Journal of Engineering and Applied Science

RESEARCH Open Access

Hybrid feature selection method for predicting software defect

A. J. Anju[1*] and J. E. Judith[2]

*Correspondence:
ajanju1@gmail.com

[1] Department of Computer Science Engineering, Eranad Knowledge City Technical Campus, Manjeri, Kerala, India
[2] Department of Computer Science Engineering, Noorul Islam Center for Higher Education, Kanyakumari, Tamil Nadu, India

Abstract

To address the challenges associated with the abundance of features in software datasets, this study proposes a novel hybrid feature selection method that combines quantum particle swarm optimization (QPSO) and principal component analysis (PCA). The objective is to identify a subset of relevant features that can effectively contribute to the accuracy of a predictive model based on an artificial neural network (ANN). The quantum particle swarm optimization algorithm is employed to optimize the selection of features by simulating the behavior of quantum particles in a search space. This approach enhances the exploration and exploitation capabilities, allowing for a more effective identification of relevant features. Furthermore, principal component analysis is integrated into the hybrid method to reduce dimensionality and remove multicollinearity among features, thereby improving the efficiency of the feature selection process. The proposed hybrid method is applied to software defect datasets, where the selected subset of features is fed into an artificial neural network for defect prediction. The performance of the hybrid model is compared with traditional feature selection methods, standalone QPSO, and PCA. Experimental results demonstrate the effectiveness of the hybrid approach in achieving superior predictive accuracy while reducing the dimensionality of the dataset. The proposed approach not only enhances prediction accuracy but also provides a more interpretable and efficient subset of features for building robust defect prediction models.

Keywords: Software defect prediction, Feature selection, Optimization

Introduction

As technology advances, an increasing amount of data is generated and gathered. However, adding more features or dimensions to a model can actually decrease its accuracy due to the curse of dimensionality, where more data needs to be generalized. To combat this, dimensionality reduction is used to simplify a model and prevent overfitting. The two basic methods for dimensionality reduction are feature extraction and feature selection. In order to construct a more concise depiction of the data, feature selection includes choosing the most pertinent subset of characteristics from the input data. This method reduces the model's complexity by removing the extraneous features while keeping the ones that are most informative, either a supervised or an unsupervised features selection. Unsupervised feature selection only considers the input data, while supervised feature selection bases its selection criteria on the target variable. Inversely, feature extraction involves changing the initial data into a new feature space where the extracted features are more insightful and pertinent to the issue at hand. Dimensionality reduction frequently employs feature extraction methods like principal component analysis (PCA) and linear discriminant analysis (LDA). Whereas LDA maximizes the class separability of the transformed data, PCA preserves the maximum variance of the original data while transforming the input data into a lower-dimensional space.

Choosing the right dimensionality reduction technique has several benefits, such as decreasing calculation time, reducing the number of dimensions and storage space needed for data, improving data quality, and streamlining classification. One of the main benefits of DR is that it speeds up analysis and modelling by removing redundant or unneeded data from computations. Additionally, DR makes data storage and retrieval more efficient by lowering the number of dimensions and storage space required to store data. By lowering noise, removing outliers, and boosting the signal-to-noise ratio, DR also improves data quality, resulting in more accurate and trustworthy data. By simplifying the data, it also streamlines classification, making it simpler to categorize and predict results.

The dimensionality reduction procedure known as PCA is frequently utilized in a variety of industries, including data mining, image processing, and machine learning. By locating and extracting a data set's most important features, PCA, an unsupervised linear transformation technique, decreases the complexity of the data set. High-dimensional data sets are broken down into lower-dimensional representations by PCA, where each dimension corresponds to a primary component. The original features are combined linearly into these principal components, where each component accounts for the most variance in the data. In other words, a lower-dimensional illustration of the data is produced by PCA, which identifies the directions in which the data have the greatest variation and projects the data onto these directions.

The largest variance in the data is explained by the first principal component, the second principal component by the second maximum variance, and so on. Only the top principal components—those that explain the most variance—are kept after the principal components are arranged in descending order of the amount of variance they account for. The data is projected onto a new subspace with the same number of dimensions as the original subspace by PCA in order to reduce the dimensionality of the data. The top principal components—those that represent the most significant information in the data and so represent the new subspace—are chosen. Modelling and analysis can therefore be completed more quickly and with more accuracy because of the improved efficiency of the projected data in the new subspace.

In order to select the best features, it is required to assess each one's value and relevance to the desired variable or outcome. The dataset is then cleaned up, with just the most crucial and instructive features remaining after the removal of any unnecessary or redundant features. The process of choosing a subdivision of characteristics that are most pertinent for a certain activity, like classification, is known as feature subset selection. Finding the lowest subset of features that can distinguish across classes and improve prediction accuracy is the aim of feature subset selection. Filter methods, wrapper methods, and embedding methods are just a few of the different approaches that can be used to pick feature subsets. The reduction of data dimensionality, improvement of algorithmic accuracy, and reduction of computing cost of training and testing are all advantages of feature selection. Feature selection can streamline the processing and analysis of high-dimensional data while enhancing the effectiveness of machine learning algorithms by removing redundant and irrelevant features.

Related works

Xu et al. [1] have introduced KPWE, Kernel principal component analysis (KPCA), and weighted extreme learning machine (WELM) methodologies used to create the defect prediction system. KPWE seeks to collect representative data features in the initial stage. By using nonlinear mapping and the KPCA method, the original data is projected onto a latent feature space. KPWE seeks to address the class disparity in the second stage. It makes use of the WELM method to train a reliable model for predicting defects using a weighting-based system.

Miholca et al. [2] have recommended the purpose of predicting software defects. In this paper created the innovative supervised classification technique HyGRAR in this study. HyGRAR is a nonlinear hybrid prototypical that uses ANN and gradual solitary association rule mining to distinguish between software objects that are substandard and those that are not. It makes obvious sense to assume that there are correlations between important software measurements, and values may be important for showing insecurity to flaws when using the HyGRAR classifier. Furthermore, because of their adaptability and expressive strength, it is believe that discovering their linkages is preferable than predefining them.

Oluwagbemiga B. A. et al. [3] have developed a hybrid multi-filter wrapper feature selection approach for the prediction of software defects. It remains an outstanding research question how to hybridize filter and wrapper-based feature selection. This is done in order to preserve excellent performance, a generalizable solution, and concurrently minimal computational cost. It is anticipated that the suggested hybrid feature technique will be able to choose pertinent and irrefutable features from SDP datasets, enhancing the predictive capability of SDP models.

