International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Software Defect Prediction Through Effective Weighted Optimization Model for Assured Software Quality

Devi Priya Gottumukkala[1], D. Ushasree[2] and T. V. Suneetha[3]

Submitted: 04/11/2023 **Revised**: 28/01/2024 **Accepted**: 05/02/2024

Abstract: Software Defect Prediction is one of the active research areas in software engineering. Defect prediction approach identifies the defect prone modules before the testing phase starts. Metrics based defect prone modules improve the software quality, reduce the cost and leading to effective allocation of resources. This paper developed an effective software defect prediction model for the software quality assurance. In the first module, the various classifier's performance is analyzed using all the metrics of the KC1 dataset. In the second module, Firefly optimization algorithm is used for selecting the minimal number of metrics and passing them as input to the SVM classifier. In this paper, the fitness function of the Firefly algorithm is modified to maximize the accuracy and minimize the number of metrics. Based on the fitness function, Firefly algorithm tries to find a better set of metrics which improve the accuracy of defect prediction. In the third module, Hybrid FF or WFCMFF (Weighted FCM Firefly Search) approach is proposed to find a better set of metrics to further improve the performance of defect prediction. This approach combines the Firefly Algorithm and the Stochastic Weighted FCM Search algorithm to select the better set of metrics. The obtained results show that, the WFCMFF approach classifies the defect prone modules better when compared to the FF based feature selection. The achieved accuracy is 93.26%. for the SVM classifier. The classification-based defect prediction Model is evaluated in terms of its accuracy in classifying the module as defective or non-defective. Results proved that the proposed defect prediction Model has improved the accuracy from 86.27 % to 93.26%. Thus, the proposed classification-based defect prediction Model using FF and WFCMFF approaches, highly improves the defect prediction task.

Keywords: *Software Defect, Firefly Optimization, Feature Selection, Weighted FCM, Classification*

1. Introduction

Software programming is a discipline that is worried about all parts of programming creation. Programmers ought to take on an orderly and coordinated way to deal with their work and utilize suitable instruments and strategies relying upon the issue to be settled, the improvement limitations and the assets accessible [1]. Computer programming is worried about all parts of programming creation from the beginning phases of framework particular till the upkeep of the framework after it has been utilized to utilize [2]. Programming is the use of a deliberate, trained, quantifiable way to deal with the turn of events, activity, and support of programming; that is, the utilization of designing to programming (IEEE 1990) [3]. The "efficient, trained, quantifiable methodology" is many times named a product cycle strategy (in the general sense) or a product improvement process (in the particular sense). Explicit programming improvement processes comprise of a specific arrangement of programming improvement rehearses, which are much of the time performed by the computer programmers in a foreordained request [4]. In Programming industry, there are a few extraordinary and major problems during the product improvement process. A portion of the issues are Manageable Medium, Evolving Prerequisites, Timetable Hopefulness and Timetable Strain [5].

Software Quality Assurance (SQA) is an arranged and efficient example of activities important to give satisfactory certainty that a product item adjusts to the necessities during programming improvement [6]. SQA comprises of strategies and procedures of evaluating the product improvement cycles and techniques, apparatuses, and advances used to guarantee the nature of the created programming. SQA is ordinarily accomplished using obvious standard works on, including instruments and cycles for quality control to guarantee the respectability and dependability of programming [7]. Programming quality confirmation is perhaps of the main part in programming project the board. Research on different points of view of programming quality and related exercises has been directed for a very long time, and numerous ends and practices have been introduced to further develop programming quality [8]. One part of the exploration in this space is to lay out programming quality assessment Models that could be utilized at the beginning phases of a venture to gauge the quality level. The

[1]*Assistant Professor, Department of Computer Science and Engineering, Malla reddy university, Hyderabad. India. Email: mantena2377@gmail.com*
[2]*Assistant Professor, Department of CSE, Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad. India. Email: dupakuntlausha@gmail.com*
[3]*Assistant professor, Department of CSE, Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad. India. Email: takkellapati9@gmail.com*

assessment results can go about as a rule to upgrade the quality confirmation execution [9].

SQA is much of the time saw as a region causing extra expenses and postpones in programming creation [10]. Expanded mechanization support for SQA is promising for cost decrease and effectiveness, yet additionally for deliberately working on the item and cycle quality through repeatability, discernibility, and consistency. Nonetheless, computerization raises difficulties, particularly because of the dynamicity and pre-adulthood of Programming improvement as a discipline. Because of the huge effect of SQA on project costs [11], the expense viability of cement SQA measures (i.e., activities) is fundamental, and ought to be founded on logically applicable Computer programming climate information for practical transformation to changing conditions and limitations. The convenient task of SQA measures requires their tight coordination in Programming advancement processes, explicitly substantial work processes that must essentially be adjusted since the specific measures are not foreknown however are logically reliant. SQA is corresponding comparative with generally project size [12], while how much exertion distributed to SQA ought to be front-stacked (up to 25% of advancement exertion) and be decreased later [13].

The Nature of the advancement interaction straightforwardly influences the nature of the conveyed items. The quality arrangement is the arrangement of value related exercises that an undertaking intends to do to accomplish the quality objective. Quality objectives are determined concerning acknowledgment rules. The conveyed programming ought to work in every one of the circumstances and experiments in the acknowledgment models. The Quality Confirmation (QA) process includes choosing norms that can be applied to the product cycle and item [14]. The QA cycle begins with arranging and leading examination and audits. It is a continuous interaction inside the Product Improvement Life Cycle (SDLC) that regularly takes a look at the created programming to guarantee that it meets the ideal quality measures. SQA processes test for quality in each period of improvement until the product is finished. With SQA, the product improvement process moves into the following stage solely after the current/past stage consents to the expected quality principles. SQA by and large deals with at least one industry principles that assistance in building programming quality rules and execution methodologies [15].

1.1 Contribution of the Paper

In this paper aimed to develop an effective model for the software defect detection or prediction to achieve desire software quality assurance. The specific contribution of the paper are presented as follows:

1. Initially, the firefly optimization model is utilized for the estimation of the metrics for the software classification.
2. Based on the estimation model FireFly model is implemented with the weighetd FCM model for the selecting the minimal number of metrics and passing them as input to the classifier.
3. With the WFCMFF (Weighted FCM Firefly Search) approach is proposed to find a better set of metrics to further improve the performance of defect prediction. This approach combines the Firefly Algorithm and the Stochastic Weighted FCM Search algorithm to select the better set of metrics.
4. The obtained results show that, the WFCMFF approach classifies the defect prone modules better when compared to the FF based feature selection. The achieved accuracy is 93.26%. for the SVM classifier. The classification-based defect prediction Model is evaluated in terms of its accuracy in classifying the module as defective or non-defective. Results proved that the proposed defect prediction Model has improved the accuracy from 86.27 % to 93.26%. Thus, the proposed classification-based defect prediction Model using FF and WFCMFF approaches, highly improves the defect prediction task.

