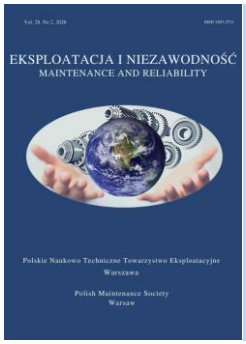


Eksploatacja i Niezawodnosc – Maintenance and Reliability
Volume 28 (2026), Issue 2
journal homepage: http://www.ein.org.pl

Article citation info:
Yüksek G, Optimized GDTP-XGBoost Framework for Wind Power Forecasting Toward Condition-Based Maintenance, Eksploatacja i Niezawodnosc – Maintenance and Reliability 2026: 28(2) http:/10.17531/ein/214376

Optimized GDTP-XGBoost Framework for Wind Power Forecasting Toward Condition-Based Maintenance

Indexed by: Web of Science Group

Gökhan Yüksek[a,*]

[a] Electrical and Electronics Engineering, Batman University, Turkey

Highlights

- A GDTP mechanism is proposed to dynamically control tree complexity in XGBoost.
- DE-Fly combines DE, Firefly, and Mayfly algorithms in a multi-phase hybrid structure.
- The enhanced GDTP-XGBoost improves wind power forecasting accuracy and speed.
- Forecast precision enhances maintenance planning and system-level reliability decisions.

Abstract

Accurate wind energy forecasting is essential for grid stability, energy-demand balance, and the efficient use of renewables. Shallow learning methods are favored for their scalability and generalization ability, yet their performance strongly depends on proper hyperparameter tuning. This study introduces an enhanced XGBoost model with a Gradient-Dynamic Tree Pruning (GDTP) mechanism to control tree complexity adaptively, optimized through a novel DE-Fly hybrid algorithm that integrates Differential Evolution, Firefly Algorithm, and Mayfly Algorithm. Experimental validation using real-world wind power data demonstrates that the proposed DE-Fly–optimized GDTP-XGBoost model achieves superior forecasting accuracy and significantly faster computation than conventional approaches. Beyond predictive performance, the framework provides practical benefits by supporting condition-based maintenance, enabling earlier anomaly detection, minimizing downtime, and enhancing the overall reliability of wind farm operations.

Keywords
maintenance planning, optimization, renewable energy, shallow learning, wind energy forecasting

1. Introduction

The global shift toward sustainable energy has made wind power one of the most significant renewable energy sources (1–3). While weather data and historical generation records are becoming increasingly accessible, forecasting wind energy remains a challenging task. Many machine learning methods have been employed to enhance forecasting accuracy. However, their success often depends on choosing the proper model parameters. The failure of poorly tuned models to deliver the desired performance suggests the need for more effective, flexible forecasting methods that can adapt to various conditions (4,5). From a maintenance perspective, accurate forecasting helps reduce unnecessary load cycling, supports asset health monitoring, and extends system lifetime. It is important for enabling Condition-Based Maintenance (CBM) strategies by providing timely and precise energy output estimates, while supporting Reliability-Centered Maintenance (RCM) frameworks when deeper failure mode insights are required.

Accurate wind energy forecasting is also important for daily operational decisions at wind farms. Short-term forecasts allow operators to better schedule and distribute generated power according to demand. They also help to adjust power

(*) Corresponding author.
E-mail addresses: G. Yüksek (ORCID: 0000-0002-6832-8622) gokhan.yuksek@batman.edu.tr

distribution to meet grid requirements. Forecasts serve as an early warning system for periods of high mechanical stress, allowing teams to plan preventive measures before failures occur. It becomes possible to schedule turbine maintenance according to expected periods of low production. In this way, forecasts become a practical decision support tool that links data-driven modeling with condition-based maintenance practices.

Recent surveys have highlighted the growing influence of artificial intelligence (AI) and big data analytics in the realm of wind energy forecasting. Machine learning and hybrid AI methodologies have been particularly efficacious in managing data heterogeneity and uncertainty (6). Forecasting accuracy is continually being refined by methodological advances, which employ hybrid deep learning and signal decomposition strategies. For instance, Wang et al. in (7) developed an improved Wavenet with multi-head self-attention for multi-step-ahead forecasting, demonstrating significant error reductions compared to classical AI models. Similarly, Nascimento et al. in (8) proposed a transformer-based deep neural network integrated with wavelet decomposition, showing enhanced accuracy and efficiency over Long-Short Term Memory (LSTM) in short-term wind speed and power prediction. Among the comparative studies most relevant to this research, Bouabdallaoui et al. (9) assessed Artificial Neural Networks (ANN), ANFIS, Decision Trees (DT), and Support Vector Machines (SVM), reporting an R^2 value of 0.95. Karaman (10) evaluated Convolutional Neural Network (CNN), LSTM, and Recurrent Neural Network (RNN) models, achieving an R^2 of 0.9574. The nonlinear behaviour of wind power generation is represented well by these advanced learning models. However, it is also highlighted that they are dependent on extensive hyperparameter tuning and considerable computational resources. Also, the fact that these methods are not easily explainable can make them less useful in real-life situations. Consequently, shallow learning approaches retain their high pertinence, delivering expeditious training, enhanced transparency, and dependable performance under conditions of limited data, which are of paramount importance for real-time wind power forecasting.

Shallow learning methods are different from deep learning models because they can be trained quickly, are easier to understand, and work well with limited data. This makes them suitable for use in industrial settings where time is important (11). Extreme Gradient Boosting (XGBoost) is a type of machine learning and is very good at finding nonlinear models and dealing with noisy or incomplete data. However, careful model configuration is essential to fully leverage XGBoost's forecasting capabilities, particularly in critical systems where reliability and uninterrupted performance are vital (12–14). The connection between hyperparameters is the main reason for this situation. These parameters work together to determine how the model aligns with the data, balances bias and variance, and adapts to new situations it has not encountered before. Finding the best mix of different settings is hard because there are so many options, and they all depend on each other (15,16). Conventional approaches, such as grid or random search, are computationally intensive and often prove ineffective in identifying globally optimal configurations, resulting in suboptimal performance. Additionally, their rigid structure hinders their ability to adapt to different datasets or specific requirements, emphasizing the necessity for a more intelligent, adaptive, and automated tuning approach that can systematically enhance XGBoost for various forecasting scenarios (17).

Nature-inspired metaheuristic algorithms are potent tools for hyperparameter optimization because of the limitations of manual and exhaustive search strategies. The mimicking of natural phenomena, such as biological evolution, swarm behaviour, or physical processes, enables efficient optimization of large, complex search spaces (18). Differential Evolution (DE), in particular, has been successfully applied to hyperparameter optimization in machine learning models, demonstrating its ability to improve generalization and prevent overfitting (19–21). Whale Optimization Algorithm (WOA) (22–24), Grey Wolf Optimizer (GWO) (25–27), and Firefly Algorithm (FA) have demonstrated success in various optimization problems by balancing global exploration and local exploitation (28,29). Nevertheless, most single-phase metaheuristics encounter difficulties such as premature convergence, limited fine-tuning capability, or becoming trapped in local optima when confronted with high-dimensional hyperparameter spaces. These limitations highlight the need for more effective hybrid optimization strategies that combine

different algorithmic strengths to achieve superior results (30,31). In recent years, hybrid metaheuristic approaches have been proposed to address the shortcomings of single-phase algorithms. One example is the Mountain Gazelle Optimization (MGO) integrated with the Nelder–Mead (NM) method (MGONM), which combines swarm-inspired exploration with simplex-based local refinement and has shown competitive results (32). Such studies illustrate the promise of hybrid methods in improving convergence reliability when tackling complex optimization problems.

Many existing metaheuristic algorithms demonstrate strong optimization performance, yet their tendency to rely predominantly on either exploration or exploitation often limits their adaptability across diverse problem types. Algorithms biased toward global exploration may struggle with precision in the fine-tuning phase, while those focused on local exploitation risk premature convergence and suboptimal solutions (33–35). This trade-off becomes particularly important when tuning complex machine learning models, where both parameter optimization and dynamic structural control are required.

Managing model complexity during training is a key challenge in developing reliable forecasting systems. This study addresses the issue by introducing a Gradient-Dynamic Tree Pruning (GDTP) mechanism that adjusts tree expansion based on gradient feedback. The goal is to avoid unnecessary growth in the model while maintaining high predictive power. Although such strategies have clear advantages, they are rarely integrated into automated machine learning setups. In this study, to close this gap, a refined XGBoost model enhanced with GDTP and optimized through a hybrid, multi-stage approach is presented. The accompanying optimizer, DE-Fly, operates in phases, first exploring broadly with DE, then narrowing the search using the FA, and finally fine-tuning solutions via the Mayfly Algorithm (MA). This coordination allows both the model structure and its parameters to adapt jointly and effectively. When tested on actual wind energy data, the framework not only achieved stronger forecasting results than traditional methods but also proved computationally efficient, making it well-suited for maintenance scheduling, system reliability assessments, and clean energy integration. The main research gaps addressed in this study are as follows:

Lack of Integrated Hyperparameter and Structural Optimization: Existing studies on XGBoost optimization primarily focus on hyperparameter tuning without addressing the dynamic control of tree structure. Current methods fail to incorporate gradient-based pruning, which can significantly improve model generalization and computational efficiency by preventing unnecessary tree growth.

Limited Adaptation Problem-Specific Search Spaces: Although various optimization techniques have been applied to machine learning models, most rely on static or problem-agnostic search strategies. These methods struggle to adapt to the unique characteristics of wind energy forecasting tasks, which require dynamic exploration of high-dimensional and problem-specific parameter spaces.

