Research on the Influence of Genetic Algorithm Parameters on XGBoost in Load Forecasting

Thanh-Ngoc Tran
Industrial University of Ho Chi Minh City, Vietnam
tranthanhngoc@iuh.edu.vn (corresponding author)

Quoc-Dai Nguyen
Industrial University of Ho Chi Minh City, Vietnam
22000701.dai@student.iuh.edu.vn

Received: 30 August 2024 | Revised: 12 October 2024 | Accepted: 16 October 2024

Licensed under a CC-BY 4.0 license | Copyright (c) by the authors | DOI: https://doi.org/10.48084/etasr.8863

ABSTRACT

Electric load forecasting is crucial in a power system comprising electricity generation, transmission, distribution, and retail. Due to its high accuracy, the ensemble learning method XGBoost has been widely applied in load forecasting. XGBoost's performance depends on its hyperparameters and the Genetic Algorithm (GA) is a commonly used algorithm in determining the optimal hyperparameters for this model. In this study, we propose a flowchart algorithm to investigate the impact of GA parameters on the accuracy of XGBoost models over the hyperparameter grid for load forecasting. The maximum load data of Queensland, Australia, are used for the research. The analysis of the results indicates that the accuracy of the XGBoost model significantly depends on the values of its hyperparameters. Using default hyperparameter values may lead to substantial errors in load forecasts, while selecting appropriate values for the GA to determine the optimal hyperparameters for the XGBoost model can significantly improve its accuracy.

Keywords-load forecasting; XGBoost; genetic algorithm

I. INTRODUCTION

Electricity load forecasting is the process of estimating the future electricity consumption an plays a significant role in an electricity system composed of production planning, operational planning, and future development planning [1-2]. Load forecasting models based on time-series data represent the changing patterns of electric load over time in an autoregressive manner. Specifically, at each time step, the load forecasting model aims to predict the current load value based on the values from the preceding time points. Numerous methods for electricity load forecasting have been proposed, including classical ones such as regression [3], exponential smoothing [4], and ARIMA [5], as well as more modern techniques like Artificial Neural Networks (ANN) [5-6], deep learning [7], and ensemble learning [8]. In recent years, among advanced load forecasting methods, the XGBoost model has been proven effective and widely used in general time series forecasting in general and load forecasting problems in particular [9-10].

The performance of ensemble learning models, including XGBoost, often depends on their hyperparameters. Thus, several optimization algorithms have been applied to determine them, such as Random Search (RS), Grid Search (GS), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Bayesian Optimization, etc. [11-13]. In recent years, many studies have successfully applied the GA to enhance the performance of the XGBoost model. It is worth noting that the accuracy of the GA algorithms is sensitive to its parameters, making it crucial to evaluate the impact of parameter values on the performance of GA when applied to an XGBoost model. Most authors either use the default parameter values or set them based on empirical values, and, to the best of our knowledge, no studies dedicated in the analysis of this aspect have been published [14-18]. Thus, in this paper, an algorithm flowchart if proposed for the evaluation of the influence of GA's parameters on the XGBoost model regarding the load forecasting problem. The study utilizes peak load data from Queensland, Australia. Boxplot charts will be used statistically to analyze the results of different experimental scenarios.

II. RESEARCH METHODS

A. *XGBoost Model*

XGBoost is a powerful boosting algorithm used for regression and classification tasks [18-19]. Given the dataset $D=\{(x_i, y_i)\}$ where $x_i \in R^m$, (with m being the dimension of each sample) and $y_i \in R^n$) (where n is the number of samples). The prediction model can be described as a tree ensemble model, which includes K decision trees as follows:

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

where f_k represents the k-th decision tree in the ensemble. Each f_k is a function that maps a sample x_i to a predicted value, and these functions are parameterized by the structure and parameters of the decision trees.

To optimize the model, XGBoost minimizes a regularized objective function, which consists of a loss function L that measures the difference between the predicted values and the actual target values, and a regularization term Ω that penalizes the complexity of the model.

