Software Defect Prediction Using Integrated Logistic Regression and Fractional Chaotic Grey Wolf Optimizer

Raja Oueslati[1] and Ghaith Manita[1,2]
[1]*Laboratory MARS, LR17ES05, ISITCom, Sousse University, Sousse, Tunisia*
[2]*ESEN, Manouba University, Manouba, Tunisia*

Keywords: Software Defect Prediction, Metaheuristic Algorithm, Grey Wolf Optimizer, Fractional Chaotic, Logistic Regression.

Abstract: Software Defect Prediction (SDP) is critical for enhancing the reliability and efficiency of software development processes. This study introduces a novel approach, integrating Logistic Regression (LR) with the Fractional Chaotic Grey Wolf Optimizer (FCGWO), to address the challenges in SDP. This integration's primary objective is to overcome LR's limitations, particularly in handling complex, high-dimensional datasets and mitigating overfitting. FCGWO, inspired by the social and hunting behaviours of grey wolves, coupled with the dynamism of Fractional Chaotic maps, offers an advanced optimization technique. It refines LR's parameter tuning, enabling it to navigate intricate data landscapes more effectively. The methodology involved applying the LR-FCGWO model to various SDP datasets, focusing on optimizing the LR parameters for enhanced prediction accuracy. The results demonstrate a significant improvement in defect prediction performance, with the LR-FCGWO model outperforming traditional LR models in accuracy and robustness. The study concludes that integrating LR and FCGWO presents a promising advance in SDP, offering a more reliable, efficient, and accurate approach for predicting software defects.

1 INTRODUCTION

Software defect prediction (SDP) (Roman et al., 2023) plays a pivotal role in enhancing the reliability and efficiency of software development processes. By anticipating potential defects early in the software lifecycle, SDP aids in allocating resources effectively and mitigating risks associated with software failures. Despite its significance, accurately predicting software defects remains challenging, primarily due to the complex and dynamic nature of software development environments.

Among the array of machine learning (ML) techniques, logistic regression (Acito, 2023) stands out as one of the most utilized algorithms in SDP. Its popularity stems from its simplicity and interpretability. Logistic regression is particularly adept at handling binary classification problems, like distinguishing between defect-prone and defect-free software modules. It estimates the probabilities of classes, providing clear insights into the likelihood of defects. This aspect is invaluable in SDP, where understanding the risk of defects is as crucial as identifying their presence.

Logistic regression models typically rely on gradient descent for parameter optimization (Manita et al., 2023). Gradient descent is a widely used approach for minimizing the cost function, guiding the model towards the most accurate predictions. However, this method has limitations, especially in navigating complex, multidimensional landscapes typical in SDP. Gradient descent can get trapped in local minima, particularly with non-convex error surfaces, leading to suboptimal parameter values and, consequently, less accurate defect predictions.

Metaheuristic algorithms (Talbi, 2009) emerge as a potent alternative to gradient descent for parameter optimization in logistic regression models. Unlike gradient descent, which incrementally adjusts parameters based on the local gradient, metaheuristics explore the parameter space more broadly and creatively. For instance, algorithms such as Genetic Algorithm (GA) (Holland, 1992), Particle Swarm Optimization (PSO) (Eberhart and Kennedy, 1995), or Grey Wolf Optimizer (GWO) (Mirjalili et al., 2014) simulate natural processes to search for optimal or near-optimal solutions. These methods excel in finding global optima in complex, irregular search spaces

Oueslati, R. and Manita, G.
Software Defect Prediction Using Integrated Logistic Regression and Fractional Chaotic Grey Wolf Optimizer.
DOI: 10.5220/0012704600003687

In *Proceedings of the 19th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2024)*, pages 633-640
ISBN: 978-989-758-696-5; ISSN: 2184-4895

by balancing exploration (searching new areas) and exploitation (refining known good areas).

By employing metaheuristic algorithms for parameter optimization, logistic regression models in SDP can overcome some of the inherent limitations of gradient descent. These algorithms enhance the model's ability to navigate complex, high-dimensional data landscapes, efficiently identifying parameter combinations that yield the most accurate predictions. This approach is particularly valuable in handling the intricate and often non-linear relationships present in software defect datasets, thereby improving the robustness and reliability of SDP models.

To address these limitations, integrating metaheuristic algorithms like GA, PSO, and GWO has become increasingly popular. These algorithms excel in optimizing ML model parameters and feature selection (Manita et al., 2012; Johnson et al., 2014; Qasim and Algamal, 2018; Panda and Azar, 2021). They enhance model performance, reduce complexity, and ensure models effectively handle imbalanced datasets, which is crucial in SDP scenarios.