This paper is organized as follows: Section 2 provides the related works for software defect prediction. The methodology for the software defect prediction is presented in section 3 and results are presented in Section 4. Finally, the overall conclusion model is presented in Section 5.

2. Related Works

In [16] utilizes the Rao optimization method and the multi-criteria decision-making (MCDM) method in a hybrid feature selection (filter–wrapper) strategy to select the most informative features in order to raise the software defect prediction rate. The proposed work estimates the wellness of the competitor arrangement by utilizing the deformity expectation rate and the component choice proportion. Three well-known NASA benchmark datasets—PC5, JM1, and KC1—are used to compare the proposed method's performance to that of current methods. On the benchmark datasets, The proposed feature subset selection scheme selects the most essential feature subset for defect prediction with an average accuracy of 95%. In terms of defect prediction rate, the experimental results demonstrate that the proposed hybrid strategy performs better than the standard strategy.

[17] examined the effect of hyperparameter optimization on defect count prediction. We discovered that hyperparameter optimization of learning methods: From 15 software defect datasets: 1) boosts MLPR, Lasso, DTR, Hubber, and SVR's prediction performance by 16.96%,

8.31%, 8.16%, 6.01%, and 5.22 percent, respectively; (2) Optimization is not taken into account by a linear regression; 3) Random search optimization outperformed grid search optimization overall in terms of prediction performance by 3.36 percent; 4) The non-significant classifier's ranking has also changed a lot, and 5) logistic regression received the highest ranking for hyperparameter optimization. Despite the fact that both methods performed well, the grid search consistently outperformed the random search when it came to the default parameter. However, with a random search, it might not be. When using parameter-sensitive regression methods, this emphasizes the significance of exploring the parameter space.

Domain adaptation (DA) is carried out with the help of kernel twin support vector machines (KTSVMs) in [18], which evaluated the distributions of the training data for various projects. Moreover, this paper uses DA-KTSVMs, or KTSVMs with DA capacities, as the CPDP model. An improved quantum particle swarm optimization algorithm (IQPSO) is used to optimize the parameters of the DA-KTSVM, which has an effect on the predictive performance of the model. The improved DA-KTSVM is referred to as DA-KTSVMO. Experiments on 17 open source software projects are carried out in order to verify the efficacy of DA-KTSVMO. DA-KTSVMO is capable of not only achieving an expectation execution that is superior to that of the other CPDP models that were examined, but it is also capable of achieving an expectation execution that is virtually identical to or superior to that of WPDP models, as demonstrated by the findings of the investigation as well as the findings of the exploratory tests. In a similar vein, DA-KTSVMO's expectation execution can be improved by utilizing the data that is currently in place and reusing incorrect data.

The newly developed High dimension software defects prediction model (HD-SDP) based on SVM is proposed in [19] as a synchronous solution to the issue of dataset class imbalance in software defects prediction and support vector machine (SVM) parameter selection. incorporating the F-metric, the rate of false positive defects, the balance value, and the detection probability as four objectives. A unified integration of many-objective optimization algorithm based on temporary offspring (UIMaOTO) is utilized in addition to the synchronous selection of the parameters for the SVM and non-defective module in this model. In order to produce formal offspring, UIMaOTO employs a temporary offspring strategy and then proposes a unified integration strategy to raise the selection pressure on the algorithm. The analysis' results are contrasted with those of other state of the art calculations utilizing UIMaOTO on the notable DTLZ test suite. In many-objective optimization problems, the findings demonstrate that the proposed algorithm outperforms the competition overall. When dealing with the HD-SDP model simultaneously, the UIMaOTO algorithm performs better than the other algorithms by 14.27 percent.

WSHCKE, a unified defect prediction predictor, was constructed using a hybrid deep neural network consisting of a kernel extreme learning machine (KELM) and a convolutional neural network (CNN) [20]. This WSHCKE can further integrate the selected features into the abstract deep semantic features generated by the CNN and improve prediction performance by fully utilizing the strong classification capacity of KELM. On twenty all-around focused on programming projects, we complete broad analyses for include choice or extraction as well as imperfection expectation on four assessment points. The consequences of the analyses show that WSHCKE and EMWS are better.

In [21], the software defect prediction model based on transfer learning was utilized. The dataset from the initial production site was used to train multitask artificial neural networks (ANNs) for defect prediction and surface gloss. After that, the previously trained mold and ANN model were moved to a different manufacturing facility. With R2 = 0.94, the new machine had the option to anticipate surface shine for the pre-prepared model precisely; However, due to various machine characteristics, the surface defect prediction was not as accurate as it could have been. On the single teachable result layer, the forecast exhibition of the exchange learning was high and stable. In addition to producing a surface defect prediction with an accuracy of 0.90 that was superior to that of surface gloss ($R2 > 0.95$) and comparable to that of surface gloss, the use of transfer learning significantly reduced the size of the dataset that was required. Physical molding experiments showed that the transferred model allowed for model-based, multi-objective process parameter optimization, which made it possible to produce injection-molded parts with high surface quality that are both durable and effective.

A robust information-driven HDP model known as IVKMP was proposed in [22] for this review. InfoGAN (Information maximizing GANs), a cutting-edge deep generation network that simultaneously achieves class balance and generates sufficient instances of defects for data augmentation, serves as our foundation. Also, the multi-objective VaEA (Vector point based Formative Estimation) progression is utilized to choose the metric subsets with the least number of representative focuses while finishing the base mistake. Last but not least, a deep defect predictor for HDP based on the light but powerful deep network PCANet (Principal Component Analysis Network) is built using a block-wise histogram and binary hashing to basically capture robust representations that are more semantically related. The IVKMP model is contrasted with various state of the art standard models

more than 542 heterogeneous undertaking matches including 26 programming projects. The outcomes of the trials demonstrate the prevalence and power of our IVKMP model.