Absence of Multi-Phase Optimization Frameworks Combining Exploration, Exploitation, and Fine-Tuning: Existing optimization strategies often merge or overlook the distinction between exploration, exploitation, and fine-tuning phases. This lack of structured coordination hinders their ability to balance global search and local refinement, resulting in suboptimal solutions and inconsistent performance.

Contribution of the Study

This study introduces a novel optimization framework that addresses these gaps by proposing the DE-Fly Optimizer, which sequentially integrates DE for exploration, FA for exploitation, and MA for fine-tuning. This multi-phase optimization strategy tunes XGBoost's conventional hyperparameters and optimizes the GDTP mechanism to control tree complexity adaptively. The proposed framework significantly enhances forecasting accuracy, computational efficiency, and model generalisation.

2. Proposed DE-Fly Method

The proposed DE-Fly method builds upon well-established metaheuristic algorithms, namely Differential Evolution (19), the Firefly Algorithm (36), and the Mayfly Algorithm (37), which are integrated into a coordinated multi-phase framework as shown in pseudo-code below.

The DE method is used in the first stage of the hybrid algorithm. DE provides an advantage in generating optimal solutions in the initial population due to its ability to perform an efficient global search in an ample solution space. The basic mechanisms of this method consist of mutation, crossover, and selection. Firstly, the mutation process, as given in Equation 1,

is performed.

$$\vartheta_i^{k+1} = \chi_{r1}^k + F \cdot (\chi_{r2}^k - \chi_{r3}^k) \quad (1)$$

Where, ϑ_i^{k+1} is the mutated solution for the i-*th* individual, χ_{r1}^k, χ_{r2}^k , χ_{r3}^k are randomly selected positions of three distinct individuals from the current population, and *F* is the differential weight factor within the range [0, 2]. After mutation, new individuals are created by combining the mutated solution with the existing population using crossover. The crossover operation is defined in Equation 2.

$$u_i^{k+1} = \begin{cases} \vartheta_i^{k+1}, & if\ rand_j \le CR\ or\ j = j_{rand} \\ \chi_i^k, & otherwise \end{cases} \quad (2)$$

Where, u_i^{k+1} is the offspring of the i-*th* individual after crossover, $rand_j$ is a random value in [0, 1], CR is the crossover probability in the range [0, 1], and j_{rand} is a randomly chosen index. In the selection step, the fitness of each new individual is evaluated, and the best individuals are selected for the next generation. The best solution is determined by equation 3.

$$g^{k+1} = arg\ \min_i f\left(u_i^{k+1}\right) \quad (3)$$

Where g^{k+1} is the global best solution at iteration *k+1*, and *f* is the objective function. The best solution from the DE phase is transferred to the FA as part of the initial population shown in Equation 4.

$$P_{Firefly}^0 = \chi_{DE}^* \quad (4)$$

The FA is based on the movement of fireflies influenced by their brightness. Brightness is inversely proportional to the objective function value; brighter individuals represent better solutions. The position of each Firefly is updated according to equation 5.

$$\chi_i^{k+1} = \chi_i^k + \beta e^{-\gamma r_{ij}} \cdot \left(\chi_j^k - \chi_i^k\right) + \alpha \cdot \varepsilon \quad (5)$$

Where, χ_i^{k+1} is the updated position of the *i-th* individual, β is the attraction coefficient, γ is the light absorption coefficient, r_{ij} is the distance between fireflies *i* and j, α is the randomization parameter, and ε is random noise. The objective function is minimized at each iteration, and the brightest Firefly is determined. The best solution is then passed to the MA, as shown in equation 6.

$$P_{Mayfly}^0 = \chi_{Firefly}^* \quad (6)$$

The MA combines social and individual dynamics to find the optimal solution. The velocity and position of each individual are updated as equations 7 and 8.

$$\vartheta_i^{k+1} = \omega \cdot \vartheta_i^k + c_1 r_1 \cdot \left(p_i^k - \chi_i^k\right) + c_2 r_2 \cdot \left(g_i^k - \chi_i^k\right) \quad (7)$$

$$\chi_i^{k+1} = \chi_i^k + \vartheta_i^{k+1} \quad (8)$$

Here, ϑ_i^{k+1} is the velocity of the *i-th* individual at iteration *k*+1, χ_i^{k+1} Position of the i-*th* individual at iteration *k+1*, ω Inertia weight, c_1 and c_2 acceleration coefficients, r_1 and r_2 are random values in [0, 1], p_i^k is the personal best position of the i-*th* individual, g_i^k is the global best position. The MA also includes a mating mechanism to explore new solutions, as shown in equation 9.

$$\chi_{child} = \alpha\chi_{male} + (1-\alpha)\chi_{female}, \alpha\epsilon[0,1] \quad (9)$$

After mating, the offspring may undergo mutation to introduce randomness and improve exploration, as shown in equation 10.

$$\chi_{mutated} = \chi_{child} + N(0, \sigma^2) \quad (10)$$

The best individual is the final solution when the MA reaches the predefined number of iterations, as in equation 11.

$$\chi_{Final}^* = \chi_{Mayfly}^* \quad (11)$$

Throughout the hybrid algorithm, the objective functions of 3 different test functions are used to evaluate solutions and guide the optimization process. The final solution χ_{Final}^*, represents the best result found by the DE-Fly technique.

```
PSEUDO CODE
ALGORITHM DE_FLY
INPUT: pop_size; max_iter_de; max_iter_fa; max_iter_ma;
    bounds; objective f(·);
    DE: F, CR;   FA: beta0, gamma, alpha;   MA: omega, c1, c2, mating_alpha, sigma
BEGIN DE_FLY
  #Phase I: Differential Evolution (global exploration)
  P = Initialize_Population(pop_size, bounds); Evaluate_Fitness(P, f)
  FOR k = 1 .. max_iter_de:
    FOR each individual i in P:
      v = Mutation_DE(P, i, F)              # Eq.(1)
      u = Crossover_DE(P[i], v, CR)         # Eq.(2)
      P[i] = Selection_DE(P[i], u, f)       # Eq.(3)
    END FOR
  END FOR
  best_de = Best(P)                         # seed for FA  (Eq.(4))
  #Phase II: Firefly Algorithm (exploitation + diversity)
  Fflies = Initialize_With_Seed(best_de, pop_size, bounds)
  Evaluate_Fitness(Fflies, f)
  FOR k = 1 .. max_iter_fa:
    FOR each pair (i,j):
      IF brightness(Fflies[i]) < brightness(Fflies[j]):
        Fflies[i] = Firefly_Update(Fflies[i], Fflies[j], beta0, gamma, alpha, bounds)  #  Eq.(5)
      END IF
    END FOR
    Update_Fitness(Fflies, f)
  END FOR
  best_fa = Best(Fflies)                    # seed for MA  ( Eq.(6))
  #Phase III: Mayfly Algorithm (fine-tuning)
  M = Initialize_With_Seed(best_fa, pop_size, bounds); Init_Velocities(M); Init_Personals(M)
  gbest = Best(M)
  FOR k = 1 .. max_iter_ma:
    FOR each mayfly i in M:
      v_i = omega*v_i + c1*rand()*(pbest_i - x_i) + c2*rand()*(gbest - x_i)          #  Eq.(7)
      x_i = Project_Bounds(x_i + v_i, bounds)                                       #  Eq.(8)
    END FOR
    # mating + mutation (optional)
    FOR each (male,female) in Pairs(M):
      child = mating_alpha*male + (1-mating_alpha)*female                           #  Eq.(9)
      child = Project_Bounds(child + Normal(0, sigma^2), bounds)                    #  Eq.(10)
      Insert_If_Better(M, child, f)
    END FOR
    Update_Personal_And_Global_Bests(M, f)
  END FOR
  RETURN gbest
      # χ_Final* (Eqs.11,14)
END DE_FLY
```

In Figure 1, the working principle of the proposed DE-Fly is given as follows.

1. All parameters for the DE algorithm are initialized. The initial population is randomly generated.

2. Differential mutation is performed: The difference vector between three randomly selected individuals is generated and added to the positions of the existing individuals.

3. The crossover process is applied: Each individual is crossed with the mutated solution to create new individuals.

4. Selection process is performed: The new and existing populations are compared according to the cost function, and the most suitable individuals are selected.

5. The DE algorithm terminates when it reaches the specified number of iterations or error criteria. The best solution is passed to the FA:

$$P_{Firefly}^{0} = \chi_{DE}^{*} \tag{12}$$

6. Initialize all parameters for the FA. The best solution from the DE is included in the initial population.

Figure 1. Flowchart of the DE-Fly Optimizer.

7. In each iteration of the FA, the brightness is calculated according to the cost functions of the individuals. Individuals gravitate towards brighter individuals. The randomization term ensures diversity.

8. The FA terminates when it reaches the specified number of iterations or error criteria. The brightest individual (best solution) is transferred to the MA:

$$P^0_{Mayfly} = \chi^*_{Firefly} \tag{13}$$

9. All parameters for the MA are initialized. The best solution from Firefly is used as the initial population.

10. In each iteration of the MA, the speed and position of the individuals are updated. The mating process creates new individuals. The mutation process ensures that new individuals are provided with diversity in the solution space.

