HSoMLSDP: A Hybrid Swarm-Optimized Machine Learning Software Defect Prediction Framework

Madhusmita Das, Biju R. Mohan, and Ram Mohana Reddy Guddeti
Department of Information Technology
National Institute of Technology Karnataka, Surathkal
Mangalore, India
{madhusmitadas.197it004, biju, profgrmreddy}@nitk.edu.in

***Abstract*—Defect prediction plays a crucial role for any software system across various domains, as its failure may cause unavoidable and undeniable scenarios. For reliable software, defect-free is considered as one of the most important criteria. This research aims to design a hybrid swarm-optimized machine learning software defect prediction (HSoMLSDP) framework to predict software defects. We strive to do this by designing a swarm-optimized machine learning defect prediction (SoMLDP) model within the HSoMLSDP framework. In pursuit of enhancing the defect prediction accuracy of the SoMLDP model, this paper introduces a hybrid swarm optimization algorithm (SOA) referred to as the gravitational force Lévy flight grasshopper optimization algorithm-artificial bee colony (GFLFGOA-ABC) algorithm. By combining the enhanced exploration feature of the gravitational force Lévy flight grasshopper optimization algorithm (GFLFGOA) with the robust exploitation capacity of the artificial bee colony (ABC), the GFLFGOA-ABC algorithm is proposed. Prior to validating the HSoMLSDP framework, the LFGFGOA-ABC algorithm's performance is first confirmed by experiments on 6 benchmark functions (BFs) to assess its mean and convergence rate. Following BF verification, the second experiment tunes the hyperparameters of ML models (ANN, GB, XGB) to improve the defect accuracy of the SoMLDP model. As an enhancement of accuracy justifies the correctness of the SoMLDP model, thus validating the HSoMLSDP framework.**

***Index Terms*—Swarm-optimization Algorithm, Benchmark Functions, Machine Learning, Software Defect Prediction.**

I. Introduction

From the past few decades, almost in every sector, the usage of safety-critical system (SCS), safety-critical system software etc., are increasing day by day. As software is a major part of all these systems, identifying and correcting software defects is crucial for delivering a reliable SCS. Every fault indicated possible points of failure, where the program might malfunction, behave erratically, or crash. Getting more probable defects based on the historical datasets can significantly make the system more robust. For defect prediction, a lot of probabilistic models, and simulators are available, but the current trend is to use computationally intelligent approaches such as machine learning (ML), deep learning (DL), metaheuristic algorithms (MAs), etc.

In this research work, we aimed to design a software defect prediction framework focusing on the improvisation of the swarm-optimized machine learning defect prediction model. Even though several MAs have been developed, every algorithm has certain drawbacks. Inspired by the above reasons, we combined the powerful exploitation characteristic of the ABC with the gravitational force (GF) and Lévy flight (LF) concepts of the GOA to create a hybrid algorithm. In accordance with our research work on the SCS, widely used NASA metrics data program datasets [1] are considered, as the software developed for NASA's operation is safety critical. The following are the main contributions of the proposed work.

- Design of hybrid swarm-optimized machine learning software defect prediction (HSoMLSDP) framework to predict the software defects.
- Design of swarm-optimized machine learning defect prediction (SoMLDP) model by integrating ML models with novel hybrid swarm optimization algorithms (SOAs).
- Design of a gravitational force Lévy flight grasshopper optimization algorithm-artificial bee colony (GFLFGOA-ABC) algorithm to accelerate convergence rate.
- Extensive experiments on benchmark functions (BFs) to verify the effectiveness of the GFLFGOA-ABC algorithm.
- Defect Prediction by validating SoMLDP model using hyperparameter tuning approach.

This paper is structured into the following sections: The literature review is covered in section II, and the background concepts used in the proposed methods are briefly summarized in section III. Section IV presents the methodology. Following this, section V presents verification of the proposed algorithm along with their results and analysis. Section VI discusses results and their analysis of the validation of the proposed framework. Lastly, Section VII presents a conclusion with future scope.