$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \quad (2)$$

The regularization term helps to control overfitting by penalizing the complexity of the individual trees f_k. The specific forms of the loss function L and the regularization term Ω can vary depending on the task (e.g. squared error for regression, logistic loss for classification) and the implementation details of XGBoost. The regularization term Ω is defined as follows:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda\|\omega\|^2 \quad (3)$$

where γ is a regularization parameter that controls the complexity of the tree by penalizing the number of leaves T and λ is a regularization parameter that controls the leaf weights w_j. In XGBoost, the prediction of the i-th instance at the t-th iteration, denoted as $\hat{y}_i^{(t)}$ can be obtained by (1). Substituting (1) into the objective function, we can rewrite it as follows:

$$L^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \quad (4)$$

To simplify the optimization process, we can use a second-order Taylor expansion to approximate the objective function:

$$L^{(t)} \simeq \sum_{i=1}^{n}[l(y_i, \hat{y}_i^{(t-1)} + g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)] + \Omega(f_t)$$
$$\forall\, g_i = \frac{\partial l(y_i, \hat{y}^{(t-1)})}{\partial \hat{y}^{(t-1)}}, h_i = \frac{\partial^2 l(y_i, \hat{y}^{(t-1)})}{\partial (\hat{y}^{(t-1)})^2} \quad (5)$$

where g_i and h_i are first and second order gradient statistics of the loss function. The residual between the prediction score $\hat{y}_i^{(t-1)}$ and y_i does not affect the optimization of the objective function, so the specific objective function at step t becomes:

$$\tilde{L}^{(t)} = \sum_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)\right] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2 \quad (6)$$

The objective function is an improvement of the XGBoost method over GBDT (Gradient Boosting Decision Tree) due to the addition of a regularization factor, which helps reduce model complexity and avoid overfitting. In XGBoost, both the first and second derivatives of the loss function are used to optimize the adjustment of the residual error. This method also supports column sampling, which reduces overfitting and speeds up computation. Due to these improvements, XGBoost has many hyperparameters, with the important ones being [20-21]:

- Learning_Rate (eta): Controls the step size at each iteration while moving toward minimizing the loss function.
- Max_Depth: Specifies the maximum depth of the trees.
- Min_Child_Weight: Determines the minimum sum of instance weight (Hessian) needed in a child.
- Subsample: Denotes the fraction of samples to be used for each tree.
- Colsample_bytree: Indicates the feature fraction to be randomly sampled for each tree.
- Gamma: Minimum loss reduction required to partition the leaf node of the tree further.
- Lambda (L2 Regularization): Adds an L2 regularization term to the loss function to avoid overfitting.
- Alpha (L1 Regularization): Adds an L1 regularization term to the loss function to avoid overfitting.

Determining the optimal hyperparameters for the XGBoost model is crucial for enhancing the model's performance. In this study, the authors use the GA to determine the optimal hyperparameters for the XGBoost model, which will be introduced below.

B. Genetic Algorithm

The GA is an evolutionary algorithm method inspired by Charles Darwin's Theory of Evolution through Natural Selection. One of GA's main advantages is its capability to efficiently explore large and complex search spaces, making it suitable for problems with numerous variables or constraints. Moreover, GA can integrate prior knowledge or constraints into the fitness function, generating solutions aligned with domain-specific knowledge. The basic steps of GA (Figure 1) are [14-18, 22]:

- Initial Population: A set of individuals is initialized randomly.
- Fitness Calculation: The fitness of each individual in the population is calculated using a portion of the training data according to the subsample ratio.
- Parent Selection: The best individuals are selected as parents for the crossover process.
- Crossover: Genetic information from two parents are combined to create new individuals.
- Mutation: Random changes are applied to some genes of the new individuals.
- New Generation: A new generation from the crossover and mutated individuals is created.
- Termination Criteria Reached: Check whether the stopping criteria have been met. If yes, the algorithm stops; if not, it goes back to the fitness calculation step.

