzing the performance metrics. Figure 7 illustrates the accuracy-based results, Figure 8 is based on the comparative results of specificity, figure 9 shows the results of sensitivity-based results, and figure 10 is based on precision-based results. All these images are represented in the graphical image. This has proven that the proposed is better than all other existing techniques.

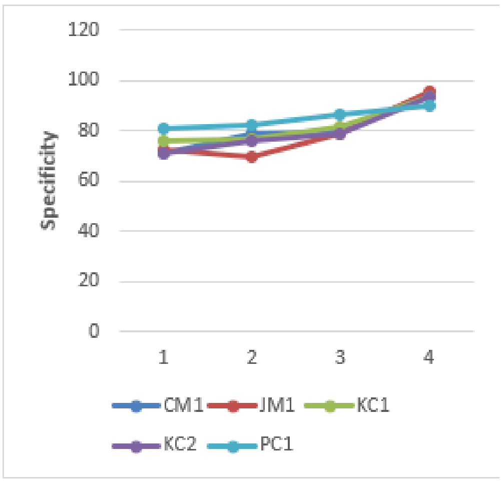
Table 4: Comparative analysis for PROMISE dataset

Algorithm	**CM1**	**JM1**	**KC1**	**KC2**	**PC1**
Accuracy					
Random forest	60.98	63.97	67.99	77.81	63.98
Naïve Bayes	64.57	60.78	65.87	74.00	60.00
SVM	78.69	70.32	79.24	87.12	92.45
Proposed	97.59	93.84	97.82	95.72	92.35
Specificity					
Random forest	71.29	72.38	75.89	70.71	80.99
Naïve Bayes	78.65	69.77	76.85	75.98	82.34
SVM	79.08	79.00	81.27	78.96	86.59
Proposed method	94.78	95.72	93.2	93.26	89.83
Sensitivity					
Random forest	70.09	66.72	72.54	70.32	82.31
Naïve Bayes	71.03	68.98	74.33	77.24	87.98
SVM	78.97	70.89	81.37	84.35	80.98
Proposed method	96.54	99.37	90.22	95.42	96.18

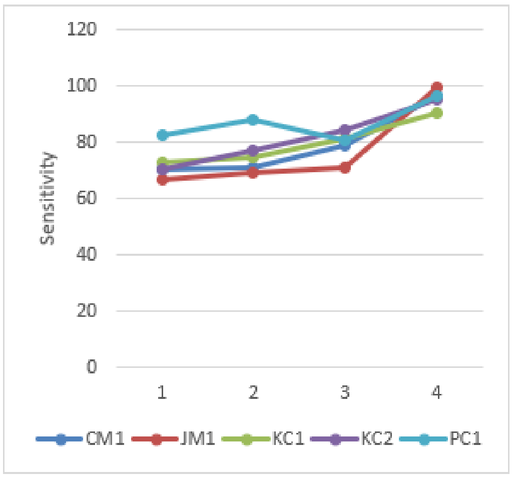
Comparative study for the dataset like CM1, JM1, KC1, PC1, and PC2 is presented in Table 4 which shows improved classification accuracy, specificity, and sensitivity for the PROMISE dataset using the proposed approach. The comparative study of the PROMISE dataset is shown in table 4. From the table, it is evident that the proposed approach obtains better performance when compared with other techniques.



(a) accuracy



(b) specificity



(c) sensitivity

Figure 11: Comparative analysis of proposed method with another classifier for PROMISE dataset

In Figure 11, we present a graphical performance analysis for software defect prediction. This figure shows a comparative analysis of random forest, Naïve Bayes, and SVM with proposed hybrid classification. From the graph, it is clear that the present approach using a hybrid technique gives better accuracy when compared to other techniques.