This synergy between ML and metaheuristic optimization offers an advanced approach to SDP. By combining the predictive capabilities of ML with the optimization strengths of metaheuristics, the field moves towards more accurate, efficient, and reliable defect prediction models. The LR-FCGWO approach is poised to offer significant advantages in SDP. By harnessing the enhanced search capabilities of Fractional Chaotic maps, LR-FCGWO can outperform traditional algorithms in terms of accuracy and computational efficiency. This approach is particularly promising in addressing the challenges of high-dimensional data and complex feature spaces commonly encountered in SDP.

This paper aims to explore the application of the LR-FCGWO approach in the context of SDP. We seek to investigate its prediction accuracy and computational efficiency performance, comparing it with existing metaheuristic algorithms. The remainder of this paper is structured as follows. In the second section, we present an overview of related work. In the third section, we introduce the GWO algorithm. In the fourth section, we define the proposed approach FCGWO. In the fifth section, we presents the LR-FCGWO to predict software defects. In the sixth, we evaluate the performance on the proposed approach on different datasets. We conclude with a conclusion and some future work.

2 RELATED WORK

In recent studies, metaheuristic optimization techniques like Particle Swarm Optimization (PSO) have been effectively applied to software defect prediction. (Buchari et al., 2018) utilized Chaotic Gaussian PSO (CGPSO) on 11 public benchmark datasets, showing improved accuracy in the majority of cases. (Jin and Jin, 2015) combined Quantum PSO (QPSO) with a hybrid Artificial Neural Network (ANN) for predicting defects, demonstrating its efficacy in correlating software metrics with fault-proneness, thus potentially reducing maintenance costs and enhancing development efficiency. Furthermore, (Wahono et al., 2014) applied Genetic Algorithms and PSO for feature selection, alongside bagging to address class imbalance, significantly enhancing prediction accuracy over traditional methods.

In (Panda and Azar, 2021), the Grey Wolf Optimizer (GWO) is used for feature selection to enhance bug detection classifiers. This method, applied to multiclass bug severity classification using MLP, LR, and RF techniques, demonstrates improved prediction accuracy on the Ant 1.7 and Tomcat datasets. The study highlights the potential of GWO in advancing software quality by efficiently identifying and classifying bugs.

The authors in (Kang et al., 2021) demonstrate the effectiveness of Harmony Search (HS) optimization in Just-in-Time software defect prediction (JIT-SDP), particularly for safety-critical maritime software. By optimizing model parameters with HS, their approach surpasses traditional models in predicting defects at the commit level, showing improved accuracy and recall rates across various datasets. This highlights the potential of HS in enhancing software quality assurance through better management of class imbalances and prediction performance. As reported in (Elsabagh et al., 2020), the authors employ the Spotted Hyena Optimizer algorithm with a multi-objective fitness function for defect prediction in cross-project scenarios. This approach aims to improve classification accuracy where historical data is sparse and varied. Testing on NASA datasets—JM1, KC1, and KC2—the algorithm outperforms traditional data mining techniques, achieving accuracy rates of 84.6% for JM1, 92.0% for KC1, and 82.4% for KC2, highlighting its effectiveness in defect prediction.

In (Niu et al., 2018), the study introduces an adaptive, multi-objective Cuckoo Search algorithm aimed at enhancing defect prediction accuracy by optimizing multiple objectives and adaptively adjusting dataset ratios to balance module sizes. This approach outperforms traditional defect prediction mod-

els, with simulations confirming its superior accuracy in defect prediction. The authors in (Zhu et al., 2021) introduce EMWS, a feature selection algorithm combining Whale Optimization Algorithm (WOA) and Simulated Annealing (SA) for software defect prediction. This method significantly improves prediction accuracy by effectively reducing irrelevant and redundant features, showcasing EMWS's efficacy in enhancing model performance.

As reported in (Zivkovic et al., 2023), the metaheuristic employed is a modified variant of the reptile search optimization algorithm, HARSA (Hyperparameter Adjustment using Reptile Search Algorithm). The results show its ability to enhance the performance of the defect prediction model. In (Balogun et al., 2020), metaheuristic search methods used in this study for SDP in the context of Wrapper Feature Selection (WFS) include 11 state-of-the-art metaheuristics and two conventional search methods. The goal is to compare the effectiveness of these methods in addressing the high dimensionality problem and improving the predictive capabilities of models in SDP.