II. Literature survey

This section outlines several relevant studies that employ diverse ML methodologies in addressing the software defect prediction (SDP) challenge. Sharma et al. in [2] developed a model for SDP and improved the prediction using hyperparameter tuning in a ML classifier. To validate the approach, the CM1, JM1, and KC1 defect datasets were considered. Mehmood et al. [3] implemented feature selection techniques in various ML techniques to achieve high accuracy considering 5 NASA defect datasets and compared the ML techniques without feature selection. Suryadi et al. in [4] employed the SMOTE to achieve balance within a dataset comprising 13

NASA MDP instances while utilizing a genetic algorithm (GA) as a methodology for feature selection and hyperparameter optimization to enhance the efficacy of the random forest classifier.

The authors in [5] emphasize the importance of SDP for enhancing software dependability. Through their study, they demonstrated the effectiveness of various ML techniques. In [6], Ali et al. presented an intelligent ensemble-based software defect model that integrates various classifiers. To identify faulty modules, the suggested model used a two-stage prediction approach. It was verified using seven historical defect datasets from the NASA MDP repository. Goyal et. al. [7] undertake a comprehensive assessment and comparative analysis of ML models particularly random forest (RF) and logistic regression (LR) in conjunction with and without GA driven feature selection. The result of their research indicate the accuracy enhancement of NASA defect dataset for SDP. Kelkar et. al [8] assessed the performance of five ML models with and without grey wolf optimization (GWO) based feature selection method and demonstrated the effectiveness of GWO in enhancing SDP for KC1, JM1 and PC5 dataset.

To the best of our understanding, the aforementioned literature review indicates that there exists potential for enhancement in the performance metrics associated with ML models for SDP. So, this observation motivates us to optimize the swarm-optimized ML defect prediction model through the methodology of hyperparameter tuning.

III. Background and preliminaries

The base SOAs that are used to design a novel hybrid SOA are briefly described in this section. Table I lists down all the symbols and their meaning.

A. Grasshopper Optimization Algorithm (GOA)

Saremi et al. [9] introduced the GOA as a method for addressing optimization challenges, which is inspired by the collective behavior exhibited by grasshopper swarms. The original mathematical model of GOA, as detailed in [9], encounters inefficiency as the grasshopper may reside within the comfort zone, which results in inefficient convergence. So, to enhance convergence efficiency, specific parameters have been incorporated into an original mathematical model. Thus, the original GOA equation can be reformulated as follows:

$$\mathrm{x}_i^{sd} = \left(\sum_{j=1, j\neq i}^{n} C \frac{UB_{sd} - LB_{sd}}{2} fs(\left|x_j^{sd} - x_i^{sd}\right|) \frac{x_j - x_i}{sd_{ij}} \right) C + \hat{T_{sd}} \tag{1}$$

The parameter C undergoes adjustment to mitigate exploration while enhancing exploitation in alignment with the number of iterations, as indicated in the equation 2.

$$C = C_{max} - iter \frac{C_{max} - C_{min}}{iter_{max}} \tag{2}$$

TABLE I: Notations of the equations

Symbols	Explanation
x_i and x_j	i^{th} and j^{th} swarm position
n	Swarm Size/ Population Size.
sd_{ij}	i^{th} to j^{th} swarm's distance.
fs	Social Force Strength.
UB_{sd} and LB_{sd}	Upper bound and lower bound of the sd^{th} dimension of the i^{th} swarm, respectively.
$\hat{T_{sd}}$	Optimal position in the sd^{th} dimension.
C	Adaptive Parameter.
C_{max}	Maximum value (1).
C_{min}	Minimum value (0.00001).
$iter$	Current iteration.
$iter_{max}$	Maximum iteration.
$x_{i,j}$	Position of the new food source of the i^{th} swarm/bee at j^{th} dimension.
β	Random number[0, 1].
i	{1,2....n} Selected from swarm size.
j	{1,2....sd} Random dimension index.
sd	Dimension in the optimization process.
UB_j and LB_j	Upper bound and lower bound of the j^{th} parameter, respectively.
K	Random number [-1,1].
$v_{i,j}$	New candidate solution.
$x_{k,j}$	Position of the k^{th} randomly selected candidate solution in j^{th} dimension and $k \neq j$ for proper exploitation.
$fitness(x_i)$	Fitness value of i^{th} solution
Pr_i	Probability of i^{th} food selection.
$Levy\_S$	Step length for the random walk.
gr	Gravitational constant (0.9).
$\hat{e_{gr}}$	$\frac{x_j - x_i}{sd_{ij}}$ Unit Vector.