11. The MA terminates when it reaches the specified number of iterations or error criteria. The best individual is considered the final solution:

$$\chi^*_{Final} = \chi^*_{Mayfly} \tag{14}$$

12. The final best solution is reported as the result of the hybrid algorithm.

3. Benchmark Test with CEC2019

The CEC2019 benchmark set is widely used for evaluating the performance of optimization algorithms, as shown in Table 1 (38,39). It was designed to test the effectiveness of evolutionary algorithms and meta-heuristic methods in solving real-world problems. Various challenging test functions are included, designed to assess the ability of algorithms to handle complex problems. These functions are characterized by being multi-modal, non-separable, and high-dimensional. Algorithms are evaluated on how well they navigate complex search spaces, and the suite has become a standard testbed frequently employed to demonstrate the effectiveness of newly proposed optimization methods (40–42).

All experiments were conducted on a workstation equipped with an AMD Ryzen 9 7950X 16-core/32-thread processor (5.7 GHz), 64 GB DDR5 7200 MHz RAM, and an NVIDIA RTX 3070 Ti GPU (8 GB). The algorithms were implemented in Python within the Visual Studio environment. Core libraries used include NumPy, Pandas, Scikit-learn, Matplotlib, and Seaborn for data preprocessing and visualization; XGBoost for regression modeling; and SciPy, along with custom Python implementations for metaheuristic optimizations. This computational setup ensured both efficiency in large-scale simulations and reproducibility of results.

Table 1. CEC2019 Test Functions.

No.	Function	D	Search Range	Best
$\mathcal{F}_1$	Storn's Chebyshev Polynomial Fitting Problem	9	[–8192, 8192]	1
$\mathcal{F}_2$	Inverse Hilbert Matrix Problem	16	[−16,384, 16,384]	1
$\mathcal{F}_3$	Lennard–Jones Minimum Energy Cluster	18	[−4, 4]	1
$\mathcal{F}_4$	Rastrigin's Function	10	[−100, 100]	1
$\mathcal{F}_5$	Griewangk's Function	10	[−100, 100]	1
$\mathcal{F}_6$	Weierstrass Function	10	[−100, 100]	1
$\mathcal{F}_7$	Modified Schwefel's Function	10	[−100, 100]	1
$\mathcal{F}_8$	Expanded Schaffer's F6 Function	10	[−100, 100]	1
$\mathcal{F}_9$	Happy Cat Function	10	[−100, 100]	1
$\mathcal{F}_{10}$	Ackley Function	10	[−100, 100]	1

The established benchmark has been further categorized into distinct groups: single-modal, multi-modal, composite, and a category devoted to real-world problems. The categorization in question facilitates the testing of algorithms concerning a wide range of theoretical and practical problems. In terms of performance evaluation, criteria such as solution quality, consistency, and computation are considered. CEC2019 provides standardized test functions, enabling fair comparisons between algorithms.

Table 2 reports the parameter settings of the proposed DE-Fly and the compared algorithms. The term popsize represents the number of individuals in the population for all algorithms,

while max_iter denotes the maximum number of iterations. In DE, F refers to the differential weight factor and C_R to the crossover probability. In the FA, r is a random number drawn from [–1,1], whereas , ℓ_{min} and , ℓ_{max} indicate the lower and upper bounds of the light absorption coefficient. Within the MA, c_1 and c_2 are acceleration coefficients that control the influence of personal and global best positions, and ω denotes the inertia weight balancing exploration and exploitation. In the WOA, a is a convergence parameter linearly decreasing from 2 to 0, b is a constant defining the bubble-net spiral motion, l is a random parameter in [–1,1] controlling exploitation behaviour, and p is the probability of the shrinking encircling mechanism. In the GWO, a is again a linearly decreasing parameter, while A and C are control coefficients derived from random vectors. For the MGO component of MGONM, α represents leader influence, β herd influence, δ escape strength, w an inertia weight decreasing from 0.9 to 0.4, r the adaptive search radius (1→0), and $jump\_prob$ the probability of a random jump. Finally, in the NM phase of MGONM, α denotes the reflection coefficient, γ the expansion coefficient, ρ the contraction coefficient, and σ the shrink coefficient.

Table 2. Optimization Algorithm Parameters.

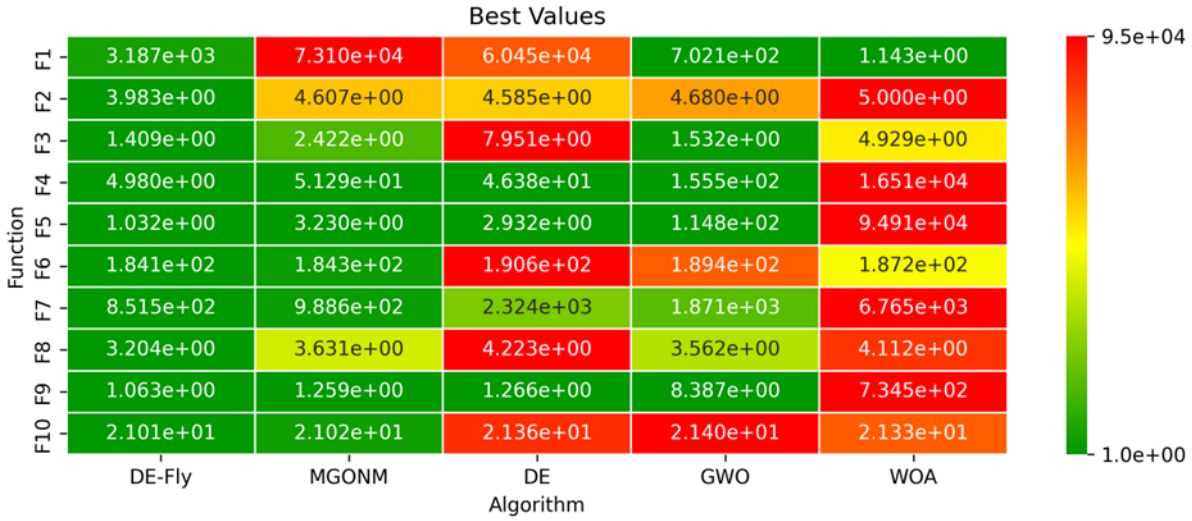
Algorithm		Parameter	Value
DE-Fly	DE	popsize, max_iter, F, C_R	20,100,0.5,0.9
	FA	r, ℓ_{min}, ℓ_{max}	$[-1,1]$, 0.0001,1
	MA	c_1, c_2, ω,	1.5, 1.5, 0.5
WOA		popsize, $\max_{\text{iter}}$, a, b, l, p	20,100, [2,0], 1, $[-1,1]$, 0.5
DE		popsize, max_iter, F, C_R	20,100,0.5,0.9
GWO		popsize, max_iter, a, A, C	$20,100, 2 \rightarrow 0, [-a, a], [0,2]$
MGONM	MGO	popsize, $\max_{\text{iter}}$, α, β, δ, w, r, jump_prob	20,100,2,1.5,0.8,0.9, 0.9 → 0.4, 1 → 0, 0.1
	NM	$\max_{\text{iter}}$, α, γ, ρ, σ	100, 1.0, 2.0, 0.5, 0.5

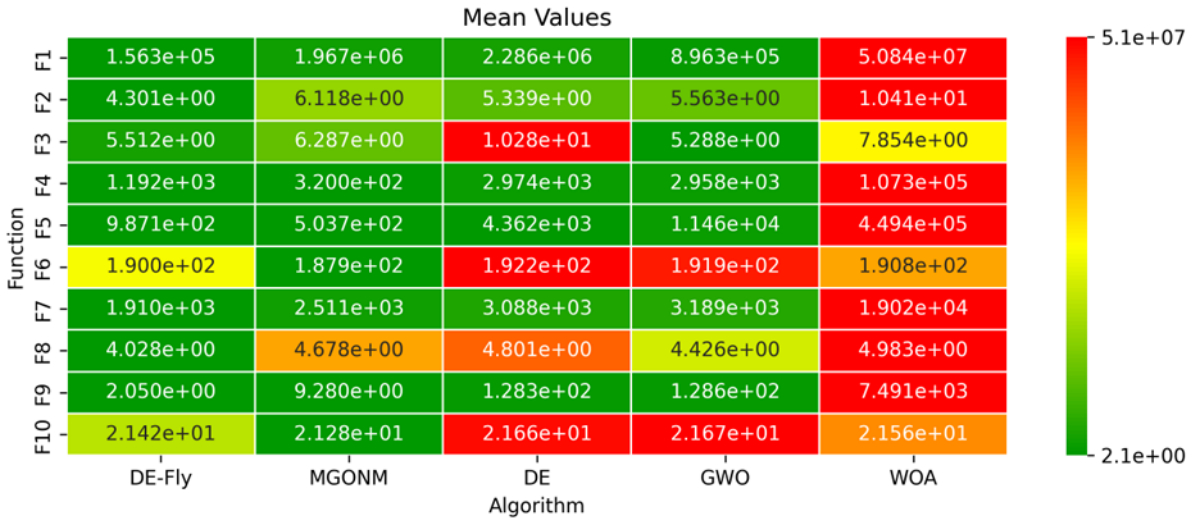
Both Table 3 and Figure 2 present the outcomes of the CEC2019 benchmark suite. Overall, the proposed DE-Fly algorithm consistently provided superior or competitive results across the majority of test functions, outperforming classical DE, swarm-based methods (GWO and WOA), and the hybrid MGONM approach. For the Ackley function, DE-Fly obtained the lowest best value (21.01) and the lowest mean (21.42), with a minimal standard deviation (0.1), indicating highly reliable convergence. In the Storn's Chebyshev Polynomial Fitting Problem, DE-Fly produced the best value (3187), outperforming MGONM and DE by a considerable margin.

Table 3. CEC2019 Performance Comparison.