A different adaptive rank aggregation-based ensemble multi-filter feature selection (AREMFFS) method has been proposed by Balogun, A. O. et al. in this publication. The AREMFFS strategy that has been proposed is based on comparing and integrating the benefits of various FFS techniques by combining multiple rank lists during the creation and selection of the highest-ranked features to be used in the SDP process. On defect datasets from multiple repositories with different levels of defect granularity, the proposed AREMFFS technique is evaluated using decision tree (DT) and Naive Bayes (NB) models. The experimental results showed that AREMFFS outperformed various baseline FFS approaches that were evaluated, current rank aggregation-based multi-filter FS methods, and variants of AREMFFS proposed in this work.

Mumtaz, B. et al. [4] have developed a hybrid strategy to deal with the problem of software module flaw detection. The strategy coupled benchmark machine-learning classifiers with the Optimised Artificial Immune Networks (Opt-aiNet) algorithm's feature selection functionality. Five open-source National Aeronautics and Space

Administration (NASA) datasets from the PROMISE repository were used to test and validate the methodology: CM1, KC2, JM1, KC1, and PC1.

Balogun, A. O. et al. [5, 6] have recommended an enhanced wrapper feature selection (EWFS) technique that iteratively and dynamically chooses features. The enhanced WFS (EWFS) method that has been presented works by incrementally choosing features while taking into account features that have already been chosen. The innovation of EWFS is based on the improvement of the subset evaluation process of WFS methods through the use of a dynamic re-ranking strategy that iteratively picks relevant features with a low subset evaluation cycle without sacrificing the performance of the resulting model's prediction. On software defect datasets with different levels of granularity, EWFS was implemented along with decision tree (DT) and Naive Bayes classifiers for evaluation.

Bala, Y. Z. et al. [7] have offered a transformation and feature selection strategy to solve the difficulty of high-dimensional characteristics in cross-project defect identification and to lessen the distribution disparity. The study used datasets that are freely available from the AEEEM repository to conduct a comparative experiment. Based on the commonly used performance evaluation parameter, F1_score, the results analysis showed that the proposed strategy, in combination with the random forest classifier, beats four other cutting-edge cross-project defect prediction approaches.

Lin, J. and Lu, L. [8] have proposed a system known as semantic feature learning via dual sequences (SFLDS), which used the abstract syntax tree (AST) to capture both semantic and structural information for feature development. They chose sample AST nodes and created an S-AST (simplified AST) from the program source code. To capture the semantic and structural information of the S-AST, their method included two sequences: one was the output of traversing the S-AST node in pre-order, and the other was made up of parent nodes. The dual sequences' tokens were each stored as a numerical vector using word embedding and mapping methods. Finally, they used a neural network built on a bidirectional long short-term memory (BiLSTM) to automatically create semantic characteristics for software defect prediction (SDP) from the dual sequences.

Architecture of proposed hybrid feature selection method for softer defect prediction

The key goal of this research analysis is to progress the precision of SDP by introducing a groundbreaking methodology called hybrid feature selected artificial neural network (HF-ANN), which combines artificial neural networks (ANN) with hybrid feature selection method (HFSM). The core of this methodology is to offer a dependable and inexpensive prediction process. The development of the HF-ANN methodology involved an extensive analysis of various research articles and methodologies utilized in contemporary software defect prediction. The researchers determined the need for an enhanced prognostic process that can surmount the challenges faced by traditional methods by analyzing the potential and inadequacies of existing approaches. The HF-ANN approach is notable for its distinctive amalgamation of artificial neural networks and HFSM (i.e., combination of principal component analysis and quantum particle swarm optimization models). Artificial neural networks are computational constructs that emulate the configuration and operation of the human brain. These networks are particularly adept at discerning intricate relationships and patterns within data, rendering them applicable for software defect prognostication. The following Fig. 1 is the overall framework for feature selection.

Data acquisition for defect prediction

The dataset used for the experiments was obtained from the NASA PROMISE open-source dataset repository. Two datasets from the PROMISE repository named CM1, JM1, KC1, and KC2 with different attributes were used. Both JM1 and KC1 have highest number of instances.

Preprocessing

Data preprocessing involves converting unprocessed data into an understandable form that the algorithms can simply comprehend and use. Pre-processing frequently involves a variety of stages, including cleaning the data by eliminating or correcting errors, addressing missing data, scaling or normalizing numerical characteristics, encoding categorical variables, and lowering the dimensionality of the data.

A method for transforming data into a standard format that can be easily studied and compared is data normalization. Reducing data redundancy and streamlining the dataset to make it easier to use is one of the key goals of data normalization. Data insertion, deletion, and updating are a few of the methods that can be used to normalize data. For instance, dividing a huge table into smaller tables that contain related data is a typical normalization strategy. To do this, data must be added to new tables, redundant data must be removed from the original database, and the original table must be updated to relate to the new tables. The dataset can be made simpler through normalization, making it simpler to carry out further data analysis activities like statistics or machine learning. By lessening the influence of redundant or unnecessary data, normalization can also increase the accuracy and performance of these tasks.

Data normalization has the following benefits over alternative methods:

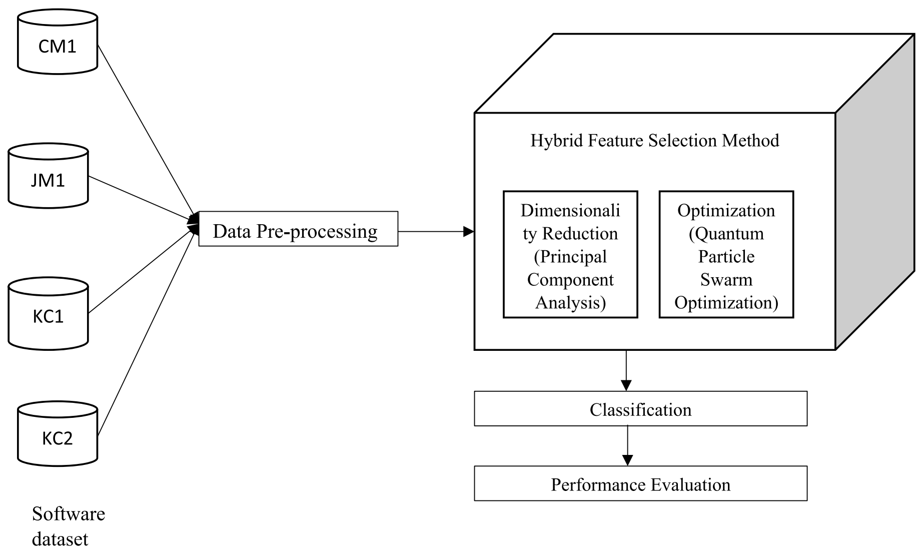
CM1
JM1
KC1
KC2
Software dataset
Data Pre-processing
Hybrid Feature Selection Method
Dimensionality Reduction (Principal Component Analysis)
Optimization (Quantum Particle Swarm Optimization)
Classification
Performance Evaluation

Fig. 1 Overall layout of proposed hybrid feature selection method

- *Improved accuracy*: Normalization can increase the accuracy of subsequent data analysis operations by removing redundancy and streamlining the data.
- *Improved performance*: By lowering the quantity and complexity of the dataset, normalization can enhance the performance of data analysis operations.
- *Fair comparisons*: As it eliminates bias resulting from variations in scale or the range of data values, normalization can help with fair comparisons across various variables.
- *Better data quality*: Normalization can assist in finding and fixing data flaws, producing higher-quality data for further analysis.