[23] provides a suggestion for the framework for dual-scale ensemble learning. The use of sample entropy (SE) and complete ensemble empirical mode decomposition adaptive noise (CEEMDAN) to decompose and reconstruct AQI series makes direct modeling simpler. Low-frequency components are predicted by the regularized extreme learning machine (RELM), while high-frequency components are predicted by the long short term memory neural network (LSTM). The hyper-parameters of the RELM and LSTM models are also optimized with the help of the improved whale optimization algorithm (WOA). Finally, the half breed expectation model presented in this paper can be used to predict the AQI of four Chinese urban communities. The accuracy of AQI predictions, which is crucial to the sustainable growth of cities, is effectively enhanced by this work.

3. Weighted FCM Firefly Optimization for Feature Extraction

This research focused on the extraction of features in the software defect prediction (SDP). The proposed (Weighetd FCM Firefly Feature Optimization Classification) WFCMFF model for the software defect identification. The WFCMFF model aimed to design the fitness function evaluation through multi-objective feature selection model to increase the classifier accuracy to increase the performance of classification. Through the integration of Firefly optimization model features are optimized for the prediction of defects. The overall process in the proposed WFCMFF model architecture is presented in figure 1.

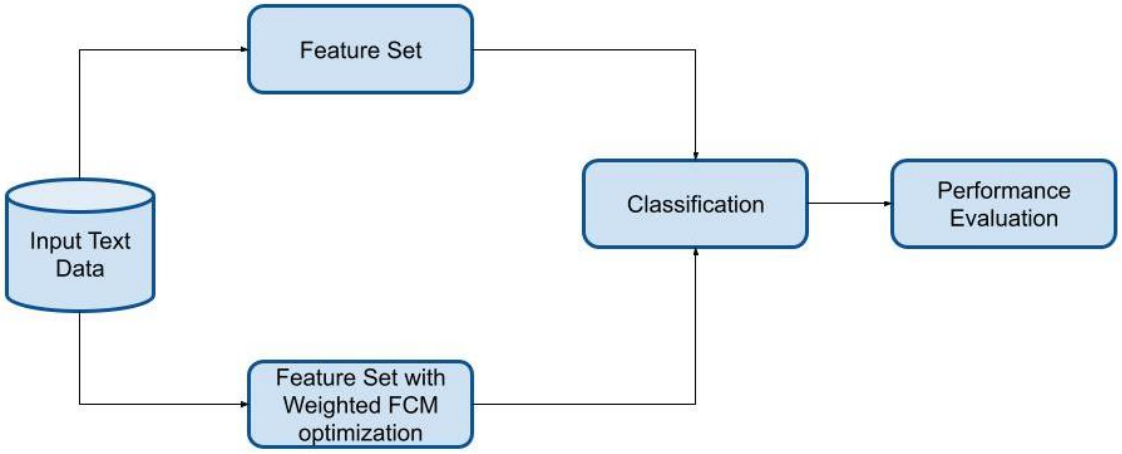

Fig 1: WFCMFF Model for Software Defect Prediction

Firefly Algorithm (FA) belongs to the class of inspired algorithm with the fireflies past and flashing of the fireflies behaviour. The fireflies flashing behavior attract towards the mating groups. The firefly those are less bright are attracted by the brighter one. The estimation is based of consideration of fireflies as unisexual those are attracted each other. The WFCMFF model comprises of the estimation of objective function with the swarm-based intelligence optimization model to resolve optimization problem those are proportional to the firefly for the quality through the problem setting. Subsequently, the brighter firefly are attracted with the partner those are in search space effectively. The FFOA model comprises of the fireflies $x_{i,}\ i\ 1,2,\ldots.,n$ those are positioned initially and evaluated in the search space intensity for the every firefly I related to the objective function $f(x) = l = \alpha f(x)$. The each firefly subjected to higher intensity with each other denoted as i.e $li > lj, j = 1,2,\ldots.,n, j \neq i.$ The brightness level of each fireflies varies from firefly to firefly based on the distance denoted as $j; r_{ij} = d(x_i, x_j)$. Additionally, the intensity of light of fireflies decreases in distance based on the source obtained through air. Thus, fireflies are limited for the specific distance. The FFOA model comprises of the real number deployed in random manner through global communication based on particles. The resulted FFOA model is effective with the multi-objective optimization model with the signal function attraction between mate to attract the prey value. Based on the square-law the light intensity is evaluated based on the distance those are inverse. The light intensity distance are minimizes effective in the formulated manner associated with the optima objective function. The proposed FFOA model flow chart is presented in figure 2 as follows:

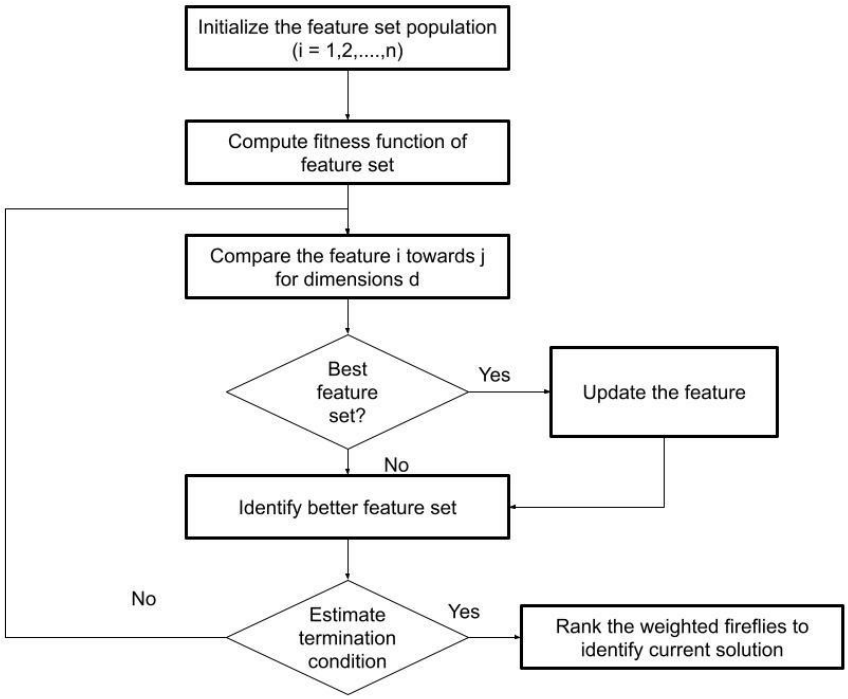

Fig 2: Flow of WFCMFF

Algorithm 1: Pseudo Code of Firefly algorithm

Generate initial population of feature set x_i $(i = 1,2, \ldots, n)$

Fitness function f at xi is determined

$$fitness = \alpha\left(\frac{T_{tot} - T_{io}}{T_{tot}}\right) + \frac{\beta}{F}$$

While (t <MaxGeneration)

for i = 1 : n all n featureset

for j = 1 : i all n featureset

if $\left(f(x_i) > f(x_j)\right)$

$$x_j = x_i + \beta_0 e^{-\gamma r_0^2}\left(x_j - x_i\right) + \alpha \in_i$$

end for j

end for i

Rank the feature sets and find the current best

end while

The feature set for the weighted FCM model based on the fireflies are estimated in feature set is presented in figure 3.