Function	Metrics	DE-Fly	MGONM	DE	GWO	WOA
$\mathcal{F}_1$	Best	3.187e+03	7.310e+04	6.045e+04	7.021e+02	1.143e+00
	Mean	1.563e+05	1.967e+06	2.286e+06	8.963e+05	5.084e+07
	Std	1.860e+05	2.035e+06	3.374e+06	1.586e+06	6.744e+07
$\mathcal{F}_2$	Best	3.983e+00	4.607e+00	4.585e+00	4.680e+00	5.000e+00
	Mean	4.301e+00	6.118e+00	5.339e+00	5.563e+00	1.041e+01
	Std	1.772e-01	9.003e-01	5.088e-01	5.417e-01	4.060e+00
$\mathcal{F}_3$	Best	1.409e+00	2.422e+00	7.951e+00	1.532e+00	4.929e+00
	Mean	5.512e+00	6.287e+00	1.028e+01	5.288e+00	7.854e+00
	Std	2.453e+00	1.625e+00	7.435e-01	2.163e+00	1.568e+00
$\mathcal{F}_4$	Best	4.980e+00	5.129e+01	4.638e+01	1.555e+02	1.651e+04
	Mean	1.192e+03	3.200e+02	2.974e+03	2.958e+03	1.073e+05
	Std	6.673e+03	4.020e+02	8.064e+03	2.842e+03	6.040e+04
Function	Metrics	DE-Fly	MGONM	DE	GWO	WOA

$\mathcal{F}_5$	Best	1.032e+00	3.230e+00	2.932e+00	1.148e+02	9.491e+04
	Mean	9.871e+02	5.037e+02	4.362e+03	1.146e+04	4.494e+05
	Std	5.042e+03	8.114e+02	1.805e+04	1.654e+04	2.788e+05
$\mathcal{F}_6$	Best	1.841e+02	1.843e+02	1.906e+02	1.894e+02	1.872e+02
	Mean	1.900e+02	1.879e+02	1.922e+02	1.919e+02	1.908e+02
	Std	1.700e+00	1.886e+00	6.988e-01	1.020e+00	1.306e+00
$\mathcal{F}_7$	Best	8.515e+02	9.886e+02	2.324e+03	1.871e+03	6.765e+03
	Mean	1.910e+03	2.511e+03	3.088e+03	3.189e+03	1.902e+04
	Std	4.216e+02	6.677e+02	8.929e+02	6.849e+02	7.384e+03
$\mathcal{F}_8$	Best	3.204e+00	3.631e+00	4.223e+00	3.562e+00	4.112e+00
	Mean	4.028e+00	4.678e+00	4.801e+00	4.426e+00	4.983e+00
	Std	3.594e-01	3.981e-01	1.730e-01	3.856e-01	2.619e-01
$\mathcal{F}_9$	Best	1.063e+00	1.259e+00	1.266e+00	8.387e+00	7.345e+02
	Mean	2.050e+00	9.280e+00	1.283e+02	1.286e+02	7.491e+03
	Std	2.526e+00	7.817e+00	3.432e+02	2.317e+02	5.656e+03
$\mathcal{F}_{10}$	Best	2.101e+01	2.102e+01	2.136e+01	2.140e+01	2.133e+01
	Mean	2.142e+01	2.128e+01	2.166e+01	2.167e+01	2.156e+01
	Std	1.000e-01	1.695e-01	1.154e-01	1.139e-01	1.214e-01





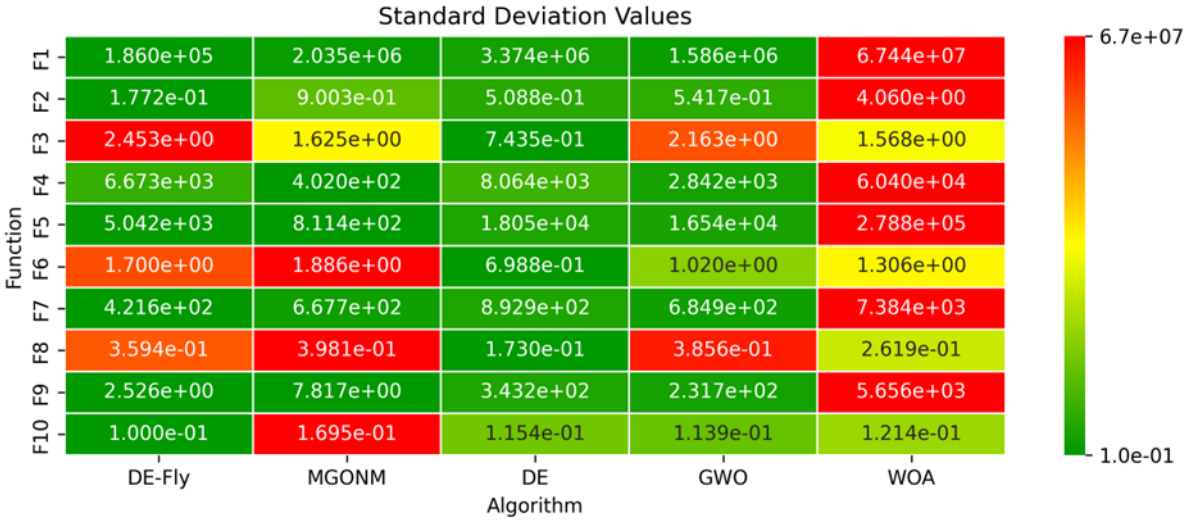


Figure 2. Heatmap of the Performance Comparison of Optimization Algorithms.

Although WOA yielded a slightly lower best result, its mean and variance were significantly higher, demonstrating unstable behavior compared to DE-Fly. On the Expanded Schaffer's F6 Function function, DE-Fly again reported the best performance (best = 3.204, mean = 4.028), with lower variance than MGONM, confirming consistent convergence. On more complex multimodal landscapes, DE-Fly maintained its superiority. For the Griewank's function, DE-Fly achieved the lowest best (1.032) and mean (987.1) values, whereas other methods showed larger deviations. The Happy Cat function further confirmed this advantage, with DE-Fly yielding both the lowest best (1.063) and mean (2.050), far outperforming MGONM (9.280) and DE (128.3). In the Inverse Hilbert Matrix Problem, DE-Fly achieved the best value (3.983) and the lowest mean (4.301), with a standard deviation (0.177) considerably smaller than those of competing methods, indicating strong robustness. For the Lennard–Jones Minimum Energy Cluster, DE-Fly again produced the best value (1.409) and lowest mean (5.512), surpassing MGONM (6.287) and DE (10.28). In the Rastrigin's function, known for its highly multimodal surface, DE-Fly was the only optimizer that maintained competitive best (4.980) and mean (1192) values with reasonable stability, whereas MGONM, DE, and WOA suffered from high error variance. On the Modified Schwefel's Function, DE-Fly achieved the lowest best (851.5) and mean (1910), while alternative methods diverged significantly. Finally, for the Weierstrass function, DE-Fly closely approximated the global optimum with a best of 184.1 and a mean of 190.0, again recording the lowest variance among all tested algorithms.

In summary, the DE-Fly algorithm consistently delivered superior results in terms of best solution quality, mean stability, and variance reduction across the CEC2019 test functions. These findings validate the effectiveness of the proposed multi-phase framework and demonstrate its advantage over both single-phase optimizers and hybrid counterparts such as MGONM.

4. GDTP Mechanism for XGBoost

XGBoost builds its predictive model by adding trees in sequence, with each new tree aiming to reduce the residual errors of the ensemble. While this iterative approach improves the capacity to represent nonlinear patterns, it may also cause overfitting when tree depth and complexity increase excessively. Conventional pruning strategies usually depend on fixed rules, such as a maximum depth limit or a constant gain threshold, and adaptive approaches often expand only those nodes that exceed a specified gain level (43,44). The GDTP mechanism differs by applying a gradient-based threshold that is updated at every boosting iteration. Through this adjustment, node expansion and pruning are regulated together according to the distribution of gradient information, ensuring that weak splits are removed and only effective expansions are retained. In this way, tree growth adapts to the learning dynamics, controlling model complexity while preserving predictive accuracy.

4.1. Background on XGBoost Formulation

XGBoost is a widely adopted tree-based ensemble learning algorithm that iteratively minimizes prediction error through additive model construction (45,46). It has been shown to

effectively handle nonlinear relationships, noisy or missing data, and high-dimensional feature spaces, making it one of the most robust shallow learning methods for forecasting tasks (47,48).

Given a dataset with n observations $\{(x_i, y_i)\}_{i=1}^{n}$, the predicted output $\hat{y}_i$ is expressed as the sum of outputs from K DTs:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathscr{F} \tag{15}$$

where $\mathcal{F}$ denotes the functional space of regression trees. The learning objective at iteration t is defined as:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) \tag{16}$$

Here, l represents the loss function, and $\Omega(f_t)$ is a regularization term penalizing model complexity, defined as:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{J=1}^{T} \omega_j^2 \tag{17}$$

where T is the number of leaves in the tree, ω_j is the score on leaf j, γ, and λ are regularization parameters controlling the trade-off between model fit and complexity.

Using a second-order Taylor expansion, the objective can be approximated as:

$$\mathcal{L}^{(t)} \approx \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \Omega(f_t) \tag{18}$$

where g_i and h_i are the first and second derivatives of the loss function with respect to the previous prediction $\hat{y}_i^{(t-1)}$.

4.2. Classical Split Gain Calculation

During tree construction, XGBoost evaluates potential split points based on the split gain, defined as:

$$\Upsilon = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right] - \gamma \tag{19}$$

where G_L and H_L are the sums of the first and second derivatives for the left child node, and G_R and H_R are the same for the right child node. A positive gain indicates that the split improves the objective and should be applied.