5.4 Comparative Analysis

A comparative analysis of the proposed technique with the existing technique is provided in terms of accuracy, specificity, sensitivity, and precision. The results based on accuracy for HANN are 96.266, 96.8, 97, and 97.8. The Firefly based results 94.322, 94.5, 95.5, and 96, For PSO, 87.82, 89.9, 90.9, and 92, For GA, 82.22, 86.5, 88.5, and 90.5. The result obtained based on specificity for HANN is 95.1705, 96.5, 97, and 97.8, for firefly, 91.5, 92, 93.7, and 95, For PSO, 85.4, 90, 92, and 94, for GA, 75.2, 78.3, 85.3, and 90. The result obtained based on sensitivity for HANN is 97.7391, 97.8, 98, and 98.7, for firefly, 92.22, 93.5, 94, and 95, for PSO, 89.82, 90.7, 92, and 94, For GA, 78.22, 79. 5, 83.5, and 91. The result obtained based on precision for HANN is 76.2, 79, 84, and 87, for firefly, 92.1, 92.2, 94, and 95.7, for PSO, 80.4, 85, 90, and 91, for GA, 76.2, 79, 84, and 87. In short of these results, the proposed work is efficient than all other existing techniques. This experimental study shows that the proposed approach is capable to obtain better classification results. This method provides reliable and significant performance which can be used for software defect prediction models. Therefore, the proposed work can be concluded as a better technique for the software prediction process.

6. CONCLUSION

In this paper, the hybrid Artificial Neural Network is associated with the Cuckoo Search-based optimization technique for obtaining the effective software prediction process. It is an important area of research that has explored various strategies to enhance the performance to determine software module failure using software flags. This work mainly dealt with the problems of classification based on the large dataset using the hybrid model to reduce features and classification. Finally, the model for validating the proposed approach to validating time and memory usage in fault prediction is worse than the current study. Moreover, the proposed work is evaluated with various performance metrics for analyzing the proposed technique with the existing technique; therefore, the obtained results are proven to be a better technique than the existing technique. In future work, the proposed work to be enhanced by adopting the hybrid technique for fault prediction with the utilization of a meta-heuristic algorithm. Besides, future studies will analyze other functional features of the fault prediction method for example Completeness, reliability, and fairness.

REFERENCES

[1]. Huo, Xuan, Ming Li. On cost-effective software defect prediction: Classification or ranking? Neurocomputing 2019; 363:339-350.

[2]. Deng, Jiehan, Lu Lu, Shaojian Qiu. Software defect prediction via LSTM. IET Software 2020.

[3]. Li, Jiahua, Ali Yamini. Clustering-based software modularisation models for resource management in enterprise systems. Enterprise Information Systems 2020; 1-21.

[4]. Sun, Zhongbin, Jingqi Zhang, Heli Sun, Xiaoyan Zhu. Collaborative filtering-based recommendation of sampling methods for software defect prediction. Applied Soft Computing 2020; 90:106163.

[5]. Haouari, Ahmed Taha, Labiba Souici-Meslati, Fadila Atil, Djamel Meslati. Empirical comparison and evaluation of Artificial Immune Systems in inter-release software fault prediction. Applied Soft Computing 2020; 96:106686.

[6]. Wu, Fei, Xiao-Yuan Jing, Ying Sun, Jing Sun, Lin Huang, Fangyi Cui, Yanfei Sun. Cross-project and within-project semisupervised software defect prediction: A unified approach. IEEE Transactions on Reliability 2018;67(2):581-597.

[7]. Majd, Amirabbas, Mojtaba Vahidi-Asl, Alireza Khalilian, Pooria Poorsarvi-Tehrani, Hassan Haghighi. SLDeep: Statement-level software defect prediction using the deep-learning model on static code features. Expert Systems with Applications 2020; 147:113156.

[8]. Huda, Shamsul, Sultan Alyahya, Md Mohsin Ali, Shafiq Ahmad, Jemal Abawajy, Hmood Al-Dossari, John Yearwood. A framework for software defect prediction and metric selection. IEEE access 2017; 6:2844-2858.

[9]. Zhang, Jie, Jiajing Wu, Chuan Chen, Zibin Zheng, Michael R. Lyu. CDS: A Cross–Version Software Defect Prediction Model with Data Selection. IEEE Access 2020; 8:110059-110072.

[10]. Malhotra, Ruchika, Megha Khanna. Dynamic selection of fitness function for software change prediction using particle swarm optimization. Information and Software Technology 2019; 112:51-67.

[11]. Ni, Chao, Xiang Chen, Fangfang Wu, Yuxiang Shen, and Qing Gu. An empirical study on pareto based multi-objective feature selection for software defect prediction. Journal of Systems and Software 2019; 152:215-238.

[12]. Wei, Hua, Changzhen Hu, Shiyou Chen, Yuan Xue, Quanxin Zhang. Establishing a software defect prediction model via effective dimension reduction. Information Sciences 2019; 477:399-409.