FeatureSet 1	1	0	1	1	0	1	1	0	1	0	1	1	0	1	0	0	1	1	0	0	1
FeatureSet 2	0	0	0	1	1	1	1	1	1	0	0	1	0	1	1	0	0	0	1	1	0
FeatureSet n	0	0	1	0	1	0	1	0	1	0	0	1	0	1	1	0	0	0	1	1	0

Fig 3: Sample Population of Feature Set

3.1 Fitness Function

Fitness function estimates the desire solution for the evaluation of suitability for the extraction of features. With WFCMFF model evaluates the fitness function assigned based on the computed assigned score solution. The score are computed based on the numerical representation of value to achieve the specific solution to resolve the problem. Inbiologyal point of view, fitness is implemented to derive the individual score value based on the conditions. The individual environment is represented based on the search space. The WFCMFF model compute the minimal fitness value to derive the possible solution value in the set. In general, fitness function F(x) is computed based on the firefly based objective function for the multiple solution to achieve near or optimal values. The optimal solution is evaluated based on the multiple population to achieve the optimal solution. The multiple objective function with WFCMFF model is stated as in equation (1)

$$F(x) = k_1 f_1(x) + k_2 f_2(x) + \cdots \ldots + k_n f_n(x) \quad (1)$$

In above equation (1) weights are stated as k_1, k_2 and k_n.

3.2 Fitness function for metric selection

With WFCMFF model the fitness function is computed based on the estimation of the optimal solution values. Through consideration of particular ranking solution feature metrices are evaluated based on the fitness function estimation with firefly algorithm integrated with multiple functions. The feature metrices are evaluated based on the fitness function integrated with the consideration of multiple objective functions. The multiple-objective function are combined with the single objective function as stated in equation (2)

$$fitness = \alpha\left(\frac{T_{tot} - T_{io}}{T_{tot}}\right) + \frac{\beta}{F} \quad (2)$$

where α and β stated as the constant lies between 0 & 1, either + =1

The firefly attraction is considered based on the light intensity associated with the encoded objective function. To overcome the optimization brightness of the firelfies are computed based on I with position x stated as I(x) ~ f(x). However, the relative factors are estimated for the attractiveness for the varying distance r_{ij} based on the i and j. The intensity of lights are evaluated based on the source with the varying degree of the absorption. The light intensity of fireflies with the varying distance $I(r)$ with the distance r those are computed with the exponential variation measured as in equation (3)

$$I = I_0 e^{-\gamma r} \quad (3)$$

In the above equation (3) the intensity of light is represented as I_0 with the absorption coefficient of light as γ. As the attractiveness of the fireflies are directly proportional to the adjacent fireflies with the attractiveness level β stated in the equation (4)

$$\beta = \beta_0 e^{-\gamma r} \quad (4)$$

The attractiveness variation based on the distance of the communicated fireflies are stated as γ denoted as $\gamma \in [0,10]$. The movement of fireflies are computed for the movement i for the firefly i attractiveness as stated in equation (5)

$$x_j = x_i + \beta_0 e^{-\gamma r}\left(x_j - x_i\right) + \alpha\varepsilon_1 \quad (5)$$

The first term is associated with the attraction and third term is defined as the randomization with the random variable vector ε_1 with the Gaussian distribution of $\alpha \subseteq [0,1]$. The fireflies distance are computed based on the cartesian distance value as stated in the equation (6)

$$r_{ij} = \left\|x_i - x_j\right\| = \sqrt{\sum_{k=1}^{d}\left(x_{i,k} - x_{j,k}\right)^2} \quad (6)$$

In above equation (6) the i[th] Firefly spatial co-ordinates of the k[th] element denoted as $x_{j,k}$. The firefly algorithm fitness function is maximized through the validation set for the training set as presented in equation (7) for the selected feature number

$$f_\theta = \omega * E + (1 - \omega)\frac{\sum_i \theta_i}{N} \quad (7)$$

In above equation (7) the fitness function is represented as f_θ with the vector size of N with 0/1 elements for the selected features. The dataset feature set total count is denoted as N and classification error is presented as E with the classification of feature values. The individual fitness is evaluated based on the threshold based extracted feature as in equation (8)

$$f_{ij} = \begin{cases} 1 \; if \; x_{ij} > 0.5 \\ 0 \quad otherwise \end{cases} \quad (8)$$

In the above equation (8) the search agent dimension are stated as x_{ij} with dimensions i and j. The firefly solution are updated based on the dimensional value with the limited constraints value of [0, 1], with the rule for truncation to ensure the limited variables. The factor those are decremented exhibits the constant rate with the end optimization of the minimal value as presented in equation (9) with the maximal exploration of optimization in the end scenario

$$\alpha_{t+1} = \alpha_t * \delta \quad (9)$$

In the above equation (9) the randomization factor α and α_t provides the rate of change as stated in δ for the iteration count of t.

3.3 Weighted FCM Search Algorithm Defect Prediction

The WFCMFF model comprises of the optimization with the multi-agent global search with the single agent interaction with tandem calling mechanism. The model comprises of the sequence of steps with foraging model to identify the position of food and single ant position is computed with SDS based description model with the social metaphor value demonstrated procedure of SDS model. The WFCMFF model with SDS perform the initialization of population to evalute the standard solution as explained below:

Initialising agents ()

While (stopping condition is not met)

Testing hypotheses()

Weighted FCM hypotheses()

End While

In the testing phase, the evaluation is computed based on every agent through the randomly selected region for the s process. The agent based mechanism perform the inactive and Weighted FCM phase for the random agent estimation for the selection of agent those are inactive and active those are generated random manner. The figure 4 provides the graphical illustration of the proposed WFCMFF model flow chart.