4.3. Proposed GDTP Strategy

In conventional XGBoost, the growth of trees is often limited by a fixed maximum depth. However, this static constraint may not be optimal for datasets with varying complexity across different regions of the feature space. To address this, we propose the GDTP mechanism, which introduces an adaptive, split-level pruning condition based on the split gain.

The proposed pruning rule is defined as $If\ \Upsilon < \gamma_{dyn}, then\ prune\ the\ branch$, where γ_{dyn} is a dynamic gradient improvement threshold that governs whether a split should proceed or be terminated. To enable data-driven adaptability, γ_{dyn} is formulated as:

$$\gamma_{dyn} = \alpha \cdot \bar{\Upsilon} \tag{20}$$

where α is a tunable scaling factor, γ is the average gain observed in the previously constructed trees or branches. This formulation allows each branch to make pruning decisions based on the overall quality of splits observed throughout the model, ensuring that only meaningful branches are expanded.

4.4. Integration with the DE-Fly Optimizer

The proposed GDTP mechanism introduces an additional control layer into the XGBoost learning process by adaptively regulating the tree expansion based on the gradient-based dynamic threshold. γ_{dyn}.

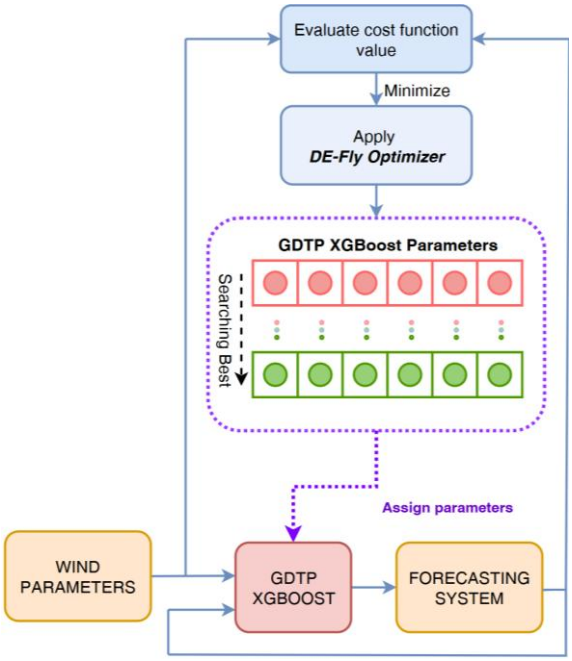


Figure 3. Block diagram of the DE-Fly Optimized GDTP-XGBoost framework for Wind Energy Forecasting.

Finding an appropriate balance between the pruning threshold and other hyperparameters presents a complex, multi-dimensional optimization problem that traditional methods often struggle to address efficiently (49). To overcome this challenge, the present study adopts the DE-Fly Optimizer, thereby enabling more effective tuning of the GDTP-XGBoost model. The overall optimization objective is formulated as follows:

$$minimize_{\Theta,a}\ Validation\ Loss(\Theta, \gamma_{dyn}) = \alpha \cdot \Upsilon \tag{21}$$

where Θ a represents the standard XGBoost hyperparameters

and α is the scaling factor controlling the dynamic pruning threshold. A balanced progression across both the hyperparameter space and the model's structural parameters is enabled by the structure of the proposed optimization approach.

As shown in Figure 3, the DE-Fly algorithm makes the GDTP-XGBoost plan better by doing cost function checks over and over again, which creates a way to change the model so that it can do what is needed for forecasting. This flexibility enables the model to adapt to the particular attributes of the prediction task. So, the DE-Fly-optimised GDTP-XGBoost framework is pretty solid. It has got much strength, can be scaled up or down as needed, and is pretty efficient. It is not just about making better predictions, but also about making sure things are reliable, avoiding failures, and planning maintenance for wind energy systems.

5. A Case Study on DE-Fly Enhanced GDTP-XGBoost

This research examines the role of optimization algorithms in improving hyperparameter tuning within a GDTP-XGBoost forecasting framework, emphasizing applications in wind energy. In addition to achieving higher predictive accuracy, the approach contributes to operational practices by supporting maintenance planning and strengthening system reliability. In practice, accurate turbine output forecasts help engineers better understand system behavior and schedule timely interventions, especially crucial for wind farms relying on CBM and RCM protocols.

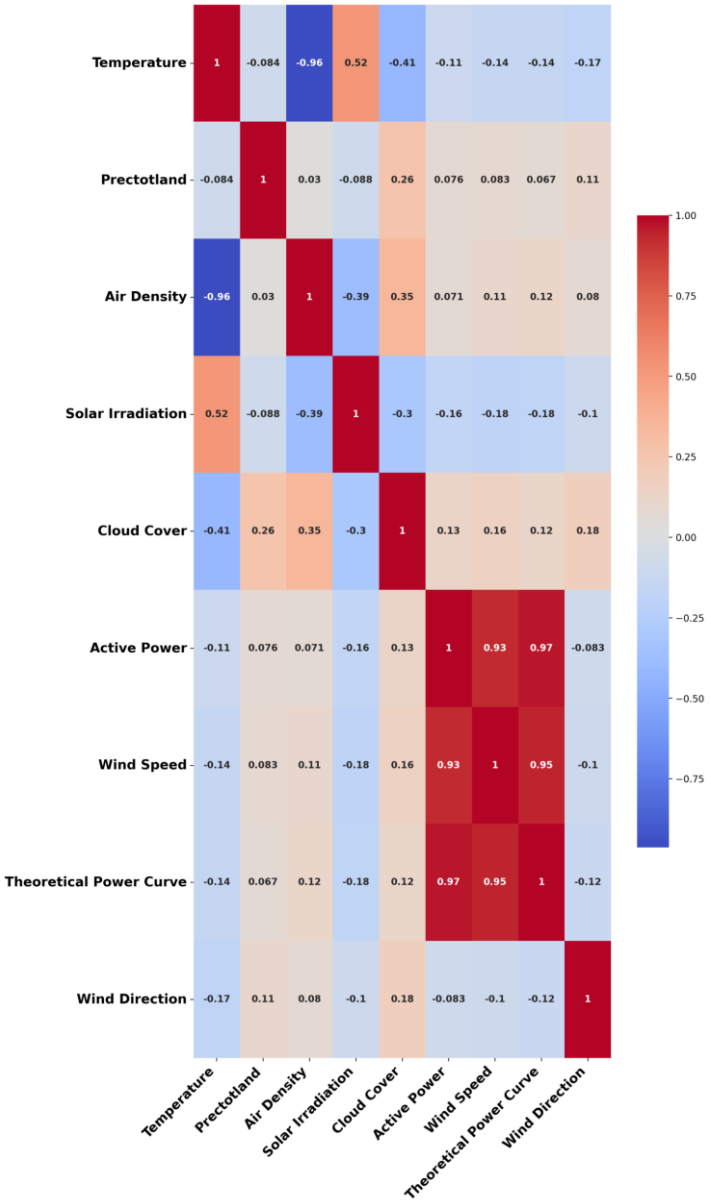


Figure 4. Correlation of turbine and meteorological parameters.

The proposed model was evaluated using operational data from a wind farm in Yalova, Turkey, collected over the course of 2018 (50). SCADA data from Nordex N117/3600 turbines, including wind speed, wind direction, theoretical power, and actual output, were merged with meteorological records to enhance forecast reliability.

The complete set of input features used in the forecasting model is summarized in Table 4, where each parameter is described in terms of its physical meaning, unit, and data source. For instance, prectotland denotes total land precipitation obtained from the MERRA-2 reanalysis dataset (mm/h), while cloud Cover represents the fraction of the sky obscured by clouds (0–1). Altogether, these variables provide both direct and indirect indicators of turbine performance, enabling the model to monitor load trends, identify anomalies, and evaluate component conditions within predictive maintenance systems. The original dataset contained 50,830 entries recorded at 10-minute intervals, which were aggregated into hourly averages, resulting in 8,760 samples. To address missing values, 484 gaps were identified and filled using a combination of moving average and linear interpolation. For model evaluation, the dataset was divided into 70% training and 30% testing subsets, and all reported performance metrics were computed on the test set.

Maintaining data consistency throughout this process was essential to ensure the accuracy of the model, particularly for applications where operational decisions rely on dependable forecasts. In this study, the forecasting task was defined as estimating the potential power output of the turbine given a set of meteorological conditions, rather than extending the prediction horizon. This design enables the model to anticipate turbine load levels under forecasted weather scenarios, supporting its practical use in condition-based maintenance applications. The forecasting model was trained on historical SCADA power data combined with meteorological records to capture the relationship between atmospheric conditions and turbine output, and this learned relationship was then used with forecasted meteorological inputs to estimate the turbine's potential future power generation.