The study (Raheem et al., 2020) investigate the effectiveness of the Firefly Algorithm (FA) and Wolf Search Algorithm (WSA) in feature selection, combined with Support Vector Machine (SVM) and Random Forest (RF) classifiers for software bug prediction. They use public software module datasets to compare the performance of these machine learning techniques. The authors in (Goyal and Bhatia, 2021) explore the Lion Optimization-based Feature Selection (LiOpFS) method, which excels in choosing an optimal feature subset from high-dimensional data, tackling the Curse of Dimensionality. By filtering out only the most relevant features, LiOpFS significantly enhances classifier performance and fault prediction model accuracy.

3 GREY WOLF OPTIMIZER (GWO)

The Grey Wolf Optimizer (GWO), as introduced in (Mirjalili et al., 2014), stands out in the metaheuristic landscape for its unique emulation of the social hierarchy and hunting behaviour of grey wolves. This algorithm encapsulates the wolves' strategy of tracking, encircling, and ultimately attacking the prey. The GWO algorithm is mainly characterized by incorporating the leadership and collaborative dynamics within a wolf pack.

The mathematical model of the GWO initiates with the encircling behaviour, described by:

$$\vec{D} = |\vec{C} \times \vec{X}_p(t) - \vec{X}(t)| \quad (1)$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \times \vec{D} \quad (2)$$

Here, $\vec{X}_p(t)$ denotes the prey's position vector, $\vec{X}(t)$ represents the position vector of a wolf, and t signifies the current iteration. The coefficients $\vec{A}$ and $\vec{C}$ are computed as:

$$\vec{A} = 2 \times \vec{a} \times \vec{r}_1 - \vec{a} \quad (3)$$

$$\vec{C} = 2 \times \vec{r}_2 \quad (4)$$

where $\vec{a}$ linearly decreases from 2 to 0 as iterations progress, and $\vec{r}_1$ and $\vec{r}_2$ are random vectors within the range [0, 1]. The social hierarchy within the wolf pack is such that the alpha (α) wolf's position is considered the current best solution. The beta (β) and delta (δ) wolves follow the alpha, and the rest of the pack (omega wolves) follow these leading members.

The phase of attacking the prey, the exploitation phase, is marked by a reduction in the magnitude of $\vec{a}$, allowing the pack to converge towards the prey's position. This interplay between exploration (high $\vec{a}$) and exploitation (low $\vec{a}$) is a testament to the algorithm's capacity to explore and exploit the search space adaptively, propelling it towards the optimal solution with an effective balance. Figure 1 depicts the flowchart of the original GWO algorithm.

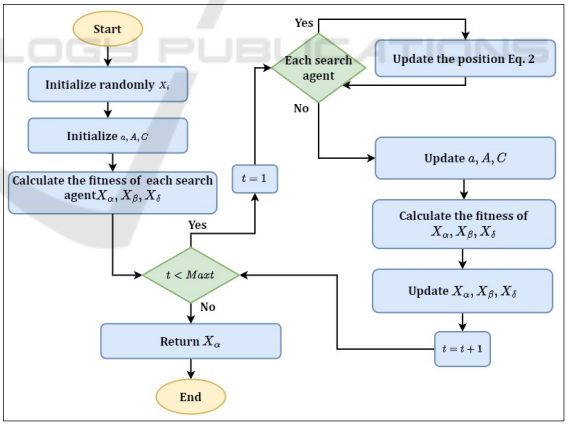

Figure 1: GWO Algorithm: A Detailed Flowchart.

4 PROPOSED APPROACH FCGWO

Achieving a delicate balance between exploration and exploitation is key in optimization algorithms to efficiently navigate complex search spaces. The Fractional Chaotic Grey Wolf Optimizer (FCGWO) innovatively combines fractional order chaos maps with

the traditional GWO to balance exploration and exploitation in complex search spaces. This approach uses the complex dynamics of chaos maps to evade local optima, enhancing search efficiency. The following discussion will detail the mechanisms of fractional chaos maps and FCGWO, illustrating its potential in solving various optimization challenges.

4.1 Fractional Chaotic

This research integrates three advanced fractional order chaos maps (Wu and Baleanu, 2014; Wu et al., 2014; ATALI et al., 2021) into the Grey Wolf Optimizer (GWO) to boost its performance by addressing local optima entrapment and optimizing the exploration-exploitation balance, essential for reaching optimal solutions.

- **Fractional Logistic Map.** Renowned for its intricate dynamical properties, the fractional logistic map is instrumental in enriching the diversity within the search space, thereby facilitating more extensive exploration.
- **Fractional Sine Map.** The map's inherent periodic nature is crucial in ensuring a comprehensive scan of the solution space, enhancing the algorithm's thoroughness.
- **Fractional Tent Map.** This map introduces a unique approach to traversing the optimization landscape, offering novel pathways through the search space.