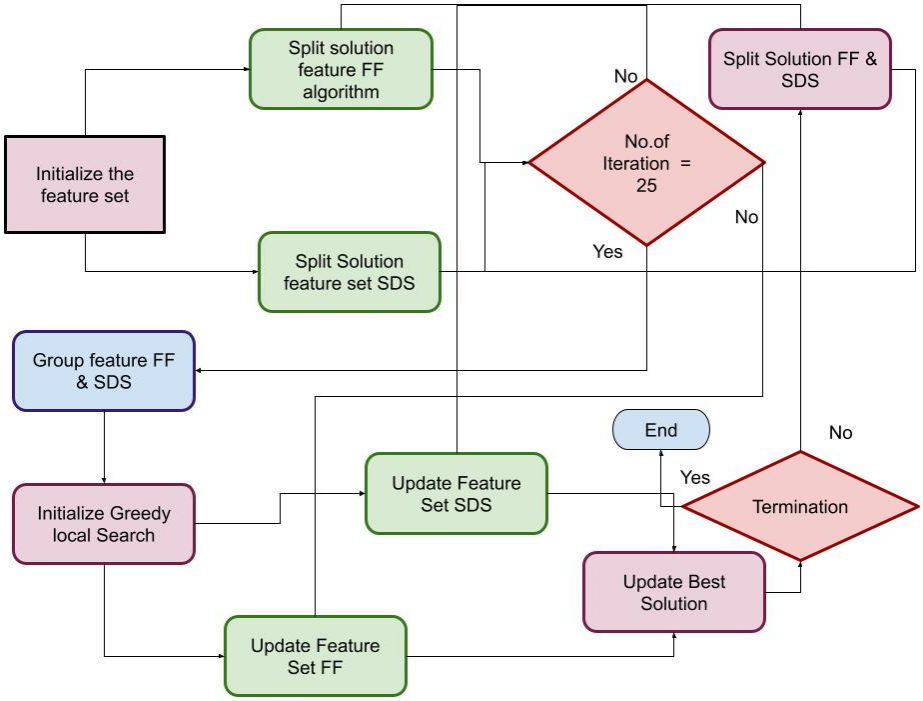

Fig 4: Overall Process of WFCMFF

Algorithm 2: Pseudo Code of WFCMFF Approach
Step 1: Initialize the population with feature sets randomly. Feature set consists of 0' s(Feature not selected) and 1's (feature selected) Step 2 : Calculate the fitness function for the feature set. Step 3 : Split the featureset and initiate the SDS and Firefly Algorithm and start the iteration. Step 4 : Update best solution at each iteration of SDS and FF Step 5 : After n*25th iterations, greedy search is initiated by grouping all the feature sets from SDS and Firefly. Step 6 : Repeat the step from 2 until the maximum iteration is reached
Greedy (Feature Set, Size, fitness function) Fitness Function f (xi) (i=1..N) Step 1 : For each feature set Xi (i=1..N) Step 2 : For each feature set Xj (j=i+1..N) Step 3 : If (f (xi))≤ f (xj))

Step 4 : Replace the fitness value of f (Xi) with fitness value of f(Xj)

Step 5 : Update the corresponding feature set.

Step 6: End if

Step 7: End For

Step 9: End For

Return feature set.

3.4 KC1 Dataset

The dataset for the WFCMFF model with consideration for repository model comprises of the explicit dataset deployed to increases the software prediction method for defect estimation with consideration of different metrices. The dataset is evaluated based on the consideration of NASA software detection based subsystem. The dataset comprises of the static code for the fault data and estimation of corresponding modules. The modules are estimated based on the defined function, procedure or method of projects. Through the effective data mining model the accurate value is assessed for the computation with classification process. The software metrices examined for the dataset is similar to that of Line of Code (LOC) with consideration of operators and operand, length of program, complexity, estimated time, effort and other metrices for the measurement of software defect. The variation in the defects are computed based on the non-normal characteristic such as excessive value, variances, collinearity and skewness. The characteristics of the dataset in analysis of the software are presented as follows:

Number of cases – a) Small (n <= 500), b) Medium (500 < n <10000), c) Large (n >= 10000).

Number of features – a) Small (p <= 6), b) Medium (6 < p <20), c) Large (p >= 20);

Distribution of values – a) Skewed, b) Outliers;

Independence of features – a) Independent, b) Multi-collinear;

Feature type – a) Discrete, b) Continuous.

The KCI dataset is evaluated based on the consideration of project data processing with the 2109 modules with C++. The Metric Data Program (MDP) comprises of the data repository with the logical group of data through Computer Software Components (CSCs) for the larger system. In those modules, the dataset comprises of 328 fault data and 1.781 non fault data. With WFCMFF prediction is evaluated based on the product metrices with the consideration of dependent and independent variables. The dataset comprises of the dependent class label those are independent to each other. The KCI model comprises of C++ system implemented for the management of storage and ground receipt for the processing to extract the feature code base don measured modules. The dataset comprises of the 22 various attributes with 4 LOC, 4 McCabe , 1 goal field,12 Halstead metrices as presented in table 1.

Table 1: List of Software metrics in KC1 dataset

Metric Type	**Notation**	**Description**
LOC	LOComment	Total number of comment lines
	LOCode	Total no of executable codes
	LOBlank	Total number of blank lines
	LOCode and Comment	Total number of Lines of code and comment
McCabe	Loc	Line of code
	v(g)	cyclomatic complexity
	ev(g)	essential complexity
	iv(g)	design complexity

Metric Type	Notation	Description
Halstead	N	Total no of operators + operands
	V	Halstead volume
	L	Halstead Program level
	D	Halstead difficulty
	I	Halstead intelligence
	E	Halstead effort
	B	Halstead error estimate
	T	Halstead's time estimator
	uniq_Op	unique operators
	Uniq_Opnd	unique operands
	total_Op	total operators
	total_Opnd	total operands
Miscellaneous	BranchCount	branch count
Class label	Defective	{True or False} i.e module has defect/no defect

4. Results and Discussion

The performance of the proposed WFCMFF model is evaluated based on the consideration of different metrices to achieve the defect prediction. The performance metrices utilized are accuracy, precision, recall and F-measure. Through the analysis the confusion matrix is formulated for the performance evaluation with the classification algorithm. In table 2 the confusion matrix estimated in presented.

Table 2: Confusion Matrix of Defect Classification

	Defect (Predicted)	**No Defect (Predicted)**
Defect (Actual)	T_p	F_n
No defect(Actual)	F_p	T_n

Tp = A Defective module is correctly classified as a defectivemodule

Fp = A Non defective module is wrongly classified as a defectivemodule

Fn = A Defective module is wrongly classified as a non defectivemodule.