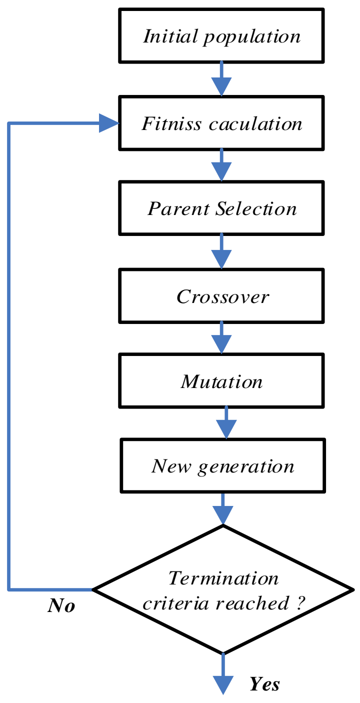

Fig. 1. Implementation steps of the GA algorithm.

A characteristic of the GA is that the results depend on its parameters. Based on the GA process cycle presented in Figure 1, the main parameters of the GA are:

- Generations: Number of iterations to run the pipeline optimization process.
- Population Size: Number of individuals to retain in the genetic programming population for each generation.
- Offspring Size: Number of siblings produced in each generation.
- Mutation Rate: It lies in the range [0.0, 1.0].
- Crossover Rate: It lies in the range [0.0, 1.0].
- Subsample: Fraction of training samples used during the optimization process.

C. The Proposed Algorithm

As mentioned in the previous section, the GA is used to determine the optimal hyperparameters of the XGBoost model, but the GA's performance may depend on its own parameters. Therefore, evaluating the impact of these parameters is crucial when applying GA to the XGBoost model. In this study, the authors propose a flowchart algorithm to investigate the GA's response as the parameters change within a predefined range, as illustrated in Figure 2. The main steps of the model are:

- Create Training Data: Generate the training data, including input variables X and target variables y.
- Define Hyperparameter Ranges for XGBoost: Specify the ranges of the hyperparameters of XGBoost that need to be optimized, including the number of trees (nei), learning rate (lr), tree depth (mdi), etc. Initialize the XGBoost model with hyperparameters within the defined ranges.
- Define Parameter Ranges for GA: Specify the ranges for the parameters of the GA, including the number of generations (gen), mutation rate (mut), subsample (sub), and other parameters.
- Apply Genetic Algorithm: Utilize the GA to search for the optimal hyperparameters of the XGBoost model.
- Obtain Final Results: The final result is an optimal set of hyperparameters (mdl_{opt}) for the XGBoost model along with the proposed corresponding error value, MAPE.

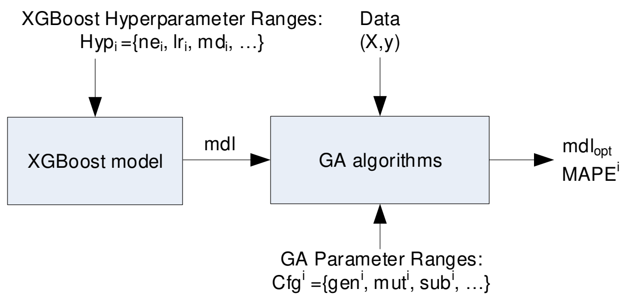
XGBoost Hyperparameter Ranges:
Hyp$_i$ ={ne$_i$, lr$_i$, md$_i$, ...}
Data
(X,y)
XGBoost model
mdl
GA algorithms
mdl$_{opt}$
MAPE^i
GA Parameter Ranges:
Cfg^i ={gen^i, mut^i, sub^i, ...}

Fig. 2. Implementation steps of the proposed algorithm.

Each value of the GA parameters set, denoted as Cfg^i, corresponds to a $MAPE^i$ value. Analyzing the MAPE values in relation to individual parameters such as gen, mut, and sub will assess the impact of each parameter on the accuracy of the GA algorithm. Additionally, this study evaluates the overall impact of these parameters on the performance of the GA algorithm when applied to the XGBoost model for load forecasting problems by analyzing the MAPE accuracy as detailed in Table I.