[13]. Huda, Shamsul, Kevin Liu, Mohamed Abdelrazek, Amani Ibrahim, Sultan Alyahya, Hmood Al-Dossari, Shafiq Ahmad. An ensemble oversampling model for class imbalance problem in software defect prediction. IEEE access 2018; 6:24184-24195.

[14]. Jayanthi, R., & Florence, L. (2019). Software defect prediction techniques using metrics based on neural network classifier. Cluster Computing, 22(1), 77-88.

[15]. Geng, Wang. Cognitive Deep Neural Networks prediction method for software fault tendency module based on Bound Particle Swarm Optimization. Cognitive Systems Research 2018; 52:12-20.

[16]. Feng, S., Keung, J., Yu, X., Xiao, Y., Bennin, K. E., Kabir, M. A., & Zhang, M. (2021). COSTE: Complexity-based OverSampling TEchnique to alleviate the class imbalance problem in software defect prediction. Information and Software Technology, 129, 106432.

[17]. Zhao, Linchang, Zhaowei Shang, Ling Zhao, Taiping Zhang, and Yuan Yan Tang. Software defect prediction via cost-sensitive Siamese parallel fully-connected neural networks. Neurocomputing 2019; 352:64-74.

[18]. Xu, Zhou, Jin Liu, Xiapu Luo, Zijiang Yang, Yifeng Zhang, Peipei Yuan, Yutian Tang, Tao Zhang. Software defect prediction based on kernel PCA and weighted extreme learning machine. Information and Software Technology 2019; 106:182-200.

[19]. Shao, Yuanxun, Bin Liu, Shihai Wang, Guoqi Li. Software defect prediction based on correlation weighted class association rule mining. Knowledge-Based Systems 2020; 105742.

[20]. Souri, Alireza, Amin Salih Mohammed, Moayad Yousif Potrus, Mazhar Hussain Malik, Fatemeh Safara, Mehdi Hosseinzadeh. Formal verification of a hybrid machine learning-based fault prediction model in the Internet of Things applications. IEEE Access 2020; 8:23863-23874.

[21]. Song, Q., Guo, Y., & Shepperd, M. (2018). A comprehensive investigation of the role of imbalanced learning for software defect prediction. IEEE Transactions on Software Engineering, 45(12), 1253-1269.

[22]. Cai, X., Niu, Y., Geng, S., Zhang, J., Cui, Z., Li, J., & Chen, J. (2020). An under-sampled software defect prediction method based on hybrid multi-objective cuckoo search. Concurrency and Computation: Practice and Experience, 32(5), e5478.

[23]. Yu, Q., Qian, J., Jiang, S., Wu, Z., & Zhang, G. (2019). An empirical study on the effectiveness of feature selection for cross-project defect prediction. IEEE Access, 7, 35710-35718.

[24]. He, H., Zhang, X., Wang, Q., Ren, J., Liu, J., Zhao, X., & Cheng, Y. (2019). Ensemble multi boost based on ripper classifier for prediction of imbalanced software defect data. IEEE Access, 7, 110333-110343.

[25]. Shippey, T., Bowes, D., & Hall, T. (2019). Automatically identifying code features for software defect prediction: Using AST N-grams. Information and Software Technology, 106, 142-160.

[26]. Zhou, T., Sun, X., Xia, X., Li, B., & Chen, X. (2019). Improving defect prediction with deep forest. Information and Software Technology, 114, 204-216.

[27]. Yu, Qiao, Shujuan Jiang, Junyan Qian, Lili Bo, Li Jiang, Gongjie Zhang. Process metrics for software defect prediction in object-oriented programs. IET Software 2020.

[28]. Song, Yuqi, Joseph Lindsay, Yong Zhao, Alireza Nasiri, Steph-Yves Louis, Jie Ling, Ming Hu, Jianjun Hu. Machine Learning-based prediction of no centrosymmetric crystal materials. Computational Materials Science 2020; 183:109792,2020.

[29]. Jin, C., & Jin, S. W. (2015). Prediction approach of software fault-proneness based on hybrid artificial neural network and quantum particle swarm optimization. Applied Soft Computing, 35, 717-725.

[30]. Tran-Ngoc, H., Khatir, S., De Roeck, G., Bui-Tien, T., & Wahab, M. A. (2019). An efficient artificial neural network for damage detection in bridges and beam-like structures by improving training parameters using a cuckoo search algorithm. Engineering Structures, 199, 109637.

[31]. https://www.kaggle.com/semustafacevik/software-defect-prediction