The normalized data is preprocessed using the Box-Cox transformation. Trained data is given for preprocessing, S_i^{train}. The data must be transferred into another format during this pre-processing phase so that it can be understood. Preprocessing is mostly done to get rid of information that has to increase the data's quality; fictional parameters are used. Additionally, the Box-Cox transformation models in pre-processing stage are turning the mismatched data into a format that is compatible with software. The ideal parameters are used in this model, which follows the normal distribution. Because of its conversion property, the data pre-processing stage can profit more. It is possible to express the Box-Cox transformation given as in Eq. 1:

$$S_i^{pre} = \begin{cases} \frac{I_n^j - 1}{j}; j \neq 0 \\ 1n(I_n); j = 0 \end{cases} \tag{1}$$

The pre-processed data in this instance is denoted by S_i^{pre}. The input data is indicated by the symbol I_n. j can be used to express the undefined parameter. However, the condition will be fulfilled, and the correct translation will occur if $I_n > 0$. The value of j determines how the input data is transformed to the standard distribution. With the aid of the probable estimation approach, j value is determined. These pre-processed data are supplied into the feature extraction procedure as input in order to extract further features. The pre-processed data for these common patterns is indicated by the symbol S_i^{pre}.

Proposed hybrid feature selection method For SDP

Hybrid feature selection method, emphasizing its potential impact on software defect prediction accuracy and efficiency. Combination of multiple techniques leverages their strengths and addresses their limitations.

a) Dimensionality reduction using PCA

Data pre-processing is necessary before using PCA for feature dimensionality reduction to make sure the data is in an analysis-ready format. Dimensionality reduction is a method for reducing the number of dimensions in high-dimensional data while preserving the important information. Using fewer parameters that match the data's inherent dimensionality, the goal is to capture the substance of the data. This is crucial because high-dimensional data may be plagued by the phenomenon known as the "curse of dimensionality," which describes how the number of likely feature combinations grows exponentially as the number of dimensions rises. This may make it challenging to evaluate, display, and comprehend the data and may also cause machine learning models to overfit. The number of parameters required to describe the data with no information loss is known as the intrinsic dimensionality of the data. Since certain dimensions could be redundant or unnecessary, this is not always the same as the total number of dimensions in the data. Since many of the pixels in a collection of photographs of faces may be background or unrelated to the attributes that are being evaluated, the inherent dimensionality may, for example, be significantly smaller than the number of pixels in the dataset.

Dimensionality reduction can be helpful for compression and feature selection in addition to making high-dimensional data easier to visualize and analyze. Data storage, transmission, and computing speed can all be improved by lowering the data's dimensionality. Also, it is feasible to develop machine learning models that are more effective and less prone to overfitting by determining the key features in the data.

High-dimensional data are transformed into a lower-dimensional representation using the well-liked linear dimensionality reduction method known as PCA. In order to project the data onto these directions and produce a lower dimensional representation, PCA is particularly helpful for determining the directions in which the data fluctuates most. The goal is to capture as much variance in the data as possible with fewer parameters, which can assist to reduce the dimensionality curse and other unfavorable characteristics of high-dimensional spaces. The main goal of PCA is to identify a linear transformation that converts the high-dimensional data from the original system to a new coordinate system with a maximum variance for the data along each axis. The eigenvectors and eigenvalues of the data's covariance matrix are found in order to perform this transformation. The data's largest directional variations are represented by the eigenvectors, while each direction's variance is represented by the eigenvalues.

Tasks like data cleaning, normalization, and standardization may fall under this category. Data cleaning is eliminating or fixing any flaws or discrepancies from the data. To assist the PCA method perform better, normalization entails scaling the data so that it has a zero mean and a standard deviation of 1. Standardization, which entails scaling the data so that it has a range of values between 0 and 1, can also aid in enhancing the PCA algorithm's performance. The eigenvalues and eigenvectors of the covariance matrix of the feature vector can be determined using the PCA algorithm after the data has been preprocessed. The feature vector is then created with a new, reduced dimensionality that retains 99% of the variance using the eigenvectors with the biggest eigenvalues. The feature vector's dimensionality can be successfully reduced while still preserving the most crucial aspects of the data by doing data preprocessing and using the PCA algorithm.

Similar to supervised learning algorithms, normalization or feature scaling depending on the n-dimensional training set $x^{(1)}, x^{(2)}, x^{(3)}, \ldots\ldots x^{(n)}$. Equation (2) is used to compute the mean of each feature.

$$\mu_i = \frac{1}{n}\sum_{j=1}^{n} x_i^{(j)} \tag{2}$$

Given that each feature has a mean value of exactly zero, then it is replaced each x_i value with its $x_i - \mu_i$ value. If distinct characters have different mean values, however, it can scale them so that they fall within a similar range. This scaling procedure of the i th element in supervised learning is specified by Eq. 3, where s_i is the $|max - mean|$ value or the static deviation of the i th feature.

$$x_i^{(j)} = \frac{x_i^{(j)} - \mu_i}{s_i} \tag{3}$$

Equation 4 is used to calculate the covariance matrix, which has the dimensions because the $x^{(j)}$ vector has $N \times 1$ dimensions and the $\left(x^{(j)}\right)^T$ has $1 \times N$ dimensions. The covariance matrix's eigenvalues and eigenvectors, which correspond to the feature vectors' new magnitudes in the transformed vector space and their accompanying directions, are next calculated. While working with the covariance matrix, the eigenvalues quantify the variance of each vector. When an eigenvector has high valued eigenvectors, it has a high variance and contains a variety of crucial dataset information. Eigenvectors, on the other hand, have tiny eigenvalues and relatively little information about the dataset.

$$covariancematrix = \frac{1}{N}\sum_{j=1}^{N} x^{(j)} \times \left(x^{(j)}\right)^T \tag{4}$$

PCA is frequently utilized in a variety of disciplines, including social sciences, finance, biology, and computer vision. PCA is used in computer vision for object detection, face recognition, and image reduction. PCA is employed in finance for credit rating, risk management, and portfolio optimization. PCA is utilized in biology for the investigation of protein structure, gene expression, and drug development. For survey analysis, market research, and opinion polling in the social sciences, PCA is employed. Compared to other dimensionality reduction methods, PCA has a number of benefits. It is simple to implement and computationally effective. Moreover, it is resistant to data noise and outliers. Since PCA is a linear method, it is simple to understand and explain the findings. In order to find patterns and relationships in the data, it can be utilized to depict the data in a lower-dimensional space.