Tn= A Non defective module is correctly classified as a nondefective module.

The performance of proposed WFCMFF model is evaluated for the consideration of different classifiers such as SVM, KNN, NB, DTNB, NB simple, PART and Bayesnet to achieve the feature selection. The comparative analysis of propsoed WFCMFF with the

conventional technique are presented as follows the selected features are presented in table 3

Table 3: List of Features Selected and parameter setting in WFCMFF

	Features
Weighted FCM FireFly (WFCMFF)	LOCode, cyclomatic complexity, design complexity, Halstead Program level, Halstead effort, Halstead error estimate, unique operators, unique operands & branch count
No of features Selected	9
Parameter setting for FF in WFCMFF	N=15 fireflies , α=0.5,β =0.5,δ =0.3,y =0.5
SVM kernel	Linear

In table 4 provides the comparative analysis of selected features for the proposed WFCMFF with consideration of 9 features for the input classifiers. The WFCMFF model comprises of the different classifiers such as NBSimple, SVM, PART, DTNB, NBUpdatable, NNgeCompliment and bayesNet. It is observed that proposed WFCMFF model exhibita the improved performance compared with other classifiers. The developed WFCMFF model exhibits the improved global optimization solution for the cluster to achieves the effective performance characteristics.

Table 4: Comparison of Results for KC1 dataset without FS and with FF FS

Classifiers		**ClassificationAccuracy**	**Precision**	**Recall**	**F -Measure**
SVM	Without WFCMFF	90.27	0.8048	0.9098	0.8541
	With WFCMFF	93.26	0.8822	0.8507	0.8654
KNN	Without WFCMFF	87.76	0.7728	0.8912	0.8278
	With WFCMFF	92.31	0.8668	0.825	0.8439
NB	Without WFCMFF	86.38	0.7581	0.8843	0.8164
	With WFCMFF	91.5	0.8418	0.8253	0.8332
PART	Without WFCMFF	87.8	0.7656	0.8047	0.7827
	With WFCMFF	91.27	0.8358	0.8239	0.8297
DTNB	Without WFCMFF	88.75	0.7813	0.8354	0.8039
	With WFCMFF	90.32	0.819	0.7994	0.8087

NBSimple	Without WFCMFF	84.01	0.7197	0.8074	0.7474
	With WFCMFF	89.75	0.8053	0.796	0.8005
BayesNet	Without WFCMFF	71.19	0.6468	0.7693	0.6392
	With WFCMFF	85	0.7218	0.7679	0.7405

The table 5 provides the comparative examination of feature selection process with the Firefly and the WFCMFF model for the software defect prediction mode.

Table 5: Comparison of Results for KC1 dataset with FF FeatureSelection and WFCMFF Feature Selection

Classifiers	**Feature Selection**	**ClassificationAccuracy**	**Precision**	**Recall**	**F -Measure**
SVM	FF (FireFly)	90.27	0.8048	0.9098	0.8541
	WFCMFF	93.26	0.8822	0.8507	0.8654
KNN	FF	87.76	0.7728	0.8912	0.8278
	WFCMFF	92.31	0.8668	0.825	0.8439
NB	FF	86.38	0.7581	0.8843	0.8164
	WFCMFF	91.5	0.8418	0.8253	0.8332
PART	FF	87.8	0.7656	0.8047	0.7827
	WFCMFF	91.27	0.8358	0.8239	0.8297
DTNB	FF	88.75	0.7813	0.8354	0.8039
	WFCMFF	90.32	0.819	0.7994	0.8087
NBSimple	FF	84.01	0.7197	0.8074	0.7474
	WFCMFF	89.75	0.8053	0.796	0.8005

BayesNet	FF	71.19	0.6468	0.7693	0.6392
	WFCMFF	85	0.7218	0.7679	0.7405

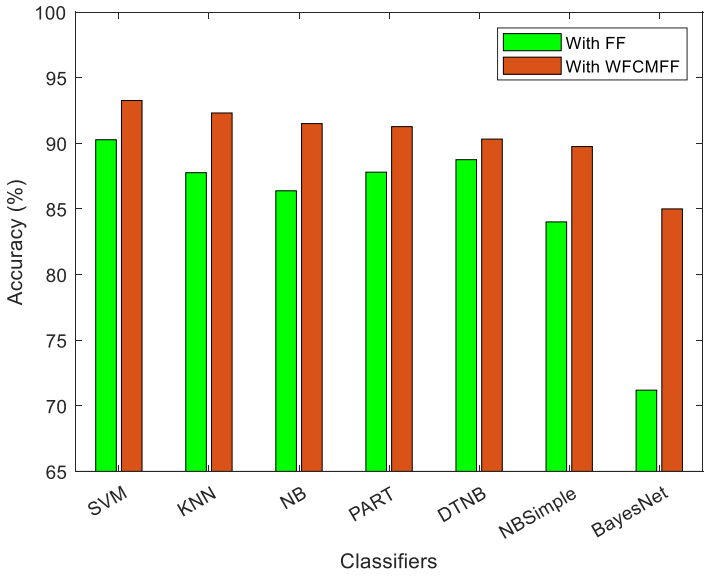

Fig 5: Comparison of different classifiers based on Accuracy forFF Feature Selection and WFCMFF Feature Selection

In the figure 5 it is observed that the proposed model exhibist the improved classification accuracy value of 3.35% than the SVM with proposed WFCMFF. The performance of WFCMFF model is 6.07% higher performance than the KNN model. NB model achieve the classification accuracy of 6.03% with PART model the accuracy of WFCMFF is 4.95% higher. In case of DTNB model the proposed WFCMFF classification achieves the 10.43% higher performance and with Bayesnet model the accuracy is 26.84% higher than WFCMFF model. The KNN achieves the minimal accuracy value of 1.9% less than the proposed WFCMFF. The performance of WFCMFF is 3.83% and 9.26% higher than the NBSimple and BayesNet model.