B. Artificial Bee Colony (ABC) Algorithm

ABC is a swarm-based optimization algorithm, which is developed by Karaboga et al. [10]. The mathematical model of this algorithm is as follows:

- Step 1: The first step includes initialization of the population (x_i) and the random generation of food sources using equation 3.

$$x_{i,j} = LB_j + \beta(UB_j - LB_j) \tag{3}$$

- Step 2: Fitness evaluation of all swarms (x_i).
- Step 3: Employed Bee Phase - In this phase, the neighborhood of each solution is exploited through a local search in the dimension j to find a better solution $v_{i,j}$ from the randomly generated food source, as defined in the equation 4.

$$v_{i,j} = x_{i,j} + K_{i,j}(x_{i,j} - x_{k,j}) \tag{4}$$

- Step 4: Onlooker Bee Phase - The onlooker bees get information from the employed bee and select the food source or solution by estimating the probability value (Pr_i) using the equation 5.

$$Pr_i = \frac{fitness(x_i)}{\sum_{i=1}^{n} fitness(x_i)} \tag{5}$$

The (Pr_i) value is compared with β value. If β < Pr_i, then the solution is updated. A solution with higher

probability value is selected and memorized for better exploration.

- Step 5: Scout Bee Phase - The employed bee becomes a scout bee when the food supply is not selected. So new food sources by the scout bees will be generated by using equation 3.

IV. Methodology

This section focuses on the performance of the HSoMLSDP framework as presented in Fig. 1.

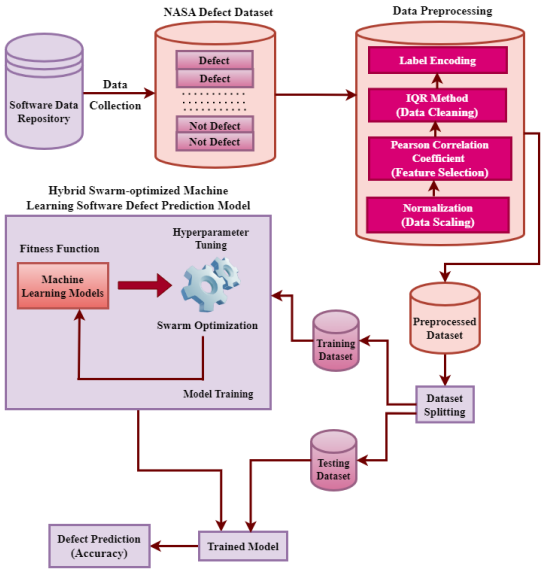


Fig. 1: HSoMLSDP Framework

The methodical process followed to achieve the objective of our framework is carried out in two major phases.

- Data Collection and Preprocessing
- Swarm-optimized machine learning defect prediction (SoMLDP) model

A. Data Collection and pre-processing

The data collection process succeeded by data pre-processing is the preliminary phase of our proposed framework, as illustrated in Fig. 1. For this research work, we utilize seven defect datasets from NASA [1] as enumerated in Table II to assess the robustness and efficacy of the HSoMLSDP framework.

Data pre-processing is essential to prevent biases towards specific features before processing the datasets into the SoMLDP model. This study uses label encoding, data cleaning, feature selection, and normalization for data pre-processing. Yes (Y) and No (N) are mapped to 0 and 1, respectively, in label encoding. Inter-quartile range (IQR) method is followed for data cleaning, while for feature selection using Pearson's correlation coefficient, we removed highly correlated independent features, which is tabulated in Table II. Finally, the min-max scaler approach is performed for normalization.

[1]NASA Defect Dataset https://github.com/klainfo/NASADefectDataset

TABLE II: NASA Defect Dataset Description

Dataset	Description	original Features	Selected Features	Instances
CM1	Software for instruments aboard spacecraft	38	26	327
KC1	Ground data processing system of the spacecraft	22	15	2107
KC4	Spacecraft operations software system	41	35	125
MW1	Satellite-based ground data system for meteorology	38	28	250
PC2	Altitude regulation software system of spacecraft	37	25	722
PC3	Earth orbiting satellite's flight software.	38	25	1053
PC5	Satellite's ground software system	39	26	1694

After pre-processing, the data is split into 75-25 ratios for training and testing respectively.

B. SoMLDP Model

By merging the SOAs with the ML models (ANN, XGB, GB), the SoMLDP model is proposed. This model seeks to maximize computational efficiency while improving accuracy and detection rates.