PCA, however, have some restrictions. It is predicated on the notion that the data are normally distributed and linearly correlated. Very nonlinear or non-Gaussian data may not respond well to it. Moreover, PCA is sensitive to the data scaling and may not perform effectively if the features are on various scales. Last but not least, PCA does not offer a probabilistic model of the data and might not be capable to extract all of the pertinent info from the data.

The process of choosing a subset of pertinent features from a larger set of features to include in a model is known as FS. The objectives of feature selection are to minimize the degree of dimensionality of the input data and to choose just those features that are essential for the current problem. The model is made easier to understand, more interpretable, and less prone to overfitting by lowering the amount of input variables. When a model fits training data too tightly and is overly complex, overfitting occurs, leading to inferior performance on new data. The model can concentrate on the most crucial aspects that are pertinent to the issue by reducing unnecessary or redundant information through feature selection, which enhances its generalization performance on fresh data.

Feature selection is represented by several terminologies such as attribute, variable subset, and variable selection. This method selects a selection of relevant features or necessary predictors for model development. The following are the reasons for adopting feature selection techniques:

- ❖ To build a model that is straightforward and easy to interpret for academics and users
- ❖ Training times are limited.
- ❖ The curse of dimensionality can be avoided by eliminating overfitting concerns.

The feature selection method can remove redundant or irrelevant characters in data that cause information loss. The terms redundant and irrelevant are used interchangeably. If there is another relevant trait with which it is closely associated, the relevant feature is said to be redundant. Feature extraction must be distinguished from feature selection. Feature extraction generates new features by merging the original features or by combining the functions of the original features, whereas feature selection yields only a subsection of the features. FS strategies are frequently utilized in areas where features outnumber sample data points. After testing the features, we had to evaluate each subset of features and finally select the one with the lowest error rate.

b) Quantum particle swarm optimization-based feature selection

A metaheuristic optimization algorithm called quantum particle swarm optimization (QPSO) is influenced by both PSO and quantum mechanics. In order to discover the search space and identify the best answer, QPSO employs a probabilistic methodology. Each particle in the swarm is represented by a quantum state vector in QPSO, which is used to indicate the likelihood that the particle will be found in a specific state. The best answer revealed by the swarm and the correct solution discovered by the particle itself are used to update the state of each particle. The updating procedure is based on the superposition and entanglement concepts found in quantum physics.

Many optimization issues, such as feature selection, image processing, and clustering, have been solved using QPSO. It has been demonstrated to be successful in resolving issues with numerous variables or a complicated search space. The ability of QPSO to escape local optima and converge to the global optimum is one of its benefits. The probabilistic methodology and application of quantum mechanics principles enable this. Moreover, QPSO has a strong balance between exploration and exploitation, which enables it to efficiently search the search space and quickly identify the ideal answer.

The output of the pre-processed data S_i^{pre} is used as the input for the feature selection step, which involves choosing the pertinent and necessary feature from the pre-processed SDP data. The quantum theory-based PSO method may be used to choose features. This paradigm can do the search operation for the feature subset using the various parameters. Additionally, each state of the space represents a subset of features. Consider a space in which there are a total number of bits (w) and a total number of features (v). Each bit of the state represents a value, and if it is signified as "1," "1" denotes the presence of features, while "0" indicates their absence. The performance can be added or removed effectively because the space operators describe the connections between the states. The state space, which consists of many evaluation functions, is produced using the heuristic function. The best optimization technique for feature selection that has been proposed in this research is based on QPSO.

Traditional PSO algorithm achieves a solution by utilizing the swarm's searching characteristics and performing continuous operations. A feature subset of software fault data is regarded to be the number of swarms in the search space in this case. The optimization process determines whether each swarm and its location are a workable option. The fitness function is assessed based on each function, such as a swarm's distance and direction. To control it, the concepts of trajectory and speed are applied. $Y_j^u = \left(y_{j1}, y_{j2}, \ldots, y_{jE}\right)$ at the $u - th$ iteration represents the localization of features in the feature space. The particle's velocity is sometimes written as $V_j^u = \left(v_{j1}, v_{j2}, \ldots, v_{jE}\right)$. In this case, the value of j is between $1andN$. The particles identified as the $pbest_j = \left(Q_{j1}, Q_{j2}, \ldots, Q_{jE}\right)$ for detecting the local optimal position of the PSO framework's local search space. The best particle in the global search space is then reported by the optimization method, which is denoted by the expression $gbest = (Q_{h1}, Q_{h2}, \ldots, Q_{hE})$. Equations (5) and 6) are used to mathematically denote the method, which changes the particles' position and velocity during each iteration:

$$v_{je}^{u+1} = w \times v_{je}^{u} + d_1 \times rnd \times \left(Q_{je} - y_{je}^{u}\right) + d_2 \times rnd \times \left(Q_{he} - y_{je}^{u}\right) \tag{5}$$

$$y_{je}^{u+!} = y_{je}^{u} + v_{je}^{u+1} \tag{6}$$

Here, $e = 1, 2, \ldots, E$. The dimension of the search space is similar to the term E. Predetermined constant term of inertia weight denoted by the symbol w. Similarly, by designating the acceleration function, the constant term is specified by d_1 and d_2. rnd, which is reliably generated in the range of 0 to 1, is used to signify the arbitrary number id. The PSO approach often fails to sustain an optimal location in a large dimension space and only achieves a limited degree of convergence. To solve these problems, the PSO model applies quantum theory. Due to the quantum theory technique's extended particle search that locates the whole viable solution space, the QPSO framework's global search performance is substantially superior than that of the conventional PSO method. The optimization strategy in this case will exclude velocity and position data. Instead, Eqs. (7–9) provide a mathematical description of the iterative updating process of particles:

$$r_j = \vartheta \times pbest_j + (1 - \vartheta) \times gbest \tag{7}$$

$$nbest = \frac{1}{N}\sum_{j=1}^{N} pbest_j = \left(\frac{1}{N}\sum_{j=1}^{N} Q_{j1}, \frac{1}{N}\sum_{j=1}^{N} Q_{j2}, \ldots, \frac{1}{N}\sum_{j=1}^{N} Q_{jE}\right) \tag{8}$$

$$Y_j^{u+1} = r_j \pm \alpha \times \left|nbest - Y_j^u\right| \times ln\left(\frac{1}{f}\right) \tag{9}$$

Here, the arbitrary values are defined by ϑ, and f are uniformly distributed from 0 to 1. The average ideal position of each particle in the population is denoted by the number *nbest*. Every particle's local attractor is calculated using its *gbest* and *pbest* values, and it is written as $r_j = \left(r_{j1}, r_{j2}, \ldots, r_{jE}\right)$. The word "$\alpha$" stands for the increasing coefficient. The QPSO-FS framework's F_i^{opt} indicator is used to choose the best features after implementing these methods. The ANN classifier is then supplied the chosen features F_i^{opt} as an input for further forecasting flaw and non-defect data.