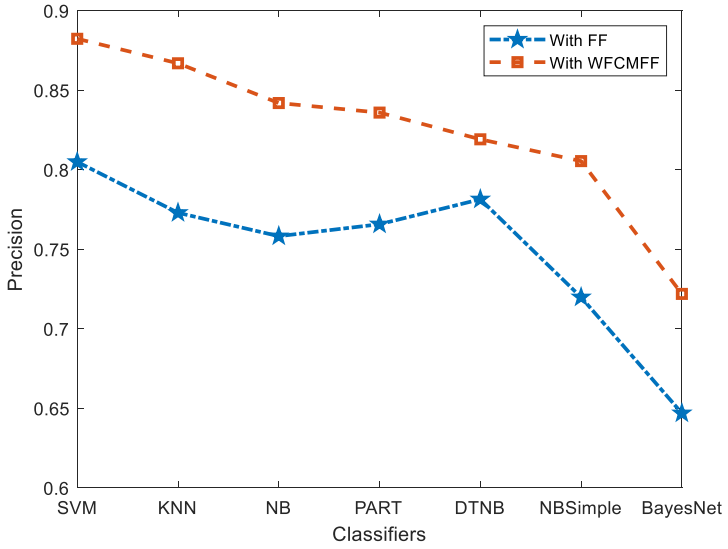

Fig 6: Comparison of different classifiers based on Precision forFF Feature Selection and WFCMFF Feature Selection

According to Table 5 and Figure 6, SVM- with WFCMFF FS has a higher precision of 9.18 percent, while KNN-with FF FS has a higher precision of 13.22 percent, NB-with FF FS has a higher precision of 15.13 percent, PART with FF FS has a higher precision of 14.2 percent, DTNB-with FF FS has a higher precision of 12.13 percent,

NBSimple with FF FS has a higher by 1.76% for KNN- with WFCMFF FS, by 4.69 percent for NB- with WFCMFF FS, by 5.4% with WFCMFF FS, by 7.43 percent with DTNB- with WFCMFF FS, by 9.11 percent with NBSimple- with WFCMFF FS, and by 20 percent with Bayes Net- with WFCMFF FS.

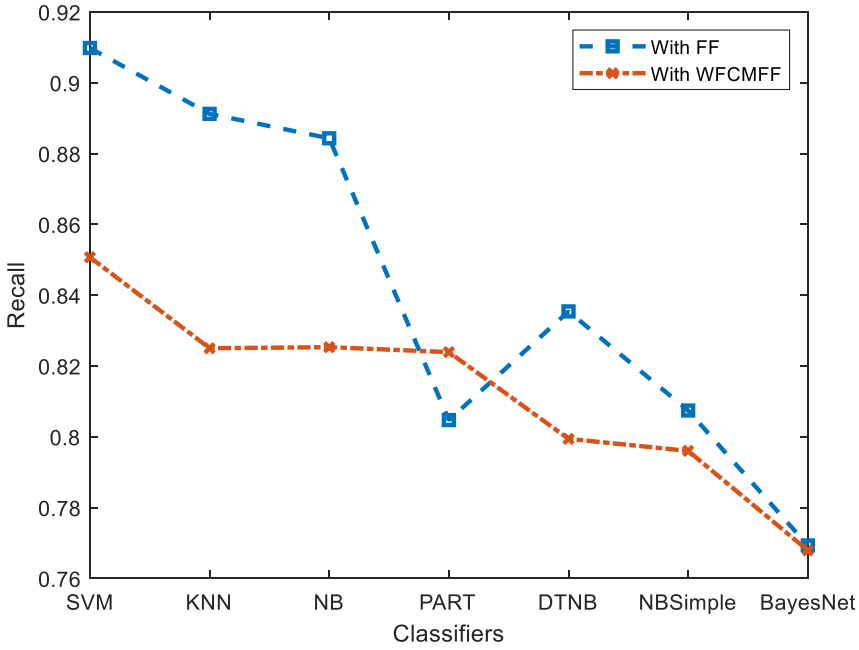

Fig 7: Comparison of different classifiers based on Recall for FFFeature Selection and WFCMFF Feature Selection

According to Table 5 and Figure 7, the recall of SVM with WFCMFF FS is significantly higher by 6.7% for SVM with FF FS, 4.65% for KNN with FF FS, 3.87 percent for NB with FF FS, 5.56 percent for PART with FF FS, 1.81 percent for DTNB with FF FS, and 5.22 percent for FF FS for NBSimple-with FF FS, by 10.05% for Bayes Net-with FF FS, by 3.07% for KNN-with WFCMFF FS, by 3.03% for NB-with WFCMFF FS, by 3.2% for PART-with WFCMFF FS, by 6.22% for DTNB-with WFCMFF FS, by 6.64% for NBSimple-with WFCMFF FS and by 10.2% for Bayes Net-with WFCMFF FS.

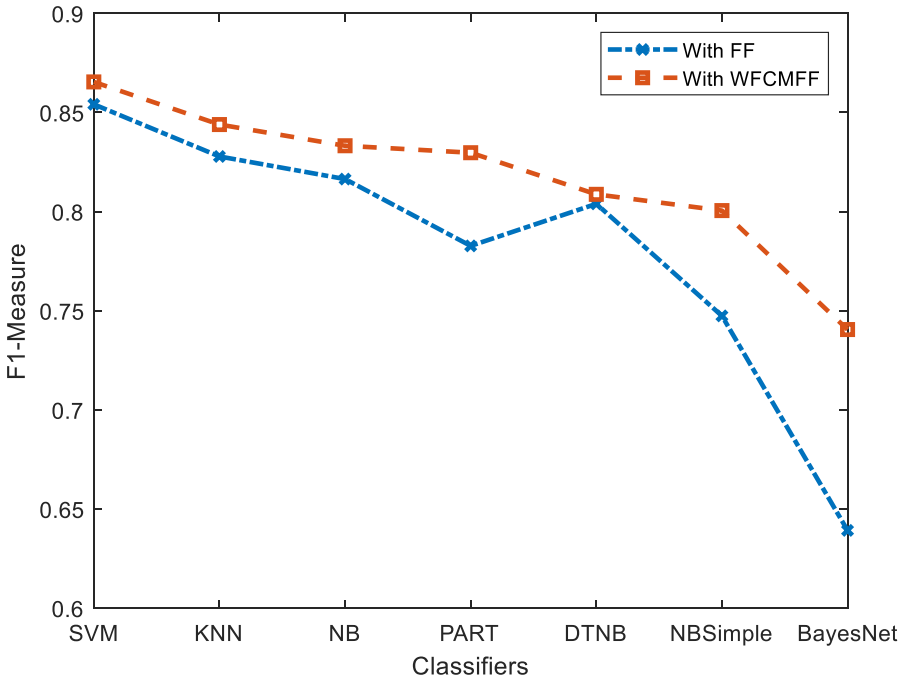

Fig 8: Comparison of different classifiers based on F -Measure forFF Feature Selection and WFCMFF Feature Selection

According to Table 5 and Figure 8, the F-Measure of SVM- with WFCMFF FS is higher by 1.3%, by 4.44 percent for KNN- with FF FS, by 5.83 percent for NB- with FF FS, by 10.04% for PART-with FF FS, by 7.37

percent for DTNB- with FF FS, by 14.6 percent for NBSimple- with FF FS, and by 30.7 percent for Bayes, by 2.52% for KNN- with WFCMFF FS, by 3.79 percent for NB-, 4.22 percent for PART-, 6.77 percent for DTNB-, 7.79 percent for NBSimple-, and 15.56 percent for Bayes Net- with WFCMFF FS.