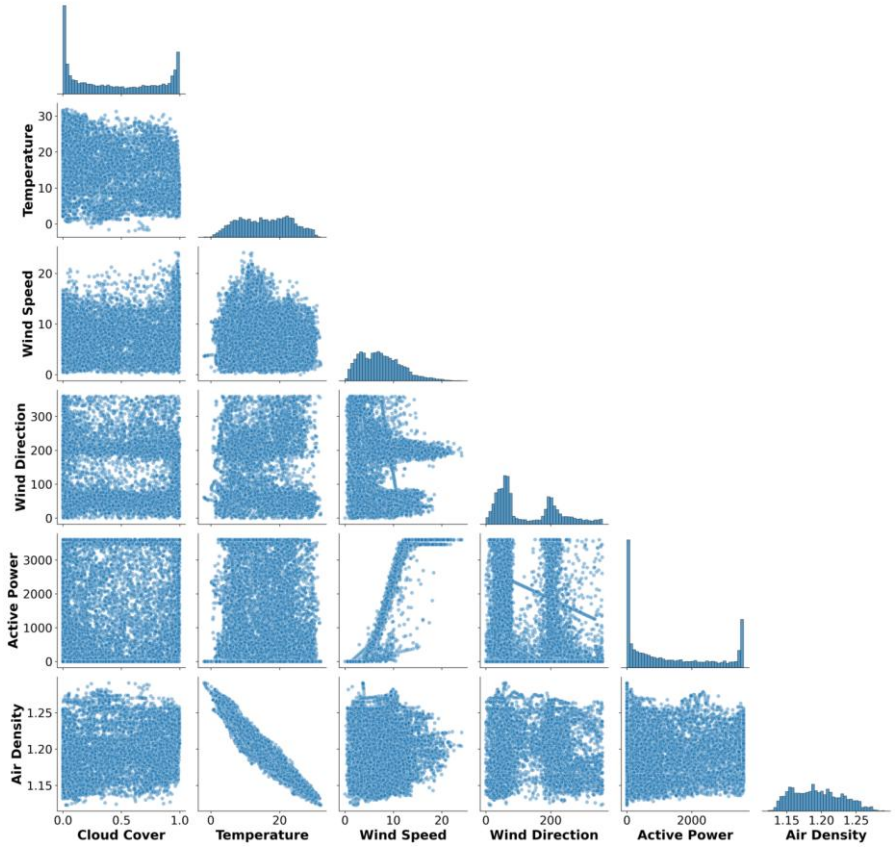


Figure 5. Bivariate Pair Plot of Selected Traits.

Following data preparation, a correlation analysis was conducted to explore the relationships between input features and turbine power output. As shown in Figure 4 and the pairwise scatter plots in Figure 5, wind speed exhibits the strongest correlation with output power (R = 0.93), confirming its dominant role in wind energy forecasting. In contrast, variables such as air density and wind direction showed weaker associations (R = 0.12), indicating secondary influence.

Table 4. Description of the input parameters used in the forecasting model.

Parameter	Description	Unit	Source
Wind Speed	Hub-height wind velocity	m/s	SCADA
Wind Direction	Wind flow orientation at hub height	°	SCADA
Air Density	Calculated from pressure and temperature	kg/m^3	MERRA-2
Temperature	Ambient air temperature	$°C$	MERRA-2
Solar Irradiation	Global tilted irradiation	W/m^2	MERRA-2
Prectotland	Total land precipitation	mm/h	MERRA-2
Cloud Cover	Fraction of sky obscured by clouds (0–1)	–	MERRA-2

However, due to their indirect effects on turbine performance and atmospheric conditions, they were still included in the model. Similarly, temperature showed a weak negative correlation with R = -0.11, cloud cover with R = -0.09, and solar radiation with R = -0.06. This information helped identify the most relevant features to be used in machine learning models, enabling the model to focus on the most effective variables.

The performance of the GDTP-XGBoost model is highly sensitive to its hyperparameter configuration, which affects learning dynamics, model complexity, and overall stability. These parameters also influence the reliability of downstream decisions, particularly in maintenance applications where accurate forecasts are used to inform operational planning. In this study, five metaheuristic algorithms were used to tune the hyperparameters and explore their impact on predictive performance.

Key parameters such as the number of estimators (n_estimators) and tree depth (max_depth) directly influence the model's ability to learn patterns without overfitting. Learning rate (learning_rate), subsample ratio, and colsample_bytree determine how training data and features are utilized, affecting the model's generalization capacity. Gamma controls tree splitting sensitivity, encouraging structural simplicity when set appropriately. The GDTP mechanism adaptively manages these interactions to balance learning precision and model efficiency. Table 5 compares the parameter sets identified by each optimization algorithm, highlighting their different tuning behaviors and guiding the selection of robust configurations.

Table 5. Comparison of the Optimized Parameters of the XGBoost Model.

Method	n_estimators	max_depth	learn_rate	subsample	colsample_bytree	Gamma
WOA	323	6	0.1317	0.7066	0.9160	2.8166
GWO	197	6	0.1598	0.7707	0.8784	3.8409
DE	187	5	0.1061	0.7938	0.9407	2.3516
DE-Fly	366	5	0.187	0.962	0.885	0.156
MGONM	311	5	0.163	0.915	0.891	1.8

Figures 6 and 7 present the performance evaluation of the GDTP-XGBoost Regressor model in forecasting wind power output. In Figure 6, the predicted values closely follow the actual power trends over a selected 100-hour interval, capturing both peak and valley patterns with moderate error. This visual consistency indicates that the model can effectively adapt to the dynamic nature of wind energy production. Figure 7 provides a complementary perspective by plotting all actual versus predicted values across the test dataset. The tight concentration of points around the diagonal reference line suggests that the model exhibits strong predictive accuracy and minimal bias. The combination of these visualizations confirms the Gradient Boosting model's capability to model nonlinear patterns in wind energy data, though its performance is ultimately outperformed by the proposed GDTP-XGBoost approach.

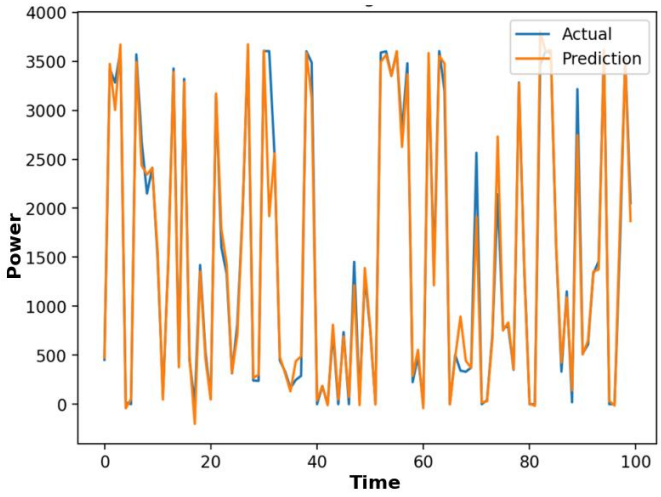


Figure 6. Prediction and actual wind power values for 100 hours using GDTP-XGBoost.

According to Table 6, which is visualized in Figure 8, the results show how the tested optimization algorithms perform based on three key metrics: coefficient of determination (R^2), root mean square error (RMSE), and residual standard error (RSE). Together, these measures compare the model's accuracy and ability to explain variation in the target variable. The coefficient of determination (R^2) reflects how well the model captures the variance in the data, with values closer to 1 indicating stronger predictive performance. In this study, the DE-Fly–optimized GDTP-XGBoost model achieved an R^2 of 0.9892, showing a strong alignment between predicted and actual

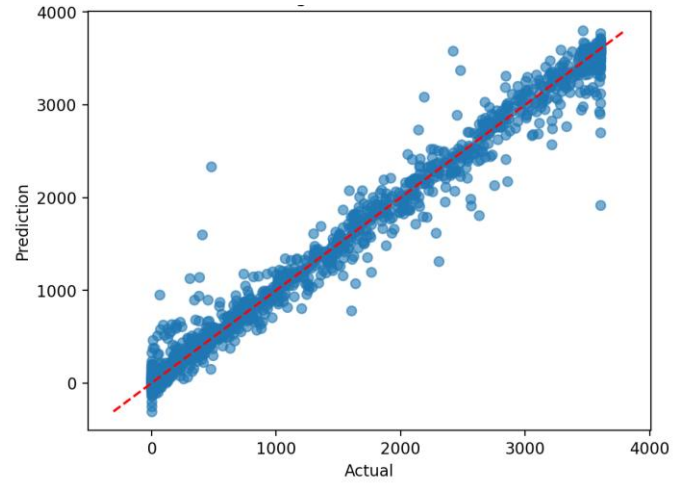


Figure 7. Scatter plot of actual vs. predicted wind power with the 1:1 reference line using GDTP-XGBoost.

values. The RMSE was employed to measure the average deviation between predicted and observed values. A smaller RMSE reflects higher forecasting precision. Among all tested methods, the DE-Fly model achieved the lowest RMSE value of 135.0388, demonstrating its strong ability to capture the dynamics of wind power generation. This outcome indicates that the proposed optimizer not only models underlying data structures with accuracy but also reduces prediction errors more consistently than competing algorithms. For maintenance planning, this improvement is especially valuable, as reliable forecasts support earlier fault identification, load management, and efficient scheduling, thereby increasing turbine availability and reducing the likelihood of unscheduled stoppages. In addition, lower RMSE values make it possible to recognize abnormal deviations between expected and actual output at an early stage, providing practical warning signs of mechanical stress or potential failures. Such information allows maintenance teams to act preventively, limiting unexpected interruptions and extending the operational life of key components.

Table 6. Performance Metric Comparison of the DE-Fly Enhanced GDTP-XGBoost.

Method	R^2	RMSE	RSE
WOA	0.9883	140.6290	0.01167
GWO	0.9879	142.7685	0.01203
DE	0.9872	146.7788	0.01272
MGONM	0.9887	138.5345	0.01116
DE-Fly	0.9892	135.0388	0.01076

The RSE achieved by DE-Fly was 0.01076, the lowest among the compared models. This finding underlines the algorithm's advantage in delivering accurate forecasts and emphasizes its relevance for reliability-oriented applications. Accurate prediction contributes directly to reducing operational risks and strengthening turbine sustainability. The improved performance over classical Differential Evolution and other single-phase optimizers can be explained by the balanced integration of its three stages: global exploration through DE, local refinement with FA, and adaptive adjustment via MA. The combination of these mechanisms avoids premature convergence, accelerates search efficiency, and provides more precise final solutions. Consequently, DE-Fly proves effective not only for tuning hyperparameters but also for managing structural complexity within the GDTP-XGBoost model, leading to improved forecasting results and operational benefits.