The integration of these maps (detailed in Table 1) is an addition to the GWO algorithm and a strategic incorporation of complex dynamical systems theory into optimization. By leveraging the intricate behaviours of these chaotic systems, we aim to boost the GWO algorithm's ability to navigate and exploit multidimensional search spaces effectively, leading to more accurate, reliable, and quicker convergence to optimal solutions across various scenarios.

Table 1: Fractional chaos maps with range 0 and 1.

Name	Equation	x(0)	Parameters
Fractional Logistic	$x_{k+1} = x_0 + \frac{\mu}{\Gamma(\alpha)} \sum_{j=1}^{k} \frac{\gamma(k-j+\alpha)}{\gamma(k-j+1)} x_{j-1}(1-x_{j-1})$	0.3	$\mu = 2.5, \alpha = 0.3$
Fractional Sine	$x_{k+1} = x_0 + \frac{\mu}{\Gamma(\alpha)} \sum_{j=1}^{k} \frac{\gamma(k-j+\alpha)}{\gamma(k-j+1)} \sin(x_{j-1})$	0.3	$\mu = 3.8, \alpha = 0.8$
Fractional Tent	$x_{k+1} = x_0 + \frac{1}{\Gamma(\alpha)} \sum_{j=1}^{k}$	0.3	$\mu = 1.9, \alpha = 0.6$
	$\left[\frac{\Gamma(k-j+\alpha)}{\Gamma(k-j+1)} \min((\mu-1)x_{j-1}, \mu-(\mu+1)x_{j-1})\right]$		

4.2 Mechanistic Foundations of FCGWO

The FCGWO builds upon the traditional GWO by replacing the random number generation process with a sequence derived from fractional chaotic maps. In FCGWO, the chaotic sequence is used to update the position and velocity of the search agents (i.e., the grey wolves) in the search space. The following outlines the key steps of the FCGWO algorithm:

1. Initialize the grey wolf population, along with their positions and velocities.
2. Evaluate the fitness of each grey wolf.
3. Calculate the social ranks (i.e., alpha, beta, and delta wolves) based on the fitness values.
4. Update the positions and velocities of the grey wolves using the standard update rules (i.e., Equations (1) and (2)). However, the calculation of the coefficients $\vec{A}$ and $\vec{C}$ are updated using Fractional Chaotic maps instead of random values following the standard behavior. Then, the coefficients $\vec{A}$ and $\vec{C}$ are now computed as:
$$\vec{A} = 2 \times \vec{a} \times \vec{\psi(t)} - \vec{a} \quad (5)$$
$$\vec{C} = 2 \times \vec{\psi(t)} \quad (6)$$
5. Repeat steps 2-4 until a termination criterion is met (e.g., maximum number of iterations, sufficient solution quality).

The introduction of the fractional chaotic sequence $\psi(t)$ in the FCGWO update rules fosters exploration and diversification, enabling the algorithm to avoid local optima and efficiently search complex problem landscapes. The following sections will delve deeper into the potential applications and performance evaluation of the FCGWO algorithm. Figure 2 depicts the flowchart of the proposed FC-GWO algorithm.

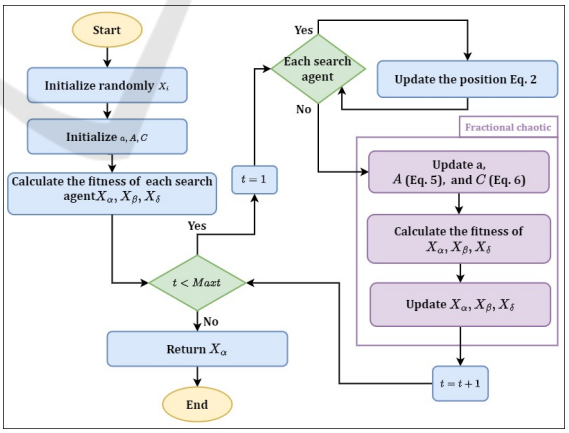

Figure 2: Flowchart of FCGWO Algorithm.

5 LR-FCGWO FOR SOFTWARE DEFECTS PREDICTION

The integration of LR with the FCGWO offers a groundbreaking approach to SDP. LR, a standard

method for binary classification, is especially useful in SDP for its simplicity and interpretability. It calculates the probability of a software module being defect-prone, a crucial aspect of software quality assurance. However, LR faces challenges such as overfitting and inefficiency in high-dimensional data spaces.