1) Proposed GFLFGOA-ABC Algorithm: A novel hybrid SOA is proposed in this research work to improve the performance of the SoMLDP model. In the original GOA, as explained in the subsection III-A, the parameter "C" was incorporated to achieve a balance between local exploitation and global exploration. Nevertheless, the original GOA exhibits a nonlinear optimization process that suffers from restricted exploratory capacity. As a consequence, GOA may stuck in local optima, leading to a slow convergence. Therefore, to mitigate the shortcomings in the original GOA, Das et al. [11] proposed GFLFGOA algorithm as presented in equation 6, where LF and GF concepts are embedded into it.

$$x_i^d = C\left(\sum_{j=1,j\neq i}^{n} C\frac{UB_{sd}-LB_{sd}}{2} fs(|x_j^{sd}-x_i^{sd}|)\frac{x_j-x_i}{sd_{ij}}\right) levy\_S - gr\hat{e}_{gr} + \hat{T_{sd}} \quad (6)$$

Considering GFLFGOA as explained above and ABC algorithm as described in subsection III-B, a novel hybrid GFLFGOA-ABC algorithm is proposed. The faster convergence of GFLFGOA and the good exploitation capacity of ABC are combined based on the probabilistic selection mechanism. The ABC will be chosen for generating the fitness solution if rand $\geq$ 0.5, whereas the GFLFGOA is considered for rand $<$ 0.5. In the case of the ABC algorithm, the demerit is its weak exploration, as the search process of this algorithm may lose its diversification by focusing only on the local area. In the ABC algorithm, the exploration process is carried out by the scout bee phase. As weak exploration may

return sub-optimal solutions by converging prematurely, so to overcome weak exploration, scout bee is not considered while proposing the GFLFGOA-ABC algorithm. In the case of ABC algorithm selection, the new position of the search agent is evaluated based on 50 percentage probability using a random number (P_i) that is generated between (0,1). So if rand $< P_i$, it follows the employee bee phase, else follows the onlooker bee phase. The above-mentioned explanation is detailed in the pseudocode of the Algorithm 1. Before

Algorithm 1: GFLFGOA-ABC

Data:
$iter_{max}$: Maximum iterations
n: Population Size / Swarm Size
C: Adaptive Parameter
Result: x_{best}(Best position), g_f(Best fitness value)

```
Initialize swarm population x_i(i = 1, 2, ...n), and
  parameters (iter_max, n, c_max, c_min);
Each swarm fitness value assessment;
Obtain the present best individual (x_best);
while ( iter < iter_max) do
    Update C (equation 2);
    for i= 1:n do
        if (rand < 0.5) then
            Distance (sd) normalization [1,4];
            Update position (equation 6);
            If the swarm exceed the boundary,
              eliminate it;
        else
            if (rand < P_i) then
                Update position (v_{i,j}) (equation 4);
                if (v_{i,j} > x_i) then
                    x_i = v_{i,j}
                end
            else
                Generate (Pr_i) for x_i (equation 5);
                if (β < Pr_i) then
                    Update Position (equation 4);
                end
            end
        end
    end
    Set the current best position;
    Update x_best if it is superior to before;
    iter = iter + 1;
end
return x_best, g_f
```

implementing the proposed GFLFGOA-ABC algorithm to validate the HSoMLSRA framework, first the goodness of the GFLFGOA-ABC is verified by conducting experiments on the benchmark functions (BFs) as detailed in the section V. The symbols used in above equations are listed in the Table I along with their meanings.

2) Parameter Settings of ML models: Model training heavily relies on hyperparameter tuning. The selected hyperparameters for the ML models are listed in Table III.

TABLE III: Hyperparameters of ML Models

ML Models	Hyperparameters
Artificial Neural Network (ANN)	Learning rate, Neurons in layer1 and layer 2, Batch size, Epochs.
XGBOOST (XGB)	Learning rate, Max depth, Subsample, N estimator.
Gradient Boosting (GB)	Learning rate, Max depth, N estimator.

V. Verification of GFLFGOA-ABC on BFs

To verify the robustness of the GFLFGOA-ABC, as detailed in subsubsection IV-B1, experiments are carried out on BFs. In this work, a total of 6 BFs are randomly selected for statistical evaluation. A detailed description of BFs and the experimental results are presented in subsections V-A and V-B, respectively.