The following are the steps for hybrid feature selection model:

- *Step 1*: Identify the QPSO's starting parameters, such as the amount of particle swarms and the parameter range.
- *Step 2*: To lower the dimension of the dataset, PCA technique is employed.
- *Step 3*: In QPSO, define the fitness function.
- *Step 4*: Iterating the fitness function updated the local optimal value *pbest* and the global optimal value *gbest* for each particle.
- *Step 5*: Update each particle's new position and determine the particle swarm's ideal position.
- *Step 6*: Identify the final state. If not, go to Step 3 until the optimal search has completed the maximum number of iterations.
- *Step 7*: Get the optimal data.

c) Artificial neural network classifier for software defect prediction

ANN can simulate intricate interactions between inputs and outputs and is built to learn from feature selected data. After feature selection, the data is given for classification. Interconnected nodes that make up ANNs are arranged into three layers: input, hidden, and output. An input feature is represented by each node in the input layer, and an output feature is represented by each node in the output layer. Nodes required to model the intricate connections between the input and output features are included in the hidden layer.

The input, hidden, and output are the three main components of the ANN classifier, which is shown in given Fig. 2. The input layer receives the input data, which the hidden layer(s) subsequently processes to separate out the features that are important for the classification task. The classification outcome is provided by the output layer based on the features that were extracted. In order to diminish the discrepancy between the expected and actual output, the weights between the network nodes are changed throughout the training phase. This is accomplished by estimating the error between the expected and actual outputs after comparing them. The weights in the network are then modified in order to minimize the error using the data from the error. After trained, the network can be utilized to categories fresh incoming data. The network receives the incoming data and processes it to identify the necessary features. The classification outcome is subsequently presented by the output layer using the features that were extracted.

By using the ANN model, a response is built by transferring weights to indicate the precise link between inputs and outputs. Training data made up 70% of the total, and

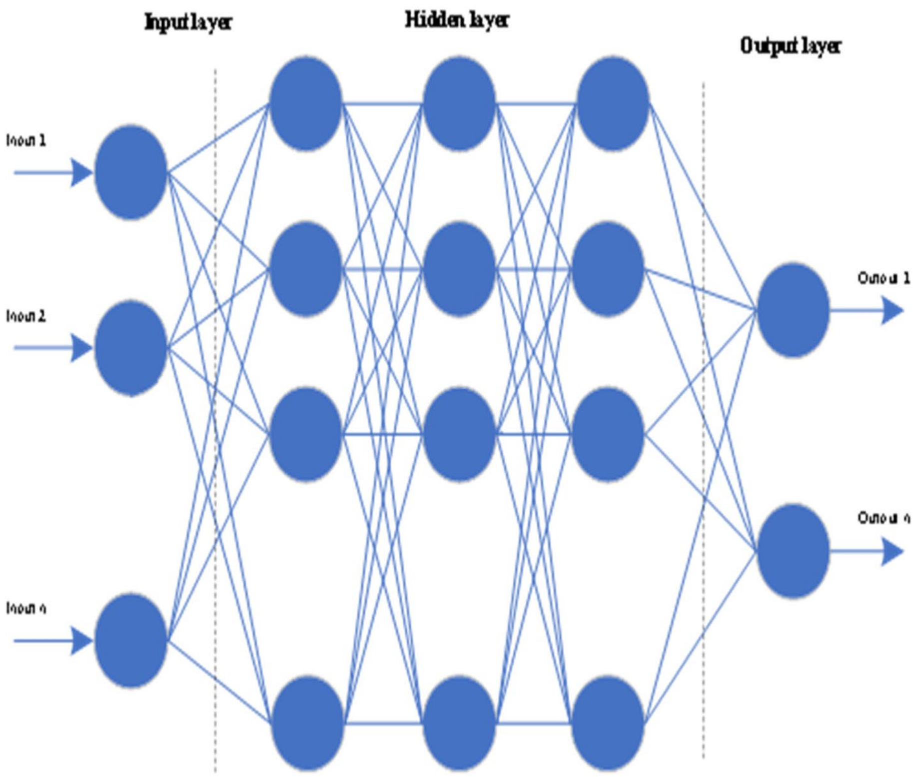

Fig. 2 Illustration of the execution process of ANN

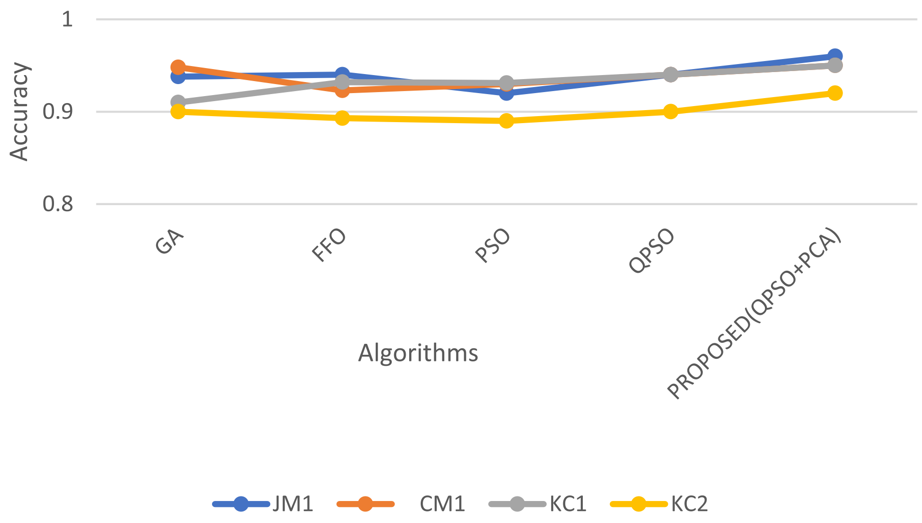

Fig. 3 Accuracy of hybrid feature selection model for different datasets

testing data made up 30%. The ANN model can be trained using the training dataset. In the Testing dataset, which provides evaluation of the ANN model's predictive skills. The ANN is effectively broadcasting the destination between the input and output layers throughout the training test. Figure 3 clearly depicts the ANN execution method.