The FCGWO, inspired by the social and hunting behaviours of grey wolves and enhanced with the dynamic capabilities of Fractional Chaotic maps, is employed to address these limitations. This advanced metaheuristic algorithm optimizes LR parameters, aiding in tackling complex, high-dimensional datasets prevalent in SDP. Traditional parameter tuning methods often must catch up in such scenarios, leading to overfitting or inadequate feature selection. In contrast, FCGWO's adaptive nature allows for a more refined and practical approach.

An added benefit of FCGWO is incorporating Fractional Chaotic maps, which boosts the algorithm's ability to escape local optima. This feature is particularly valuable in handling the non-linear relationships characteristic of SDP data, which standard LR models may not effectively capture.

5.1 Objective Function

As previously mentioned, we employ the FCGWO algorithm as a training mechanism for the logistic regression model. Specifically, each potential solution, comprising a set of weights and biases produced by FCGWO, is evaluated using the LR model. The performance of each solution (f_i) is then quantified using the discrepancy (E_i) between the predicted (p) and actual (y) outputs, as depicted in the equations below:

$$f_i = \frac{1}{1+E_i} \tag{7}$$

$$E_i = \sum_{1}^{n} (p_i^j - y_i^j)^2 \tag{8}$$

where n is the number of rows used in the training set.

5.2 Data Preprocessing

For our work, we opted for the Min-Max method (Islam et al., 2022), which simplifies value comparisons. Following normalization, we applied a data transformation method, which is an essential part of the initial data preparation before statistical analysis. This method also facilitates comparison and interpretation. Moreover, true (yes) and false (no) values were transformed into 1s and 0s, respectively. We also employed cross-validation, a data resampling technique, to assess the generalization ability of predictive models and prevent overfitting (Berrar, 2019).

5.3 LR-FCGWO Steps

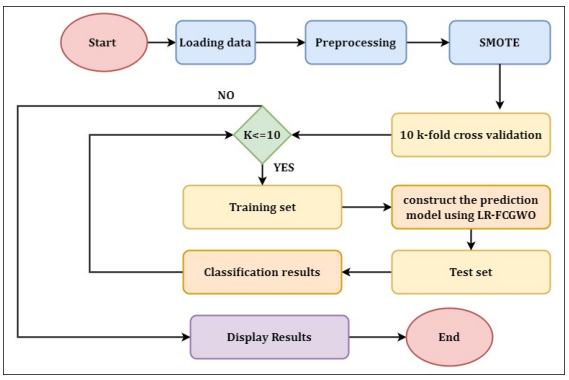

Figure 3: Flowchart of the LR-FCGWO Model for Software Defect Prediction.

This section presents an overview of the LR-FCGWO model's workflow, illustrated in Figure 3. This workflow is designed to systematically process software metrics, apply optimization techniques, and utilize predictive analytics to identify potential defects.

1. **Loading Data.** The process begins by importing the relevant datasets to be used for software defect prediction.
2. **Preprocessing.** Subsequently, data undergoes preprocessing to confirm it is ready for analysis. This includes data cleaning, normalization, and transformation procedures.
3. **SMOTE.** To counteract class imbalance within the datasets, the Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al., 2002) is employed, which helps prevent model bias towards the majority class.
4. **10 k-fold Cross Validation.** Data is partitioned into ten segments, and the model is cross-validated to ensure robustness and to reduce the risk of overfitting. This iterative cross-validation is indicated by the decision diamond 'K¡=10'.
5. **Training Set.** The LR-FCGWO model is then constructed using the training set obtained from the cross-validation process.
6. **Test Set.** The model, now trained, is applied to the test set to perform defect prediction.
7. **Classification Results.** Outcomes of the classification are compiled to reflect the model's performance, as judged by metrics like accuracy, precision, and recall.
8. **Display Results.** The results from the classification process are then displayed.

6 EXPERIMENTAL RESULTS

6.1 Datasets

In this study, five datasets from the PROMISE REPOSITORY (Sayyad Shirabad and Menzies, 2005) are selected, as shown in Table 2, varying in terms of the number of lines and defect rates.

Table 2: Description of the Datasets.

	Number of Attributes	Yes	No	Number of Instances	Number of Missing Values
CM1	22	49	449	498	0
JM1	22	8779	2106	10885	0
KC1	22	326	1783	2109	0
KC2	22	105	415	522	0
PC1	22	1032	77	1109	0

6.2 Results and Discussions

In this section, all experiments were repeated 51 times independently to obtain statistically significant results, implemented using MATLAB R2020a.