TABLE I. PROPOSED SURVEY MODELS

Model	Description
XGBoost	Use the default hyperparameters of XGBoost.
GA-XGBoost	Use the default parameters of GA to determine the optimal hyperparameters of XGBoost.
GS-GA-XGBoost	Change the values of GA's parameters based on their Grid Space (GS) values. Each combination of parameter sets will yield a corresponding optimal set of hyperparameters for XGBoost and MAPE.

III. RESULTS AND DISCUSSION

A. Setting Model Parameters

In this study, we used the maximum electricity load dataset from Queensland, Australia, from July 1, 2019, to September 30, 2019 [23]. The peak load values of this dataset are presented in Table II. This dataset will be split into X and y using a sliding window method with a window size of 7 [24]. Table III presents the hyperparameters investigated for the XGBoost model. It includes the default values and the range of values set when applying the GA to determine the optimal hyperparameters of the XGBoost model.

TABLE II. PEAK LOAD VALUES FOR QUEENSLAND DATASET

DATE	Peak Load (MW)
2019-07-01	7353.24
2019-07-02	7209.22
2019-07-03	7191.09
…	
2019-09-28	6693.78
2019-09-29	6848.42
2019-09-30	7291.57

TABLE III. HYPERPARAMETERS FOR THE XGBOOST MODEL

Hyperparameter	Type	Range	Default
n_estimators (ne)	Interger	(1, 500)	100
learning_rate (lr)	Floating-point	(0, 0.5)	0.3
max_depth (md)	Interger	(1, 15)	6
subsample (ss)	Floating-point	(0.5, 1)	1
colsample_bytree (cb)	Floating-point	(0.5, 1)	1
reg_alpha (ra)	Floating-point	(0, 0.3)	0
reg_lambda (rl)	Floating-point	(0, 1)	1

We used the TPOTRegressor syntax for the GA. TPOTRegressor is a part of the TPOT (Tree-based Pipeline Optimization Tool) library. It automates the machine learning pipeline design process using genetic programming to optimize the pipelines [25]. The grid value space for the parameters of the GA is presented in Table IV.

TABLE IV. RANGE OF VALUES FOR THE INVESTIGATED PARAMETERS OF THE GENETIC ALGORITHM

Hyperparameter	Type	Range	Default
generations	Interger	[60, 80, 100, 120, 140]	100
mutation_rate	floating-point	[0.5, 0.6, 0.7, 0.8, 0.9]	0.9
subsample	floating-point	[0.5, 0.6, 0.7, 0.8, 0.9, 1]	1

B. Evaluation of the Overall Impact of the Genetic Algorithm

Figure 3 presents the error rate results for the surveyed cases set up as shown in Table I.

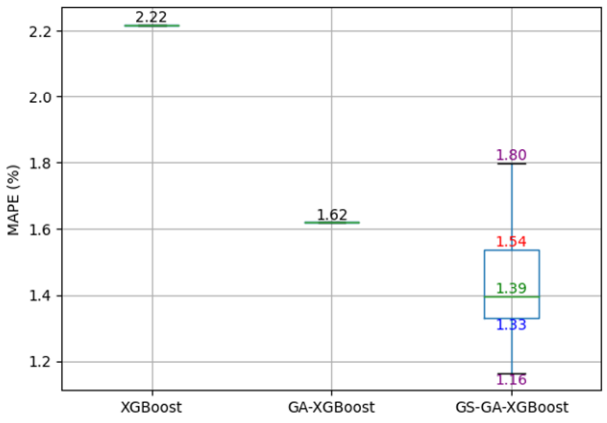

Fig. 3. Response results of the considered XGBoost models.

The results shown in Figure 3 demonstrate that the hyperparameters play an important role in the performance of the XGBoost model. Indeed, when applying the XGBoost model with the default hyperparameters, the error rate is 2.22%. In contrast, using the GA algorithm with its default parameters (GA-XGBoost) to determine the optimal hyperparameters for the XGBoost model, the obtained results have a much lower error rate of 1.62%. Furthermore, the response of the GA algorithm when changing its parameters (GS-GA-XGBoost) yields much better results. Specifically, the values for the GS-GA-XGBoost model are minimum at 1.16%, with the first quartile at 1.33%, the second quartile at 1.39%, and the third quartile at 1.54%, all of which are lower than the value of the GA-XGBoost model of 1.62%.