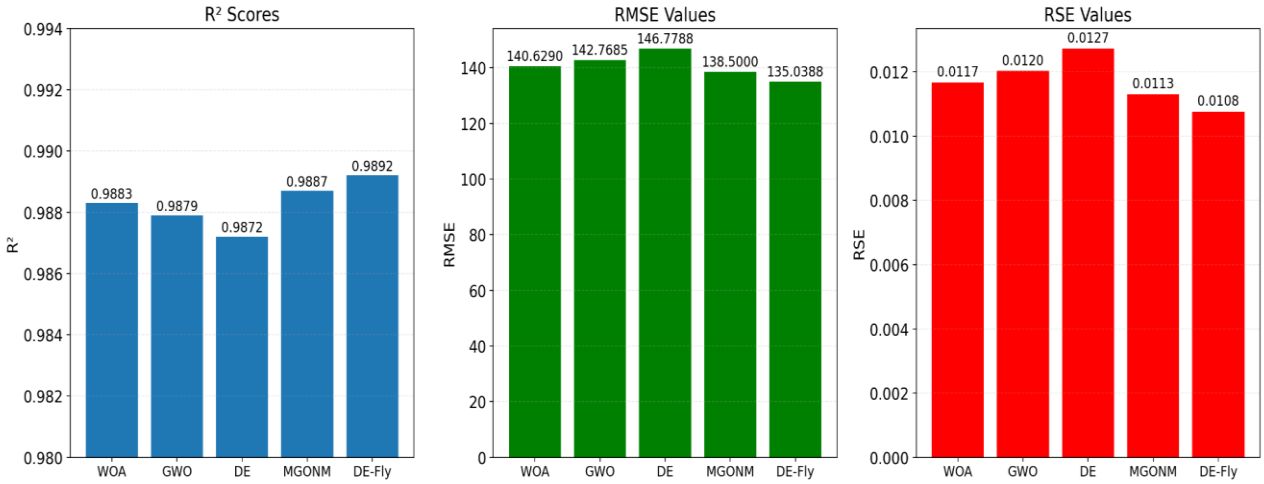


Figure 8. Performance Comparison of Algorithms on GDTP-XGBoost.

The proposed study is also compared with previously published results on the same dataset, as shown in Table 7. In (9), wind power prediction was carried out using ANN, ANFIS, DT, and SVM, achieving an R^2 value of 0.95. Similarly, in (10), deep learning-based methods such as ANN, CNN, LSTM, and RNN were employed, with the best model yielding an R^2 of 0.9574. In addition to these literature benchmarks, several regression and deep learning models were implemented in this study for direct comparison. Among these, the classical Linear Regression (LR) model yielded the lowest performance with an R^2 of 0.8638, followed by ANN (0.9149) and Random Forest (RF) (0.9496). Deep learning models such as Gated Recurrent Unit (GRU) (0.96), LSTM (0.9515), and CNN (0.9809) demonstrated strong performance on the dataset. Notably, the DT model alone achieved an R^2 of 0.9668. However, the highest accuracy was obtained by the proposed GDTP-XGBoost model optimized via the DE-Fly algorithm, achieving an R^2 of 0.9892, outperforming all other tested models as well as the best results reported in (9) and (10). This improvement highlights the strength of the proposed method in modeling complex, nonlinear wind power generation dynamics and providing highly reliable predictions for operational use. Additionally, an ablation study was conducted by comparing the proposed GDTP-XGBoost with the standard XGBoost. The results confirmed the superiority of the proposed approach, achieving an R^2 of 0.9892 with a training time of 0.18 s, compared to 0.9492 and 1.49 s for the baseline XGBoost, respectively.

Table 7. Performance Comparison of Proposed Models with Existing Studies.

Method	**Performance Comparison**	
	R^2	**Train Time (sec.)**
XGBoost	0.9492	1.49
ANN	0.9149	7.13
RF	0.9496	0.49
Proposed GDTP-XGBoost	0.9892	0.18
DT	0.9668	0.05
LR	0.8638	0.01
CNN	0.9809	24
GRU	0.96	60.37
LSTM	0.9515	68.65
(9)	0,95	NA
(10)	0,9574	85.8

In addition to its predictive accuracy, the GDTP-XGBoost framework also demonstrated a remarkable advantage in terms of computational time efficiency. By dynamically pruning unnecessary tree growth during training and employing the DE-Fly optimizer for more targeted hyperparameter tuning, the model converged faster than conventional XGBoost and deep learning counterparts. This balance of accuracy and speed underlines the practical suitability of the proposed approach for real-time forecasting and maintenance decision support, where both reliability and timely computation are essential.

The fact that the proposed GDTP-XGBoost framework outperformed CNN and LSTM is particularly significant. While deep learning models are effective at modeling temporal dependencies, they typically require large datasets, extensive training time, and considerable computational resources. In contrast, tree-based methods, when equipped with adaptive hyperparameters and structural tuning, can rival or surpass deep learning performance at a fraction of the computational cost. This balance of accuracy, efficiency, and interpretability makes the proposed method especially attractive for real-time forecasting and condition-based maintenance applications.

6. Conclusion

This study introduces a forecasting framework which integrates XGBoost with the GDTP mechanism. The aim is to enhance the accuracy of wind power predictions and to strengthen operational reliability through predictive insights. The GDTP-XGBoost configuration is engineered to curtail superfluous model proliferation while retaining its capacity for generalisation, thus enhancing its aptitude to discern variations in wind speed. To further refine performance, the DE-Fly optimiser is used to adjust the hyperparameters automatically. In a study that looked at how well different strategies could predict wind turbine performance, the DE-Fly–optimized GDTP-XGBoost model was better than other strategies at predicting how well turbines would perform. Maintenance planning is supported by stronger predictive quality, which also reduces reliance on corrective repairs and enables more effective data-driven management in wind energy systems. The framework has forecasting capability. This contributes directly to maintenance and reliability practices. Precise predictions show possible mechanical problems early on, which helps with condition-based maintenance by doing things at the right time and in the right way. Long-term application in reliability-focused environments is supported by the integrated optimisation process, which also improves model stability. For this reason, the model is used not only to predict the future, but also to make sure that the company can continue to work effectively.

Planned future work involves a comprehensive evaluation of the GDTP-XGBoost model's adaptability across multiple energy domains, geographical settings, and time series scenarios. The next steps will be to develop the DE-Fly optimiser to handle multi-objective problems and to incorporate the forecasting framework into real-time maintenance platforms. Here, any differences between forecasts and actual data can indicate problems with individual components. The study will also be extended to additional wind farm sites and turbine types to verify the broader applicability of the proposed approach.

References

1. Hassan Q, Viktor P, J. Al-Musawi T, Mahmood Ali B, Algburi S, Alzoubi HM, et al. The renewable energy role in the global energy Transformations. Renewable Energy Focus. 2024 Mar;48:100545.
2. Yüksek G. Forecasting Wind Energy Production: Analysis of Meteorological and Temporal Variables Using Optimized Regression Modeling. In: 2024 Global Energy Conference (GEC). IEEE; 2024. p. 146–52. https://doi.org/10.1109/GEC61857.2024.10880956
3. Salgado Duarte Y, Szpytko J. Reliability-oriented twin model for integrating offshore wind farm maintenance activities. Eksploatacja i Niezawodność – Maintenance and Reliability. 2024 Dec 27; https://doi.org/10.17531/ein/199355
4. Sulaiman MH, Mustaffa Z. Enhancing wind power forecasting accuracy with hybrid deep learning and teaching-learning-based optimization. Cleaner Energy Systems. 2024 Dec;9:100139.
5. AlShafeey M, Csaki C. Adaptive machine learning for forecasting in wind energy: A dynamic, multi-algorithmic approach for short and long-term predictions. Heliyon. 2024 Aug;10(15):e34807.
6. Zhao E, Sun S, Wang S. New developments in wind energy forecasting with artificial intelligence and big data: a scientometric insight. Data Science and Management. 2022 Jun;5(2):84–95. https://doi.org/10.1016/j.dsm.2022.05.002