The comparative analysis encompassed in the presented tables evaluates a range of machine learning models: Logistic Regression (LR-standard), Random Forest (Breiman, 2001), Gradient Boosting (Natekin and Knoll, 2013), AdaBoost (Freund and Schapire, 1997), Support Vector Machine (SVM) (Kramer and Kramer, 2013), K-Nearest Neighbors (K-NN) (Kramer and Kramer, 2013), Decision Tree (Rokach and Maimon, 2005), XGBoost (Chen and Guestrin, 2016), and Logistic Regression with CFGWO (LR-CFGWO). These models are assessed across various datasets, namely CM1, JM1, KC1, KC2, and PC1, focusing on four key performance metrics: Accuracy, Precision, Recall, and F1 Score.

In the CM1 dataset, LR-CFGWO stands out as the most proficient model, achieving the highest scores across all metrics (Table 3). This model's superior performance is evident in its exceptional balance of precision and recall, leading to a robust F1 Score. Similarly, in the JM1 dataset, the Random Forest model exhibits outstanding performance, marking the highest scores in all evaluated metrics (Table 4). This fact indicates its effectiveness in handling this dataset, especially regarding accuracy and precision.

For the KC1 dataset, LR-CFGWO again shows remarkable performance, topping the charts in all metrics (Table 5). This consistency underscores the model's robustness and adaptability across different datasets. In the analysis of the KC2 dataset, LR-CFGWO and Random Forest models show competitive performance. However, LR-CFGWO slightly edges out with superior accuracy and precision, while Random Forest excels in recall (Table 6), indicating its strength in identifying relevant instances.

Table 3: Comparative Performance Analysis of Various Machine Learning Models on the CM1 Dataset.

Model	Accuracy	Precision	Recall	F1 Score
LR-standard	80.735	80.396	81.292	80.842
Random Forest	92.428	89.278	96.437	92.719
Gradient Boosting	90.646	86.573	96.214	91.139
AdaBoost	86.971	82.937	93.096	87.723
SVM	81.180	79.046	84.855	81.847
K-NN	83.519	75.904	98.218	85.631
Decision Tree	85.635	82.787	89.978	86.233
XGBoost	92.094	88.730	96.437	92.423
LR-CFGWO	**92.984**	**89.549**	**97.327**	**93.276**

Table 4: Comparative Performance Analysis of Various Machine Learning Models on the JM1 Dataset.

Model	Accuracy	Precision	Recall	F1 Score
LR-Standard	66.751	69.031	60.760	64.632
Random Forest	**91.703**	**91.546**	**91.893**	**91.719**
Gradient Boosting	81.957	84.140	78.759	81.361
AdaBoost	73.782	73.991	73.345	73.667
SVM	67.336	71.100	58.418	64.138
K-NN	81.826	75.132	95.145	83.962
Decision Tree	86.560	86.752	86.299	86.525
XGBoost	88.510	92.318	84.010	87.969
LR-CFGWO	87.875	92.210	82.740	87.219

Table 5: Comparative Performance Analysis of Various Machine Learning Models on the KC1 Dataset.

Model	Accuracy	Precision	Recall	F1 Score
LR-Standard	71.901	71.682	72.406	72.042
Random Forest	88.923	85.626	93.550	89.413
Gradient Boosting	84.773	83.120	87.269	85.144
AdaBoost	78.435	74.877	85.586	79.874
SVM	73.472	70.998	79.361	74.947
K-NN	81.716	75.167	94.728	83.821
Decision Tree	83.707	81.833	86.652	84.173
XGBoost	88.839	87.251	90.970	89.072
LR-CFGWO	**89.484**	**88.344**	**90.970**	**89.638**

Table 6: Comparative Performance Analysis of Various Machine Learning Models on the KC2 Dataset.

Model	Accuracy	Precision	Recall	F1 Score
LR-Standard	78.795	78.935	78.554	78.744
Random Forest	88.675	85.275	**93.494**	**89.195**
Gradient Boosting	87.711	85.812	90.361	88.028
AdaBoost	82.892	82.578	83.373	82.974
SVM	78.675	77.674	80.482	79.053
K-NN	81.084	76.113	90.602	82.728
Decision Tree	86.265	86.618	85.783	86.199
XGBoost	87.711	85.488	90.843	88.084
LR-CFGWO	**88.916**	**87.298**	91.084	89.151

Table 7: Comparative Performance Analysis of Various Machine Learning Models on the PC1 Dataset.