C. Effect of Individual Parameters of the Genetic Algorithm

Figure 4 presents the boxplot chart of MAPE error rates of the XGBoost models according to the varying values of the generation parameter of the GA algorithm. Table V presents the corresponding statistical results of Figure 4. The analysis results in Figure 4 and Table V show that the model's error decreases gradually when changing the generation values from the default value of 100 (with statistical values Min: 1.1737, Q1: 1.3342, Q2: 1.3939, Q3: 1.5352, Max: 1.7870) to a value of 150 (with statistical values Min: 1.1640, Q1: 1.3260, Q2: 1.3857, Q3: 1.5092, Max: 1.7860).

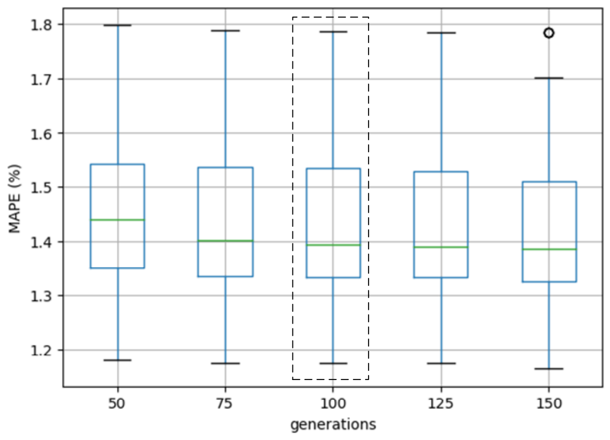

Fig. 4. Error rate according to the generation parameter.

TABLE V. ERROR RATE BY GENERATION PARAMETER VALUES

Value	Min	Q1	Q2	Q3	Max
50	1.1799	1.3515	1.4396	1.5430	1.7986
75	1.1737	1.3357	1.4012	1.5367	1.7881
100	1.1737	1.3342	1.3939	1.5352	1.7870
125	1.1737	1.3334	1.3900	1.5283	1.7860
150	1.1640	1.3260	1.3857	1.5092	1.7860

Figure 5 presents the boxplot of MAPE error rates according to the values of the mutation_rate parameter of the GA algorithm and Table VI provides the corresponding statistical results. The data analyzed in Figure 5 and Table VI indicate that the default value of mutation_rate, which is 0.9 (with statistical values Min: 1.2605, Q1: 1.3299, Q2: 1.4035, Q3: 1.4885), is not the optimal value for model performance. Specifically, a mutation_rate of 0.8 (with statistical values Min: 1.2599, Q1: 1.3212, Q2: 1.3618, Q3: 1.4880) yields better results.

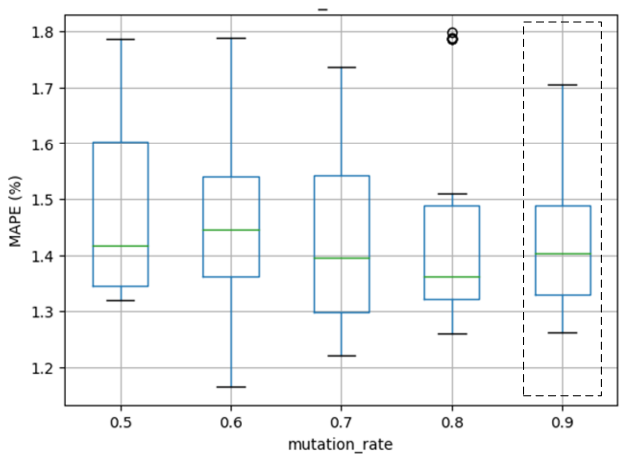

Fig. 5. Error rate according to the mutation_rate parameter.