A. BFs Description

BFs act as artificial problems that can be utilized to measure the operational characteristics and efficiency of optimization algorithms in various intricate scenarios. A total of 6 BFs, namely Schwefel 2.21 (fn_1), shifted Schwefel 1.2 (fn_2), sphere (fn_3), easom (fn_4), Beale (fn_5), and Ackley (fn_6) are considered randomly to evaluate the performance of GFLFGOA-ABC algorithms. Out of which fn_1-fn_3 are unimodal and the remaining are of multimodal BFs.

B. Results and Discussion

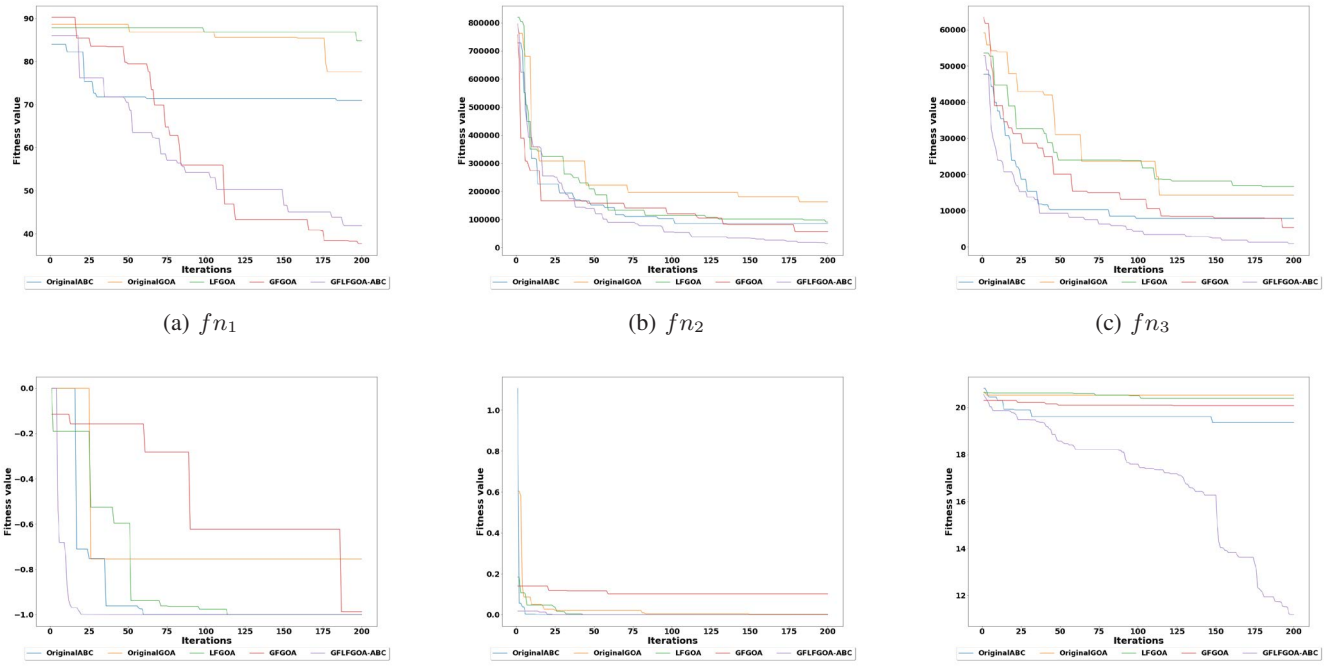
This subsection explains in detail about the results and their analysis that are generated from the experiments carried out on BFs. The proposed GFLFGOA-ABC algorithm, as explained in subsubsection IV-B1, is written in Python and uses the Jupyter notebook environment to compile.

1) Results: Table IV represents the mean value and their ranking, after conducting experiments for 200 iterations. The convergence graphs of BFs for 200 iterations are plotted in Fig. 2.

2) Discussion: The mean values, together with their respective rankings, are considered to assess the statistical efficacy of the GFLFGOA-ABC algorithm on BFs. The highest position, denoted in bold, is achieved by obtaining the minimal mean value for each BF. From Table IV, it can be clearly asserted that the GFLFGOA-ABC algorithm exhibits superior performance for fn_1-fn_6. In addition to attaining the lowest mean value, the GFLFGOA-ABC algorithm exhibits a more rapid convergence rate as compared with the base algorithms, as illustrated in Fig. 2. Through unimodal BFs, one can assert a considerable capacity for exploitation, whereas the capacity for exploration is assessed through multimodal BFs. Therefore, based on the preceding discourse, one can substantiate the efficacy of the GFLFGOA-ABC algorithm.