The pseudocode of proposed hybrid feature selection method for SDP is given below:

```
------------------------------------------------------------------------------------------------------------------
Initialize:
  Initialize QPSO parameters (enumber of particles, maximum iterations, etc.)
  Initialize ANN architecture (number of layers, neurons per layer, activation functions, etc.)
  Initialize PCA parameters (number of principal components to keep)
  Initialize particle positions (representing weights and biases of ANN and PCA components)
  Initialize particle velocities
  Evaluate initial fitness of particles using ANN and PCA
Main loop:
  For iteration in range(max_iterations):
    For particle in particles:
      Update particle velocity using Equation 7
      Update particle position based on velocity
      Clip particle positions to valid ranges (−1 to 1 for weights and biases, adjust PCA components)
      Apply PCA to the particle's weights and biases
      Evaluate fitness of the particle using Equation 9
      Update personal best position and fitness of the particle
      Update global best position and fitness if necessary
  Update QPSO parameters (update inertia weight, acceleration coefficients, etc.)
Finalize:
  Return the best particle (weights, biases, and PCA components) as the optimized solution for the ANN with PCA
```

Results and discussion

Experimental environment

The study is carried out on SDP dataset with two classes that is publicly available on the NASA software repository online site. It is a new open-source standard that will be used to test the proposed automated detection system. This analysis makes use of the NASA dataset, which has 10,885 instances and 22 attributes. Table 3 shows the software defect prediction using NASA datasets. The NASA dataset was divided into 20% testing and 80% training. The training phase took 103.86432 s, and the SDP method took 9.71 s. Furthermore, the predictive accuracy is proven by comparing the proposed technique's evaluation metrics to those of existing methodologies. The confusion matrix obtained from the experimental findings is used to generate the evaluation matrices.

Dataset description

The description of the datasets used for experiment are given in the following paragraph. These datasets are taken from NASA PROMISE software repository. These datasets are used to train software defect prediction models. Each datasets contain a number of software modules, and each module can accept input from a wide range of quality indicators. The numerous software attributes that these quality metrics commonly measure include complexity, size, and maintainability.

i. Characteristics of datasets used for SDP

Different parameter descriptions of the datasets under consideration are given in Table 1, and it contains several instances, number of attributes, designed languages, and percentage of defect and non-defect data.

These datasets can be used in test develop and predictive models, which can subsequently be applied to find potentially problematic software modules. The ultimate objective of software defect prediction is to raise software quality by finding and correcting errors before they impact end users.

ii. Attribute description for NASA PROMISE dataset

The dataset consists of a collection of attributes related to software modules and their associated defects. These attributes are used to train and test defect prediction models to identify whether a software module is likely to contain defects or not. Each dataset consists of 22 attributes. Table 2 shows the attribute description of NASA datasets.

Performance metrics

This section explains the performance metrics for the suggested model, some of which have already undergone validation and testing. "True positive (TP), false positive (FP), true negative (TN), and false negative (FN)" measures were employed in the evaluation. TP is the number of correctly classified faulty modules.

- TN is the perfectly detected non-defective module.
- FP is the number of correctly categorized non-defective modules.
- FN is the number of incorrectly classified non-defective modules.

The following mathematical expression describes the analysis of the performance measures taken into account in the planned and past analyses:

Table 1 Characteristics of NASA PROMISE software defect datasets

Datasets	Number of instances	Number of attributes	Language	Defect	Non-defect
CM1	498	22	C	49	449
JM1	10,885	22	C	8779	2106
KC1	2109	22	C++	326	1783
KC2	522	22	C++	105	415

Table 2 Attribute description of NASA PROMISE datasets for SDP

Attributes	Data type	Abbreviations
Branch count	Numeric	The flow graph
total_Opnd	Numeric	Total operands
total_Op	Numeric	Total operators
Defects	{False, true}	Reported defects
Loc	Numeric	Line count of code
ev (g)	Numeric	Essential complexity
v (g)	Numeric	Cyclomatic complexity
iv (g)	Numeric	Design complexity
D	Numeric	Difficulty
E	Numeric	Effort
I	Numeric	Intelligence
N	Numeric	Total operators + operands
L	Numeric	Program length
V	Numeric	Volume
lOBlank	Numeric	Count of blank lines
Uniq Op	Numeric	Unique operators
lOCodeAndComment	Numeric	Line count of comments
B	Numeric	Blank
lOComment	Numeric	Count of lines of comments
T	Numeric	Time estimator
uniq_Opnd	Numeric	Unique operands
lOCode	Numeric	Line count

❖ Prediction accuracy

Accuracy is a quantitative measure that typically characterizes the model's overall performance across all categories. It proves advantageous when all categories hold equal significance. The ratio of accurate predictions to the total number of predictions determines its computation. It is formulated as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

❖ Specificity

The metric referred to as the true-negative rate is recognized for evaluating the proportion of negative outcomes that are accurately detected. It is measured as follows:

$$Specificity = \frac{TN}{TN + FP} \tag{11}$$

❖ Sensitivity

It is known as true-positive rate, measuring the proportion of positives which are correctly identified.

$$Sensitivity = \frac{TP}{TP + TN} \tag{12}$$

❖ Precision

The precision metric is determined through the calculation of the ratio between the total number of samples classified as positive, including both correctly and incorrectly classified samples and the number of positive samples that have been accurately classified. This measure evaluates the accuracy of the model in identifying a sample as positive.

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

❖ Recall

The recall metric is derived from the division of correctly classified positive samples by the total number of positive samples. It serves as an indicator of the model's capacity to identify positive samples. An increase in recall value denotes an increase in the detection of positive samples.

$$Recall = \frac{TP}{TP + FN} \tag{14}$$

❖ F-measure

It is the harmonic mean of precision and recall

$$F - Measure = \frac{2 \times precision \times recall}{Precision + recall} \tag{15}$$

❖ False-positive rate (FPR)

A false positive refers to an instance where the model inaccurately predicts the positive class.

$$FPR = \frac{FP}{FP + TN} \tag{16}$$

❖ False-negative rate (FNR)

A false negative refers to an event in which the model makes an inaccurate prediction of the negative class.

$$FNR = \frac{FN}{TP + FN} \tag{17}$$

Performance analysis of proposed hybrid feature selection method for SDP

a) Analysis of prediction accuracy

Table 3 Accuracy of proposed hybrid feature selection method for different datasets

Model/datasets	JM1	CM1	KC1	KC2
GA	0.938	0.948	0.91	0.9
FFO	0.94	0.923	0.932	0.893
PSO	0.92	0.93	0.931	0.89
QPSO	0.94	0.94	0.94	0.9
Proposed (QPSO + PCA)	**0.96**	**0.95**	**0.95**	**0.92**