TABLE VI. ERROR RATE BY MUTATION_RATE PARAMETER VALUES

Values	Min	Q1	Q2	Q3	Max
0.5	1.3187	1.3445	1.4169	1.6018	1.7871
0.6	1.1640	1.3619	1.4465	1.5412	1.7881
0.7	1.2212	1.2974	1.3945	1.5420	1.7368
0.8	1.2599	1.3212	1.3618	1.4880	1.7986
0.9	1.2605	1.3299	1.4035	1.4885	1.7049

Similarly, when considering the values of the subsample parameter of the GA algorithm, the boxplot of error rates and the corresponding statistical data are presented in Figure 6 and Table VII. Apparently, the XGBoost model with the default subsample value of 1.0 is not optimal (with statistical values Min: 1.6096, Q1: 1.7026, Q2: 1.4035, Q3: 1.7860, Max: 1.7986). The error rate decreases progressively from the default subsample value of 1.0 to a subsample value of 0.7 (with statistical values Min: 1.2605, Q1: 1.2965, Q2: 1.3202, Q3: 1.3326, and Max: 1.3665).

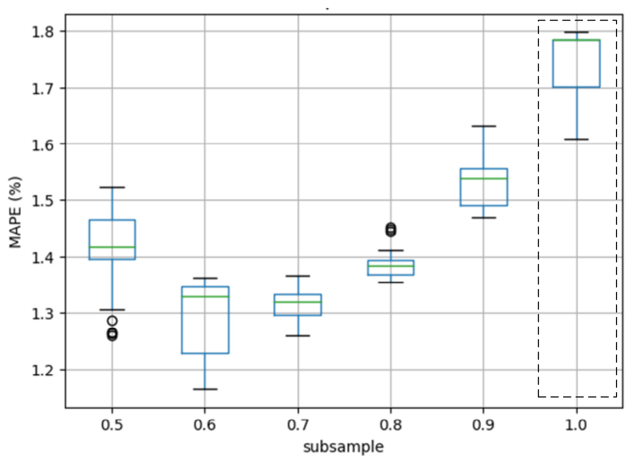

Fig. 6. Error rate according to the subsample parameter.

TABLE VII. ERROR RATE BY SUBSAMPLE PARAMETER VALUES

Values	Min	Q1	Q2	Q3	Max
0.5	1.2599	1.3950	1.4170	1.4658	1.5234
0.6	1.1640	1.2286	1.3293	1.3471	1.3619
0.7	1.2605	1.2965	1.3202	1.3326	1.3665
0.8	1.3548	1.3687	1.3841	1.3942	1.4514
0.9	1.4689	1.4911	1.5382	1.5562	1.6312
1	1.6096	1.7026	1.7860	1.7860	1.7986

The above analysis allows concluding that the values of the generation, mutation_rate, and subsample parameters of the GA algorithm significantly impact the performance of the XGBoost model for load forecasting. The default values of these parameters result in noticeably higher prediction errors than other values. Therefore, choosing GA parameters is crucial when applying the XGBoost model to load forecasting problems.

IV. CONCLUSIONS

In this study, a process is proposed for the investigation of the influence of the Genetic Algorithm (GA) parameters on the XGBoost model applied to the load forecasting problem. The investigated models include the XGBoost model with default hyperparameter values, the XGBoost model with hyperparameters determined by the default GA, and the XGBoost model with hyperparameters determined by the GA, but with varying parameters. The performance of the GA is also examined for each of its parameters. The results indicate that the GA enhances the accuracy of the XGBoost model, and the default values of the GA parameters do not correspond to the optimal values. The findings emphasize the importance of the GA in determining optimal hyperparameters on the XGBoost model to solve the load forecasting issue. In addition, the obtained results also underline the significance of identifying reasonable values for the parameters of these optimization algorithms to further improve the model's performance. Future research will be expanded to investigate the influence of parameters on other optimization algorithms, such as PSO, TPE, or Bayesian optimization. Additionally, it will be extended to other machine learning models, particularly CNN and LSTM, to evaluate the effectiveness of these models in load forecasting problems.

REFERENCES

[1] M. Q. Raza and A. Khosravi, "A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings," *Renewable and Sustainable Energy Reviews*, vol. 50, pp. 1352–1372, Oct. 2015, https://doi.org/10.1016/j.rser.2015.04.065.