Model	Accuracy	Precision	Recall	F1 Score
LR-standard	80.814	79.336	83.333	81.285
Random Forest	95.736	94.444	97.190	95.798
Gradient Boosting	93.750	91.536	96.415	93.912
AdaBoost	89.874	87.240	93.411	90.220
SVM	83.285	77.769	93.217	84.795
K-NN	90.843	85.037	**99.128**	91.544
Decision Tree	91.812	90.065	93.992	91.987
XGBoost	95.785	94.787	96.899	95.831
LR-CFGWO	**96.124**	**95.161**	97.190	**96.165**

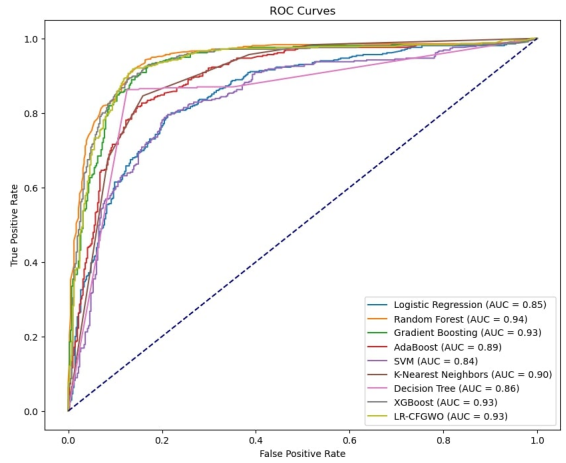

Figure 4: ROC Curve Analysis of Classifier Performance on KC2 Dataset.

In the PC1 dataset, LR-CFGWO again demonstrates its dominance, leading in accuracy, precision, and F1 Score (Table 7). The model's consistent performance across different datasets highlights its effectiveness and reliability in various contexts.

After that, we present the Receiver Operating Characteristic (ROC) curves for different classifiers applied to KC2 dataset. ROC curves are a graphical representation that illustrate the diagnostic ability of binary classifiers as their discrimination threshold is varied. The area under the ROC curve (AUC) provides a single measure of overall performance of a classifier. The closer the ROC curve is to the top-left corner, the higher the overall accuracy of the test.

In Figure 4, the ROC curve for the KC2 dataset indicates that the LR-FCGWO classifier achieves a remarkable AUC of 0.99, paralleling the excellent performance of the Gradient Boosting classifier. The high AUC value underscores the LR-FCGWO classifier's exceptional proficiency in discriminating between the classes.

Overall, the analysis reveals that while different models have their strengths in specific datasets, LR-CFGWO consistently emerges as a top performer, showcasing its versatility and effectiveness in various predictive tasks.

7 CONCLUSIONS

The LR-FCGWO methodology introduces an innovative approach to Software Defect Prediction (SDP), combining the predictive capabilities of Logistic Regression with the optimization strengths of the Fractional Chaotic Grey Wolf Optimizer (FCGWO). This method enhances defect prediction by optimizing model parameters, effectively addressing overfitting and the challenges posed by high-dimensional data. It leads to an improved and more detailed predictive model for detecting software defects.

The potential avenues for further research and application of the LR-FCGWO model are manifold and promising. One prospective area of exploration is the continued refinement and optimization of the FCGWO algorithm itself. Enhancing algorithmic efficiency or adaptability could further elevate the performance of LR-FCGWO, especially in scenarios characterized by exceedingly large or complex datasets.

REFERENCES

Acito, F. (2023). Logistic regression. In *Predictive Analytics with KNIME: Analytics for Citizen Data Scientists*, pages 125–167. Springer.

ATALI, G., PehlIvan, İ., Gürevin, B., and ŞEKER, H. İ. (2021). Chaos in metaheuristic based artificial intelligence algorithms: a short review. *Turkish Journal of Electrical Engineering and Computer Sciences*, 29(3):1354–1367.

Balogun, A. O., Basri, S., Jadid, S. A., Mahamad, S., Almomani, M. A., Bajeh, A. O., and Alazzawi, A. K. (2020). Search-based wrapper feature selection methods in software defect prediction: an empirical analysis. In *Intelligent Algorithms in Software Engineering: Proceedings of the 9th Computer Science Online Conference 2020, Volume 1 9*, pages 492–503. Springer.

Berrar, D. (2019). Cross-validation. *Encyclopedia of bioinformatics and computational biology*, 1:542–545.

Breiman, L. (2001). Random forests. *Machine learning*, 45:5–32.