7. Wang Y, Chen T, Zhou S, Zhang F, Zou R, Hu Q. An improved Wavenet network for multi-step-ahead wind energy forecasting. Energy Convers Manag. 2023 Feb;278:116709.
8. Nascimento EGS, de Melo TAC, Moreira DM. A transformer-based deep neural network with wavelet transform for forecasting wind speed and wind energy. Energy. 2023 Sep;278:127678.
9. Bouabdallaoui D, Haidi T, Elmariami F, Derri M, Mellouli EM. Application of four machine-learning methods to predict short-horizon wind energy. 2023;6:726–37. Available from: www.sciencedirect.com/journal/global-energy-interconnection. https://doi.org/10.1016/j.gloei.2023.11.006
10. Karaman ÖA. Prediction of Wind Power with Machine Learning Models. Applied Sciences. 2023 Oct 19;13(20):11455.
11. Li D, Qi Z, Zhou Y, Elchalakani M. Machine Learning Applications in Building Energy Systems: Review and Prospects. Buildings. 2025 Feb 19;15(4):648.
12. Abdelsattar M, A. Ismeil M, Menoufi K, AbdelMoety A, Emad-Eldeen A. Evaluating Machine Learning and Deep Learning models for predicting Wind Turbine power output from environmental factors. PLoS One. 2025 Jan 23;20(1):e0317619.
13. Trizoglou P, Liu X, Lin Z. Fault detection by an ensemble framework of Extreme Gradient Boosting (XGBoost) in the operation of offshore wind turbines. Renew Energy. 2021 Dec;179:945–62. https://doi.org/10.1016/j.renene.2021.07.085
14. Aslan E. Temperature Prediction and Performance Comparison of Permanent Magnet Synchronous Motors Using Different Machine Learning Techniques for Early Failure Detection. Eksploatacja i Niezawodność – Maintenance and Reliability. 2024 Aug 9;27(1). https://doi.org/10.17531/ein/192164
15. Moorthy PK, Goel N, Baghel S. Regression Methods and Models. In: Deep Learning Concepts in Operations Research. New York: Auerbach Publications; 2024. p. 199–225. https://doi.org/10.1201/9781003433309-17
16. Bai FJJS, Jasmine RA. Optimization of tree-based machine learning algorithms for improving the predictive accuracy of hepatitis C disease. In: Decision-Making Models. Elsevier; 2024. p. 523–45. https://doi.org/10.1016/B978-0-443-16147-6.00015-3
17. Khan MS, Peng T, Akhlaq H, Khan MA. Comparative Analysis of Automated Machine Learning for Hyperparameter Optimization and Explainable Artificial Intelligence Models. IEEE Access. 2025;1–1. https://doi.org/10.1109/ACCESS.2025.3566427
18. Oliva D, Houssein EH, Hinojosa S, editors. Metaheuristics in Machine Learning: Theory and Applications. Vol. 967. Cham: Springer International Publishing; 2021. https://doi.org/10.1007/978-3-030-70542-8
19. Das S, Suganthan PN. Differential Evolution: A Survey of the State-of-the-Art. IEEE Transactions on Evolutionary Computation. 2011 Feb;15(1):4–31. https://doi.org/10.1109/TEVC.2010.2059031
20. Schmidt M, Safarani S, Gastinger J, Jacobs T, Nicolas S, Schulke A. On the Performance of Differential Evolution for Hyperparameter Tuning. In: 2019 International Joint Conference on Neural Networks (IJCNN). IEEE; 2019. p. 1–8. https://doi.org/10.1109/IJCNN.2019.8851978
21. Sen A, Gupta V, Tang C. Differential Evolution Algorithm Based Hyperparameter Selection of Gated Recurrent Unit for Electrical Load Forecasting [Internet]. 2023. Available from: https://arxiv.org/abs/2309.13019
22. Ali Y, Awwad E, Al-Razgan M, Maarouf A. Hyperparameter Search for Machine Learning Algorithms for Optimizing the Computational Complexity. Processes. 2023 Jan 21;11(2):349.
23. Brodzicki A, Piekarski M, Jaworek-Korjakowska J. The Whale Optimization Algorithm Approach for Deep Neural Networks. Sensors. 2021 Nov 30;21(23):8003.
24. Oladejo SO, Ekwe SO, Ajibare AT, Akinyemi LA, Mirjalili S. Tuning SVMs' hyperparameters using the whale optimization algorithm. In: Handbook of Whale Optimization Algorithm. Elsevier; 2024. p. 495–521. https://doi.org/10.1016/B978-0-32-395365-8.00042-7
25. Yu X, Xu W, Wu X, Wang X. Reinforced exploitation and exploration grey wolf optimizer for numerical and real-world optimization problems. Applied Intelligence. 2022 Jun 28;52(8):8412–27. https://doi.org/10.1007/s10489-021-02795-4
26. Sumathi S, Rajesh R. HybGBS: A hybrid neural network and grey wolf optimizer for intrusion detection in a cloud computing environment. Concurr Comput. 2024 Nov 19;36(24). https://doi.org/10.1002/cpe.8264
27. Aufa BZ, Suyanto S, Arifianto A. Hyperparameter Setting of LSTM-based Language Model using Grey Wolf Optimizer. In: 2020 International Conference on Data Science and Its Applications (ICoDSA). IEEE; 2020. p. 1–5. https://doi.org/10.1109/ICoDSA50139.2020.9213031

28. Rahamathulla MY, Ramaiah M. Optimizing anomaly detection models for edge IIoT with an enhanced firefly algorithm-based hyperparameter tuning strategy. Results in Engineering. 2025 Sep;27:105843.
29. Nayak J, Naik B, Dash PB, Souri A, Shanmuganathan V. Hyper-parameter tuned light gradient boosting machine using memetic firefly algorithm for hand gesture recognition. Appl Soft Comput. 2021 Aug;107:107478.
30. Izci D, Hekimoğlu B, Ekinci S. A new artificial ecosystem-based optimization integrated with Nelder-Mead method for PID controller design of buck converter. Alexandria Engineering Journal. 2022 Mar;61(3):2030–44. https://doi.org/10.1016/j.aej.2021.07.037
31. Boddu Y, Manimaran A, Arunkumar B, Ramkumar D. Design of an Iterative Dual Metaheuristic VARMAx Model Enhancing Efficiency of Time Series Predictions. IEEE Access. 2024;12:128071–84. https://doi.org/10.1109/ACCESS.2024.3454540
32. Ekinci S, Izci D, Yilmaz M. Efficient Speed Control for DC Motors Using Novel Gazelle Simplex Optimizer. IEEE Access. 2023;11:105830–42. https://doi.org/10.1109/ACCESS.2023.3319596
33. Snášel V, Rizk-Allah RM, Izci D, Ekinci S. Weighted mean of vectors optimization algorithm and its application in designing the power system stabilizer. Appl Soft Comput. 2023 Mar;136:110085.
34. Eker E, Kayri M, Ekinci S, Izci D. A New Fusion of ASO with SA Algorithm and Its Applications to MLP Training and DC Motor Speed Control. Arab J Sci Eng. 2021 Apr 2;46(4):3889–911. https://doi.org/10.1007/s13369-020-05228-5
35. Lale T. GWO-WOA-AOA: Multi-stage Hybrid Metaheuristic Optimization Approach. In: 2025 9th International Symposium on Innovative Approaches in Smart Technologies (ISAS). IEEE; 2025. p. 1–8. https://doi.org/10.1109/ISAS66241.2025.11101906
36. Yang XS. Firefly Algorithm, Stochastic Test Functions and Design Optimisation. 2010 Mar 6; Available from: http://arxiv.org/abs/1003.1409
37. Zervoudakis K, Tsafarakis S. A mayfly optimization algorithm. Comput Ind Eng. 2020 Jul;145:106559.
38. Brest J, Maucec MS, Boskovic B. The 100-Digit Challenge: Algorithm jDE100. In: 2019 IEEE Congress on Evolutionary Computation (CEC). IEEE; 2019. p. 19–26. https://doi.org/10.1109/CEC.2019.8789904
39. Viktorin A, Senkerik R, Pluhacek M, Kadavy T, Zamuda A. DISH Algorithm Solving the CEC 2019 100-Digit Challenge. In: 2019 IEEE Congress on Evolutionary Computation (CEC). IEEE; 2019. p. 1–6. https://doi.org/10.1109/CEC.2019.8789936
40. Braik M, Al-Hiary H. Rüppell's fox optimizer: A novel meta-heuristic approach for solving global optimization problems. Cluster Comput. 2025 Aug 28;28(5):292. https://doi.org/10.1007/s10586-024-04950-1
41. Sharma P, Raju S. Techno-economic analysis of Retired Electric Vehicle Batteries with Grid-Connected Hybrid Energy System. Energy Convers Manag. 2025 Sep;339:119870.
42. Eker E. Development of Random Walks Strategy-Based Dandelion Optimizer and Its Application to Engineering Design Problems. IEEE Access. 2025;13:56547–75. https://doi.org/10.1109/ACCESS.2025.3554505
43. Chen T, Guestrin C. XGBoost: A Scalable Tree Boosting System. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, NY, USA: ACM; 2016. p. 785–94. https://doi.org/10.1145/2939672.2939785
44. Ostroumova L, Gusev G, Vorobev A, Dorogush AV, Gulin A. CatBoost: unbiased boosting with categorical features. In: Neural Information Processing Systems [Internet]. 2017. Available from: https://api.semanticscholar.org/CorpusID:5044218
45. Hakkal S, Lahcen AA. XGBoost To Enhance Learner Performance Prediction. Computers and Education: Artificial Intelligence. 2024 Dec;7:100254.
46. Niazkar M, Menapace A, Brentan B, Piraei R, Jimenez D, Dhawan P, et al. Applications of XGBoost in water resources engineering: A systematic literature review (Dec 2018–May 2023). Environmental Modelling & Software. 2024 Mar;174:105971.
47. Xu Y, Zheng S, Zhu Q, Wong K chun, Wang X, Lin Q. A complementary fused method using GRU and XGBoost models for long-term solar energy hourly forecasting. Expert Syst Appl. 2024 Nov;254:124286.
48. Zhang L, Jánošík D. Enhanced short-term load forecasting with hybrid machine learning models: CatBoost and XGBoost approaches. Expert Syst Appl. 2024 May;241:122686.
49. Fan M, Xiao K, Sun L, Zhang S, Xu Y. Automated Hyperparameter Optimization of Gradient Boosting Decision Tree Approach for Gold Mineral Prospectivity Mapping in the Xiong'ershan Area. Minerals. 2022 Dec 16;12(12):1621.
50. Erisen B. https://www.kaggle.com/datasets/berkerisen/wind-turbine-scada-dataset/data. Wind Turbine Scada Dataset.