TABLE IV: Mean value and ranking of five SOAs

BF	Index	GFLFGOA-ABC	LFGOA	GFGOA	GOA	ABC
fn_1	Mean	**5.80E+01**	8.72E+01	5.98E+01	8.57E+01	7.28E+01
	Rank	**1**	5	2	4	3
fn_2	Mean	**1.08E+05**	1.74E+05	1.35E+05	2.38E+05	1.42E+05
	Rank	**1**	4	2	5	3
fn_3	Mean	**7.71E+03**	2.43E+04	1.64E+04	2.57E+04	1.23E+04
	Rank	**1**	4	3	5	2
fn_4	Mean	**-9.68E-01**	-8.27E-01	-4.57E-01	-6.59E-01	-8.89E-01
	Rank	**1**	3	5	4	2
fn_5	Mean	**1.65E-03**	9.36E-03	1.09E-01	2.35E-02	6.66E-03
	Rank	**1**	3	5	4	2
fn_6	Mean	**1.69E+01**	2.05E+01	2.01E+01	2.05E+01	1.96E+01
	Rank	**1**	4	3	4	2



(a) fn_1 (b) fn_2 (c) fn_3

(d) fn_4 (e) fn_5 (f) fn_6

Fig. 2: Convergence Rate of Benchmark Functions (BFs)

VI. Validation of HSoMLSDP Framework

To validate the HSoMLSDP framework, experiments are conducted using seven NASA defect datasets as tabulated in Table II. The experiments are conducted on a 64-bit Ubuntu operating system with a hardware configuration of 32GB RAM, 512GB hard disk, and an Intel core i7 processor. The results and their analysis are discussed in detail in subsection VI-A and VI-B, respectively.

A. SoMLDP Results

After the experiments are conducted for 100 generations, the accuracy results are noted in Table V.

B. SoMLDP Analysis

In the context of our research work, the concept of fitness function employed in the SOA is comparable to the accuracy metrics. As presented in Table V, it is evident that across all datasets, there exists an enhancement in accuracy ranging from approximately 0.01 to 0.28 when the ML models (ANN, XGB, GB) are optimized using the SOAs. Furthermore, as documented in Table V, for the XGB model, the GFLFGOA-ABC algorithm exhibits superior optimization performance in comparison to the baseline algorithms for the KC1, MW1, PC3, and PC5 datasets. In hyperparameter optimization for the ANN model, the GFLFGOA-ABC algorithm demonstrates enhanced accuracy for the KC1, MW1, PC2, and PC5 datasets. For hyperparameter tuning of GB model, GFLFGOA-ABC performs better accuracy for KC1, MW1, PC2, PC5, and PC3. Overall, there is an enhancement of approximately 0.06-0.28, 0.01-0.05, and 0.02-0.10 for ANN, GB, and XGB, respectively, when tuning with SOAs. Based on the aforementioned analysis, it can be inferred that our SoMLDP model demonstrates a distinct advantage in terms of accuracy.