Table 4 Precision of hybrid feature selection model for different datasets

Model/datasets	JM1	CM1	KC1	KC2
GA	0.93	0.9	0.92	0.9
FFO	0.89	0.86	0.93	0.88
PSO	0.91	0.89	0.931	0.9
QPSO	0.94	0.91	0.94	0.905
Proposed (QPSO + PCA)	**0.96**	**0.95**	**0.97**	**0.915**

Table 3 lists the accuracy results from assessments utilizing various feature selection methods on various datasets. The proposed hybrid feature selection model has the maximum accuracy across all datasets, as shown by the results. The suggested model's accuracy is 1% better than the second-best model for the KC1 dataset, 1% better than the second-best model for the CM1 dataset, 2% better than the second-best model for the KC2 dataset, and 2% better than the third-best model for the JM1 dataset. The accuracy analysis of the suggested and current models is shown in Fig. 3.

b) Precision

Table 4 lists the precision that was determined through assessments utilizing hybrid feature selection models on various datasets. The suggested hybrid feature selection model has the maximum precision across all datasets, as shown by results. This indicates that the proposed model has the highest accuracy in correctly predicting software defects. The second highest model for the jm1 dataset, the second highest model for the CM1 dataset, the second highest model for the KC1 dataset, and the second highest model for the KC2 dataset are all outperformed by this metric by 2%, 4%, 3%, and 1%, respectively. The precision of the proposed and existing models is shown in Fig. 4.

c) Recall (sensitivity or true-positive rate)

Table 5 lists the recall results from procedures utilizing hybrid feature selection models on various datasets. The suggested hybrid feature selection model has the highest recall for all datasets, as can be observed from the results. This suggests the proposed model has the highest recall in correctly predicting software defect. This measure implies the 2.2% greater than the second highest model for the jm1 dataset, 2.5% greater than the second highest model for the CM1 dataset, 1% greater than the second highest model for the KC1 dataset, and 1% greater than the second highest model for the KC2 dataset. Figure 5 visualizes the recall of the proposed and the existing models.

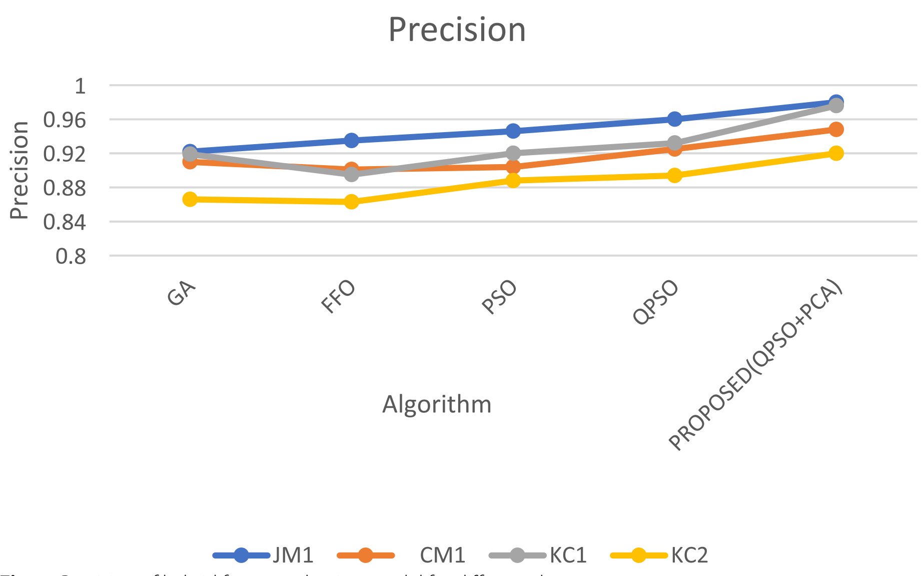

Fig. 4 Precision of hybrid feature selection model for different datasets

Table 5 Recall of hybrid feature selection model for different datasets

Model/datasets	JM1	CM1	KC1	KC2
GA	0.93	0.91	0.9	0.92
FFO	0.92	0.92	0.89	0.9
PSO	0.91	0.929	0.89	0.9
QPSO	0.941	0.93	0.9	0.93
Proposed (QPSO + PCA)	**0.97**	**0.94**	**0.93**	**0.94**

d) F-measure

The F-measure obtained from the experiments for different datasets using hybrid feature selection models is tabulated in Table 6. It can be seen from the results that the proposed hybrid feature selection model has the highest F-measure for all datasets. This means that the percentage of correctly predicted software defect is highest for proposed model. This measure implies the 1% greater than the second highest model for the jm1 dataset, 2% greater than the second highest model for the CM1 dataset, 2% greater than the second highest model for the KC1 dataset, and 1% greater than the second highest model for the KC2 dataset. Figure 6 visualize the F-measure of the proposed and the existing models.

Conclusions

In this chapter, the emphasis is on effective preprocessing techniques that will increase the accuracy of software defect analysis. These techniques include data normalization and data transformation with HFS-ANN method, i.e., it contains dimensionality reduction using PCA, FS using quantum theory-based particle swarm optimization (QPSO) and classified using ANN classifier. Preparing the data for analysis by addressing problems such varied scales, outliers, and irrelevant features is the goal of the preprocessing