Buchari, M., Mardiyanto, S., and Hendradjaya, B. (2018). Implementation of chaotic gaussian particle swarm optimization for optimize learning-to-rank software defect prediction model construction. In *Journal of Physics: Conference Series*, volume 978, page 012079. IOP Publishing.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority oversampling technique. *Journal of artificial intelligence research*, 16:321–357.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Eberhart, R. and Kennedy, J. (1995). Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks*, volume 4, pages 1942–1948. Citeseer.

Elsabagh, M., Farhan, M., and Gafar, M. (2020). Cross-projects software defect prediction using spotted hyena optimizer algorithm. *SN Applied Sciences*, 2:1–15.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

Goyal, S. and Bhatia, P. K. (2021). Software fault prediction using lion optimization algorithm. *International Journal of Information Technology*, 13:2185–2190.

Holland, J. H. (1992). Genetic algorithms. *Scientific american*, 267(1):66–73.

Islam, M. J., Ahmad, S., Haque, F., Reaz, M. B. I., Bhuiyan, M. A. S., and Islam, M. R. (2022). Application of min-max normalization on subject-invariant emg pattern recognition. *IEEE Transactions on Instrumentation and Measurement*, 71:1–12.

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

Johnson, P., Vandewater, L., Wilson, W., Maruff, P., Savage, G., Graham, P., Macaulay, L. S., Ellis, K. A., Szoeke, C., Martins, R. N., et al. (2014). Genetic algorithm with logistic regression for prediction of progression to alzheimer's disease. *BMC bioinformatics*, 15:1–14.

Kang, J., Kwon, S., Ryu, D., and Baik, J. (2021). Haspo: Harmony search-based parameter optimization for just-in-time software defect prediction in maritime software. *Applied Sciences*, 11(5):2002.

Kramer, O. and Kramer, O. (2013). K-nearest neighbors. *Dimensionality reduction with unsupervised nearest neighbors*, pages 13–23.

Manita, G., Chhabra, A., and Korbaa, O. (2023). Efficient e-mail spam filtering approach combining logistic regression model and orthogonal atomic orbital search algorithm. *Applied Soft Computing*, 144:110478.

Manita, G., Khanchel, R., and Limam, M. (2012). Consensus functions for cluster ensembles. *Applied Artificial Intelligence*, 26(6):598–614.

Mirjalili, S., Mirjalili, S. M., and Lewis, A. (2014). Grey wolf optimizer. *Advances in engineering software*, 69:46–61.

Natekin, A. and Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21.

Niu, Y., Tian, Z., Zhang, M., Cai, X., and Li, J. (2018). Adaptive two-svm multi-objective cuckoo search algorithm for software defect prediction. *International Journal of Computing Science and Mathematics*, 9(6):547–554.

Panda, M. and Azar, A. T. (2021). Hybrid multi-objective grey wolf search optimizer and machine learning approach for software bug prediction. In *Handbook of Research on Modeling, Analysis, and Control of Complex Systems*, pages 314–337. IGI Global.

Qasim, O. S. and Algamal, Z. Y. (2018). Feature selection using particle swarm optimization-based logistic regression model. *Chemometrics and Intelligent Laboratory Systems*, 182:41–46.

Raheem, M., Ameen, A., Ayinla, F., and Ayeyemi, B. (2020). Software defect prediction using metaheuristic algorithms and classification techniques. *Ilorin Journal of Computer Science and Information Technology*, 3(1):23–39.

Rokach, L. and Maimon, O. (2005). Decision trees. *Data mining and knowledge discovery handbook*, pages 165–192.

Roman, A., Brożek, R., and Hryszko, J. (2023). Predictive power of two data flow metrics in software defect prediction.

Sayyad Shirabad, J. and Menzies, T. (2005). The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa, Canada.

Talbi, E.-G. (2009). *Metaheuristics: from design to implementation*. John Wiley & Sons.

Wahono, R. S., Suryana, N., and Ahmad, S. (2014). Metaheuristic optimization based feature selection for software defect prediction. *J. Softw.*, 9(5):1324–1333.

Wu, G.-C. and Baleanu, D. (2014). Discrete fractional logistic map and its chaos. *Nonlinear Dynamics*, 75(1):283–287.

Wu, G.-C., Baleanu, D., and Zeng, S.-D. (2014). Discrete chaos in fractional sine and standard maps. *Physics Letters A*, 378(5-6):484–487.

Zhu, K., Ying, S., Zhang, N., and 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.

Zivkovic, T., Nikolic, B., Simic, V., Pamucar, D., and Bacanin, N. (2023). Software defects prediction by metaheuristics tuned extreme gradient boosting and analysis based on shapley additive explanations. *Applied Soft Computing*, 146:110659.