[2] J. Walther, D. Spanier, N. Panten, and E. Abele, "Very short-term load forecasting on factory level – A machine learning approach," *Procedia CIRP*, vol. 80, pp. 705–710, Jan. 2019, https://doi.org/10.1016/j.procir.2019.01.060.

[3] J. Kim, S. Cho, K. Ko, and R. R. Rao, "Short-term Electric Load Prediction Using Multiple Linear Regression Method," in *IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids*, Aalborg, Denmark, Oct. 2018, pp. 1–6, https://doi.org/10.1109/SmartGridComm.2018.8587489.

[4] N. A. A. Jalil, M. H. Ahmad, and N. Mohamed, "Electricity Load Demand Forecasting Using Exponential Smoothing Methods," *World Applied Sciences Journal*, vol. 22, no. 11, pp. 1540–1543, 2013, https://doi.org/10.5829/idosi.wasj.2013.22.11.2891.

[5] R. K. Jagait, M. N. Fekri, K. Grolinger, and S. Mir, "Load Forecasting Under Concept Drift: Online Ensemble Learning With Recurrent Neural Network and ARIMA," *IEEE Access*, vol. 9, pp. 98992–99008, Jan. 2021, https://doi.org/10.1109/ACCESS.2021.3095420.

[6] A. I. Arvanitidis, D. Bargiotas, A. Daskalopulu, V. M. Laitsos, and L. H. Tsoukalas, "Enhanced Short-Term Load Forecasting Using Artificial Neural Networks," *Energies*, vol. 14, no. 22, Jan. 2021, Art. no. 7788, https://doi.org/10.3390/en14227788.

[7] N.-H. Duong, M.-T. Nguyen, T.-H. Nguyen, and T.-P. Tran, "Application of Seasonal Trend Decomposition using Loess and Long Short-Term Memory in Peak Load Forecasting Model in Tien Giang," *Engineering, Technology & Applied Science Research*, vol. 13, no. 5, pp. 11628–11634, Oct. 2023, https://doi.org/10.48084/etasr.6181.

[8] M. Bhatnagar, V. Dwivedi, D. Singh, and G. Rozinaj, "Comprehensive Electric load forecasting using ensemble machine learning methods," in *29th International Conference on Systems, Signals and Image Processing*, Sofia, Bulgaria, Jun. 2022, vol. CFP2255E-ART, pp. 1–4, https://doi.org/10.1109/IWSSIP55020.2022.9854390.

[9] R. A. Abbasi, N. Javaid, M. N. J. Ghuman, Z. A. Khan, S. Ur Rehman, and Amanullah, "Short Term Load Forecasting Using XGBoost," in *Web, Artificial Intelligence and Network Applications*, L. Barolli, M. Takizawa, F. Xhafa, and T. Enokido, Eds. New York, NY, USA: Springer, 2019, pp. 1120–1131.

[10] Y. Liu, H. Luo, B. Zhao, X. Zhao, and Z. Han, "Short-Term Power Load Forecasting Based on Clustering and XGBoost Method," in *9th International Conference on Software Engineering and Service Science*, Beijing, China, Nov. 2018, pp. 536–539, https://doi.org/10.1109/ICSESS.2018.8663907.

[11] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020, https://doi.org/10.1016/j.neucom.2020.07.061.

[12] A. Avci, M. Kocakulak, N. Acir, E. Gunes, and S. Turan, "A study on the monitoring of weld quality using XGBoost with Particle Swarm Optimization," *Ain Shams Engineering Journal*, vol. 15, no. 4, Apr. 2024, Art. no. 102651, https://doi.org/10.1016/j.asej.2024.102651.

[13] N. Ghatasheh, I. Altaharwa, and K. Aldebei, "Modified Genetic Algorithm for Feature Selection and Hyper Parameter Optimization: Case of XGBoost in Spam Prediction," *IEEE Access*, vol. 10, pp. 84365–84383, Jan. 2022, https://doi.org/10.1109/ACCESS.2022.3196905.