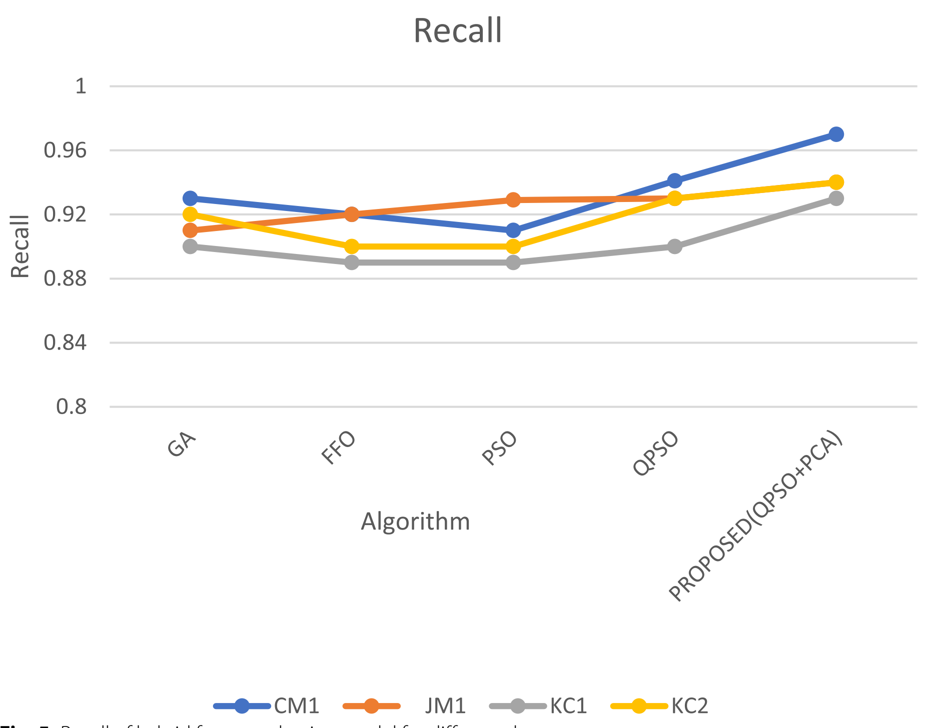

Fig. 5 Recall of hybrid feature selection model for different datasets

Table 6 F-measure of hybrid feature selection model for different datasets

Model/datasets	JM1	CM1	KC1	KC2
GA	0.895	0.936	0.9	0.866
FFO	0.88	0.767	0.88	0.863
PSO	0.91	0.89	0.9	0.90
QPSO	0.93	0.9	0.92	0.91
Proposed (QPSO + PCA)	**0.94**	**0.92**	0.94	**0.92**

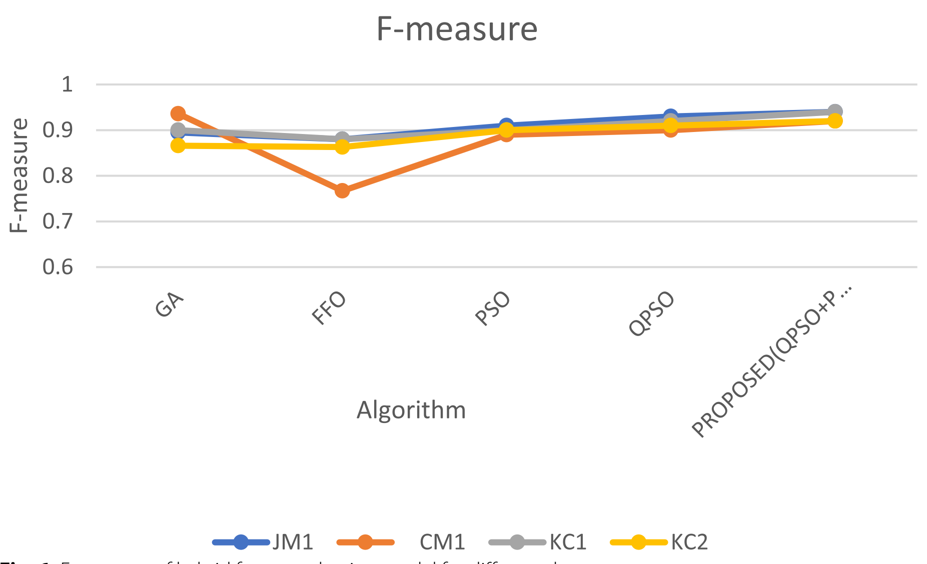

Fig. 6 F-measure of hybrid feature selection model for different datasets

stage. Another key preprocessing step is reducing the number of dimensions, and PCA is the method used to do this. The feature space's dimensionality is decreased through PCA, but the most important data is kept. It accomplishes this by converting the initial features into a fresh set of principal components, which are orthogonal variables. The suggested strategy is then contrasted with other strategies used in the field. Accuracy, detection error, specificity, and sensitivity are some of the evaluation factors used to compare the performance of the suggested strategy to existing methods. The evaluation's findings show that, in terms of precision and predicting accuracy for software defect analysis, the suggested solution outperforms current approaches. The suggested method successfully addresses the difficulties associated with software defect analysis and produces more precise and trustworthy results by utilizing HFS-ANN. This approach can be used by academics and industry professionals to improve the efficiency of software defect analysis and produce more accurate forecasts.

Abbreviations

PCA	Principal component analysis
SDP	Software defect prediction
QPSO	Quantum theory-particle swarm optimization
ANN	Artificial neural network
LDA	Linear discriminant analysis
KPCA	Kernel principal component analysis
FDNB	Feature dependent Naive Bayes
WELM	Weighted extreme learning machine
AREMFFS	Aggregation-based ensemble multi-filter feature selection
EWFS	Enhanced wrapper feature selection
SFLDS	Semantic feature learning via dual sequences
AS	Abstract syntax tree
HF-ANN	Hybrid feature selected artificial neural network

Acknowledgements
I would like to express my deep gratitude to Professor J. E., Judith, my research supervisor, for her patient guidance, enthusiastic encouragement, and useful critiques of this research work.

Authors' contributions
Confirms sole responsibility for the following: study conception and design, data collection, analysis and interpretation of results, and manuscript preparation.

Funding
The authors state that this work has not received any funding.

Availability of data and materials
Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Ethical approval and consent to participate
Institutional Review Board approval was not required.

Competing interests
The authors declare that they have no competing interests.

Received: 17 September 2023 Accepted: 11 May 2024
Published online: 30 May 2024

References

1. Xu Z, Liu J, Luo X, Yang Z, Zhang Y, Yuan P, Tang Y, Zhang T (2019) Software defect prediction based on kernel PCA and weighted extreme learning machine. Inf Softw Technol 106:182–200
2. Miholca DL, Czibula G, Czibula IG (2018) A novel approach for software defect prediction through hybridizing gradual relational association rules with artificial neural networks. Inf Sci 441:152–170

3. Oluwagbemiga BA, Shuib B, Abdulkadir S, Marian G, Thabeb A (2019) A hybrid ant colony tabu search algorithm for solving next release problems. Int J Innovative Technol Exploring Eng 8:191–198
4. Mumtaz B, Kanwal S, Alamri S, Khan F (2021) Feature selection using artificial immune network: an approach for software defect prediction. Intell Auto Soft Computing 29(3):669–684
5. Balogun AO, Basri S, Capretz LF, Mahamad S, Imam AA, Almomani MA, Adeyemo VE, Kumar G (2021) An adaptive rank aggregation-based ensemble multi-filter feature selection method in software defect prediction. Entropy 23(10):1274–1276
6. Balogun AO, Lafenwa-Balogun FB, Mojeed HA, Usman-Hamza FE, Bajeh AO, Adeyemo KS, Jimoh RG (2021) Data sampling-based feature selection framework for software defect prediction. International Conference on Emerging Applications and Technologies for Industry 4.0 (EATI'2020) Emerging Applications and Technologies for Industry, 4.0. pp 39–52
7. Bala YZ, Samat PA, Sharif KY, Manshor N (2022) Improving cross-project software defect prediction method through transformation and feature selection approach. IEEE Access 11:2318–2326
8. Lin J, Lu L (2021) Semantic feature learning via dual sequences for defect prediction. IEEE Access 9:13112–13124