5. Conclusion

This paper constructed WFCMFF model to select the minimal subset of features, and classifiers like SVM, NB, KNN, PART, DTNB, BayesNet and NBSimple were used in experiments to observe the accuracy, precision, recall and F-Measure. The following feature which can be eliminated from the KC1 dataset. They are LOComment, LOBlank, LOCode and Comment, essential complexity, Total no of operators and operands, Halstead volume, Halstead Difficulty, Halstead Intelligence, Halstead's time estimator, total operators and total operands. The results have confirmed that the classification accuracy, recall, precision and F-Measure values of SVM -with WFCMFF were better when compared to other classifiers. The improved accuracy in Hybrid Firefly algorithm is from 90.27 to 93.26 %.It is being achieved because of the feature set generated using FF and SDS. Thus the performance of the defect prediction improved with the effective software quality.

References

[1] Shu, R., Xia, T., Williams, L., & Menzies, T. (2022, May). Dazzle: using optimized generative adversarial networks to address security data class imbalance issue. In *Proceedings of the 19th International Conference on Mining Software Repositories* (pp. 144-155).

[2] Lavanya, S., Prasanth, A., Jayachitra, S., & Shenbagarajan, A. (2021). A Tuned classification approach for efficient heterogeneous fault diagnosis in IoT-enabled WSN applications. *Measurement*, *183*, 109771.

[3] Tabjula, J. L., Kanakambaran, S., Kalyani, S., Rajagopal, P., & Srinivasan, B. (2021). Outlier analysis for defect detection using sparse sampling in guided wave structural health monitoring. *Structural Control and Health Monitoring*, *28*(3), e2690.

[4] Kumar, A., Zhou, Y., & Xiang, J. (2021). Optimization of VMD using kernel-based mutual information for the extraction of weak features to detect bearing defects. *Measurement*, *168*, 108402.

[5] Ghoneim, S. S., Mahmoud, K., Lehtonen, M., & Darwish, M. M. (2021). Enhancing diagnostic accuracy of transformer faults using teaching-learning-based optimization. *Ieee Access*, *9*, 30817-30832.

[6] Abid, A., Khan, M. T., & Iqbal, J. (2021). A review on fault detection and diagnosis techniques: basics and beyond. *Artificial Intelligence Review*, *54*, 3639-3664.

[7] Susan, S., & Kumar, A. (2021). The balancing trick: Optimized sampling of imbalanced datasets—A brief survey of the recent State of the Art. *Engineering Reports*, *3*(4), e12298.

[8] Xie, R., Qiu, H., Zhai, Q., & Peng, R. (2022). A model of software fault detection and correction processes considering heterogeneous faults. *Quality and Reliability Engineering International*.

[9] Zhu, M., & Pham, H. (2022). A generalized multiple environmental factors software reliability model with stochastic fault detection process. *Annals of Operations Research*, 1-22.

[10] Pritoni, M., Lin, G., Chen, Y., Vitti, R., Weyandt, C., & Granderson, J. (2022). From fault-detection to automated fault correction: A field study. *Building and Environment*, *214*, 108900.

[11] Gupta, N., Sharma, A., & Pachariya, M. K. (2022). Multi-objective test suite optimization for detection and localization of software faults. *Journal of King Saud University-Computer and Information Sciences*, *34*(6), 2897-2909.

[12] Gokilavani, N., & Bharathi, B. (2021). Test case prioritization to examine software for fault detection using PCA extraction and K-means clustering with ranking. *Soft Computing*, *25*(7), 5163-5172.

[13] Zhang, L., Leach, M., Bae, Y., Cui, B., Bhattacharya, S., Lee, S., ... & Kuruganti, T. (2021). Sensor impact evaluation and verification for fault detection and diagnostics in building energy systems: A review. *Advances in Applied Energy*, *3*, 100055.

[14] You, L. (2023). Multi-channel data flow software fault detection for social internet of things with system assurance concerns. *International Journal of System Assurance Engineering and Management*, 1-11.

[15] Granderson, J., Lin, G., Singla, R., Mayhorn, E., Ehrlich, P., Vrabie, D., & Frank, S. (2021). Commercial fault detection and diagnostics tools: what they offer, how they differ, and what's still needed.

[16] Thirumoorthy, K. (2022). A feature selection model for software defect prediction using binary Rao optimization algorithm. *Applied Soft Computing*, *131*, 109737.

[17] Nevendra, M., & Singh, P. (2022). Empirical investigation of hyperparameter optimization for software defect count prediction. *Expert Systems with Applications*, *191*, 116217.

[18] Jin, C. (2021). Cross-project software defect prediction based on domain adaptation learning and

optimization. *Expert Systems with Applications*, *171*, 114637.

[19] Cai, X., Geng, S., Wu, D., & Chen, J. (2021). Unified integration of many-objective optimization algorithm based on temporary offspring for software defects prediction. *Swarm and Evolutionary Computation*, *63*, 100871.

[20] Zhu, K., Ying, S., Zhang, N., & Zhu, D. (2021). Software defect prediction based on enhanced metaheuristic feature selection optimization and a hybrid deep neural network. *Journal of Systems and Software*, *180*, 111026.

[21] Gim, J., Yang, H., & Turng, L. S. (2023). Transfer learning of machine learning models for multi-objective process optimization of a transferred mold to ensure efficient and robust injection molding of high surface quality parts. *Journal of Manufacturing Processes*, *87*, 11-24.

[22] Zhu, K., Ying, S., Ding, W., Zhang, N., & Zhu, D. (2022). IVKMP: A robust data-driven heterogeneous defect model based on deep representation optimization learning. *Information Sciences*, *583*, 332-363.

[23] Ji, C., Zhang, C., Hua, L., Ma, H., Nazir, M. S., & Peng, T. (2022). A multi-scale evolutionary deep learning model based on CEEMDAN, improved whale optimization algorithm, regularized extreme learning machine and LSTM for AQI prediction. *Environmental Research*, *215*, 114228.