[14] Y. Jiang, G. Tong, H. Yin, and N. Xiong, "A Pedestrian Detection Method Based on Genetic Algorithm for Optimize XGBoost Training Parameters," *IEEE Access*, vol. 7, pp. 118310–118321, 2019, https://doi.org/10.1109/ACCESS.2019.2936454.

[15] V. H. Masand *et al.*, "GA-XGBoost, an explainable AI technique, for analysis of thrombin inhibitory activity of diverse pool of molecules and supported by X-ray," *Chemometrics and Intelligent Laboratory Systems*, vol. 253, Oct. 2024, Art. no. 105197, https://doi.org/10.1016/j.chemolab.2024.105197.

[16] J. Deng, Y. Fu, Q. Liu, L. Chang, H. Li, and S. Liu, "Automatic Cardiopulmonary Endurance Assessment: A Machine Learning Approach Based on GA-XGBOOST," *Diagnostics*, vol. 12, no. 10, Oct. 2022, Art. no. 2538, https://doi.org/10.3390/diagnostics12102538.

[17] Y. Qu, Z. Lin, H. Li, and X. Zhang, "Feature Recognition of Urban Road Traffic Accidents Based on GA-XGBoost in the Context of Big Data," *IEEE Access*, vol. 7, pp. 170106–170115, Jan. 2019, https://doi.org/10.1109/ACCESS.2019.2952655.

[18] N. Thi Thuy Linh *et al.*, "Flood susceptibility modeling based on new hybrid intelligence model: Optimization of XGboost model using GA metaheuristic algorithm," *Advances in Space Research*, vol. 69, no. 9, pp. 3301–3318, May 2022, https://doi.org/10.1016/j.asr.2022.02.027.

[19] N. T. Tran, T. T. G. Tran, T. A. Nguyen, and M. B. Lam, "A new grid search algorithm based on XGBoost model for load forecasting," *Bulletin of Electrical Engineering and Informatics*, vol. 12, no. 4, pp. 1857–1866, Aug. 2023, https://doi.org/10.11591/eei.v12i4.5016.

[20] L. Jovanovic *et al.*, "Improving Phishing Website Detection using a Hybrid Two-level Framework for Feature Selection and XGBoost Tuning," *Journal of Web Engineering*, vol. 22, no. 3, pp. 543–574, Feb. 2023, https://doi.org/10.13052/jwe1540-9589.2237.

[21] S.-E. Ryu, D.-H. Shin, and K. Chung, "Prediction Model of Dementia Risk Based on XGBoost Using Derived Variable Extraction and Hyper Parameter Optimization," *IEEE Access*, vol. 8, pp. 177708–177720, Jan. 2020, https://doi.org/10.1109/ACCESS.2020.3025553.

[22] M. A. El-Shorbagy and A. M. El-Refaey, "Hybridization of Grasshopper Optimization Algorithm With Genetic Algorithm for Solving System of Non-Linear Equations," *IEEE Access*, vol. 8, pp. 220944–220961, Jan. 2020, https://doi.org/10.1109/ACCESS.2020.3043029.

[23] K. Ni, J. Wang, G. Tang, and D. Wei, "Research and Application of a Novel Hybrid Model Based on a Deep Neural Network for Electricity Load Forecasting: A Case Study in Australia," *Energies*, vol. 12, no. 13, Jan. 2019, Art. no. 2467, https://doi.org/10.3390/en12132467.

[24] J.-S. Chou, D.-N. Truong, and T.-L. Le, "Interval Forecasting of Financial Time Series by Accelerated Particle Swarm-Optimized Multi-Output Machine Learning System," *IEEE Access*, vol. 8, pp. 14798–14808, Jan. 2020, https://doi.org/10.1109/ACCESS.2020.2965598.

[25] R. S. Olson and J. H. Moore, "TPOT: A Tree-based Pipeline Optimization Tool for Automating Machine Learning," in *Automatic Machine Learning: Methods, Systems, Challenges*, Springer, 2019, pp. 151–160.