TABLE V: Accuracy Comparision

Dataset	Models	Without Optimization	GFLFGOA-ABC	LFGOA	GFGOA	GOA	ABC
CM1	XGB	0.89024	**0.93902**	0.92683	0.92683	0.92683	**0.93902**
	ANN	0.87805	**0.93902**	0.91463	**0.93902**	0.91463	**0.93902**
	GB	0.90909	0.91919	0.91919	0.91919	0.91919	**0.92929**
KC1	XGB	0.86338	**0.88046**	0.86717	0.87856	0.86907	0.87856
	ANN	0.84820	**0.87856**	0.87287	0.87097	0.87097	0.87476
	GB	0.84676	**0.87362**	0.86888	0.86730	0.87046	0.87204
KC4	XGB	0.68750	**0.78125**	0.71875	0.78125	0.75000	**0.78125**
	ANN	0.56250	**0.84375**	0.71875	**0.84375**	0.71875	0.71875
	GB	0.63158	**0.68421**	**0.68421**	**0.68421**	**0.68421**	**0.68421**
MW1	XGB	0.90476	**0.92063**	0.90476	0.90476	0.90476	0.90476
	ANN	0.88889	**0.95238**	0.93651	0.93651	0.93651	0.93651
	GB	0.86667	**0.89333**	0.88000	0.88000	0.86667	0.88000
PC2	XGB	0.98895	**0.99448**	**0.99448**	**0.99448**	**0.99448**	**0.99448**
	ANN	0.98895	**1.00000**	0.99448	0.99448	0.99448	0.99448
	GB	0.99539	**1.00000**	0.99539	0.99539	0.99539	0.99539
PC3	XGB	0.86364	**0.88258**	0.87879	0.87879	0.87500	0.87879
	ANN	0.82576	**0.87879**	0.87500	**0.87879**	0.85606	**0.87879**
	GB	0.84494	**0.87658**	0.86392	0.87025	0.85759	0.87025
PC5	XGB	0.77830	**0.82075**	0.81368	0.81604	0.81132	0.81840
	ANN	0.74057	**0.81182**	0.80660	0.80660	0.80660	0.80896
	GB	0.77407	**0.80157**	0.79371	0.79371	0.78585	0.79568

VII. Conclusion and Future Scope

In this paper, we proposed a HSoMLSDP framework to predict software defect of NASA defect datasets, which were generated from safety critical softwares. Inside HSoMLSDP framework, SoMLDP model is proposed by integrating the ML models (ANN, XGB, GB) with the novel hybrid SOA. To validate the HSoMLSDP framework, seven NASA defect datasets are trained and tested into the SoMLDP model. To enhance the SoMLDP model, hyperparameter tuning is applied by proposing GFLFGOA-ABC algorithm. This proposed HSoMLSDP framework facilitates SDP and is found to outperform state-of-the-art approach for NASA defect datasets. Future scope includes validation of HSoMLSDP framework for other SDP datasets and DL models.

References

[1] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data quality: Some comments on the nasa software defect datasets," *IEEE Transactions on software engineering*, vol. 39, no. 9, pp. 1208–1215, 2013.

[2] D. N. Sharma and D. K. Yadav, "Machine learning based approach for software defect prediction using hyperparameter," 2024.

[3] I. Mehmood, S. Shahid, H. Hussain, *et al.*, "A novel approach to improve software defect prediction accuracy using machine learning," *IEEE Access*, vol. 11, pp. 63 579–63 597, 2023.

[4] M. K. Suryadi, R. Herteno, S. W. Saputro, M. R. Faisal, and R. A. Nugroho, "Comparative study of various hyperparameter tuning on random forest classification with smote and feature selection using genetic algorithm in software defect prediction," *Journal of Electronics, Electromedical Engineering, and Medical Informatics*, vol. 6, no. 2, pp. 137–147, 2024.

[5] B. Alsangari and G. BİRCİK, "Performance evaluation of various ml techniques for software fault prediction using nasa dataset," in *2023 5th International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, IEEE, 2023, pp. 1–7.

[6] M. Ali, T. Mazhar, Y. Arif, *et al.*, "Software defect prediction using an intelligent ensemble-based model," *IEEE Access*, 2024.

[7] G. Goyal, K. Sharma, V. Mittal, B. Singla, M. Das, *et al.*, "Hybrid genetic algorithm and machine learning approach for software reliability assessment in safety-critical systems," in *2024 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, IEEE, vol. 2, 2024, pp. 1–6.

[8] S. Kelkar, S. P. Vishvasrao, A. Agarwal, C. Rajput, M. Das, *et al.*, "Comparative analysis of software reliability using grey wolf optimisation and machine learning," in *2024 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, IEEE, vol. 2, 2024, pp. 1–5.

[9] S. Saremi, S. Mirjalili, and A. Lewis, "Grasshopper optimisation algorithm: Theory and application," *Advances in engineering software*, vol. 105, pp. 30–47, 2017.

[10] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (abc) algorithm," *Journal of global optimization*, vol. 39, pp. 459–471, 2007.

[11] M. Das, B. R. Mohan, R. M. R. Guddeti, and N. Prasad, "Hybrid bio-optimized algorithms for hyperparameter tuning in machine learning models: A software defect prediction case study," *Mathematics*, vol. 12, no. 16, p. 2521, 2024.