Arabian Journal for Science and Engineering
https://doi.org/10.1007/s13369-024-09460-1

RESEARCH ARTICLE-ELECTRICAL ENGINEERING

Research on the Impact of the Differencing Operator on Ensemble Learning Algorithms in the Case of Peak Load Forecasting

Thanh Ngoc Tran[1]

Received: 15 January 2024 / Accepted: 30 July 2024

Abstract

Peak load forecasting is a critical aspect of power system operations and planning. Accurate forecasting of peak loads significantly impacts the overall efficiency and reliability of a power system. Among the numerous load forecasting methods that are used, ensemble learning algorithms have emerged as a popular choice due to their high accuracy. In this research, the author proposes an innovative methodology that integrates the Differencing Operator with the Sliding Window procedure for training and predicting peak loads using commonly employed ensemble learning models such as GBDT, XGBoost, LightGBM, and CatBoost. The performance of the proposed approach was evaluated by analyzing the prediction error and execution time. The results obtained demonstrated improved accuracy in peak load forecasting, with no impact on execution time.

Keywords Load forecasting · Ensemble learning · Window procedure · Differencing Operator

1 Introduction

Electrical peak load forecasting plays a vital role in ensuring the reliability, safety, and economic efficiency of power system operations and planning. It provides valuable reference values and guidance for integrating renewable energy sources, such as wind and solar power, into the smart grid [1–4]. The research literature has proposed numerous models for electrical peak load forecasting, which can be categorized into two major groups: (1) classic stage, and (2) advanced stage. The classic stage category includes well-known forecasting methods such as Regression [5, 6], Stochastic time series [7, 8], and Exponential Smoothing [9, 10]. In the advanced stage group, researchers have reported the effectiveness of Fuzzy logic [11, 12], Artificial neural network [13–15], Support Vector Machines [16, 17], Hybrid Techniques [18–20], and Ensemble Learning [21, 22]. In this context, ensemble learning is a machine learning technique that combines predictions from two or more models to increase the accuracy and reliability of the final results. By leveraging the collective predictions of multiple models, ensemble learning can achieve higher accuracy and more reliable predictions compared to individual models. Recently, there has been extensive research on the application of ensemble learning using decision tree-based machine learning algorithms in load forecasting, resulting in remarkable outcomes. Notably, this paper will consider models such as GBDT, XGBoost, LightGBM, and CatBoost, which have demonstrated promising performance in this field [23–30].

Since peak load is a time series, it is common to utilize the Sliding Window procedure when applying ensemble algorithms. This procedure helps partition the data into input and target sets, enabling the training and forecasting processes for the load profile to be performed using ensemble algorithm models [28, 31, 32]. Another aspect examined in this study is the periodicity of peak load. For example, the load characteristics of a specific Monday may exhibit similarities to those of the previous Monday. When using only the Sliding Window procedure in data processing, the cyclic nature of the load data may inadvertently be overlooked. Therefore, in this study, the author recommends a novel approach that involves incorporating the input data Differencing Operator to account for the cyclical characteristics of the load data. More specifically, the analysis will focus on the series $Z_t = Y_t – Y_{t\text{-}d}$ as an alternative to using the original data Y_t, where d represents the differencing order. The Differencing

✉ Thanh Ngoc Tran
tranthanhngoc@iuh.edu.vn

[1] Faculty of Electrical Engineering Technology, Industrial University of HCM City (IUH), Ho Chi Minh City, Vietnam

Published online: 26 August 2024

Operator, integrated with the Sliding Window procedure, will be employed in combination with ensemble learning algorithms. The proposed method's effectiveness will be assessed through the evaluation of forecast errors and program execution time. The GBDT, XGBoost, LightGBM, and CatBoost algorithms will be sequentially investigated. Each algorithm will consider a large number of hyperparameter combinations. Furthermore, this study will utilize peak load data from two Australian states, New South Wales and Queensland, enhancing the reliability of the research results.

This paper is organized as follows: In Sect. 2, a brief introduction to ensemble algorithms is presented; Sect. 3 proposes a new approach through the combination of the Sliding Window procedure with the Differencing Operator; Sect. 4 conducts empirical assessments on real datasets from two states of Australia; and finally, Sect. 5 presents the conclusions.

2 Review of Ensemble Algorithm

Ensemble learning is a technique that enhances predictive performance by combining multiple models, surpassing the performance of individual models used in isolation. There are three main classes of ensemble learning: bagging, stacking, and boosting [33]. Bagging is an ensemble learning algorithm that creates a diverse group of ensemble members by training models on different subsets of the training dataset. On the other hand, stacking involves training different types of models on the training data to generate predictions, which are then combined using another model. Boosting is an ensemble algorithm that leverages the mistakes made by previous predictors to improve future predictions. Boosting algorithms have gained significant attention in recent years and will also be employed in this paper. Notably, boosting algorithms come in various forms, including GBDT (Gradient Boosting Decision Trees), XGBoost (Extreme Gradient Boosting), LightGBM (Light Gradient Boosting Machine), and CatBoost (Categorical Boosting) [34–36].

2.1 Ensemble Algorithms

GBDT algorithm was first introduced by Friedman in 2001, presenting a novel approach that combines Gradient Boosting and Decision Trees in machine learning [37, 38]. In Gradient Boosting, multiple weak learners are connected sequentially, with each learner aiming to minimize the error of the previous learner. Gradient Boosting utilizes gradient descent to construct new weak learners along the direction of the current model's loss function. The Decision Tree plays a crucial role as the main component of GBDT and serves as a weak learner within the Gradient Boosting process. The integration of Gradient Boosting and Decision Trees in GBDT leads to enhanced effectiveness in learning and optimization.

XGBoost is a scalable, end-to-end tree boosting method developed by Chen and Guestrin in 2016 [39]. It is an improved algorithm based on the GBDT model, which uses second-order Taylor expansion on the loss function and incorporates regular terms into the objective function to achieve the optimal solution. This approach helps control the decline of the objective function and the complexity of a model, resulting in better convergence, prevention of overfitting, and ultimately providing higher forecasting accuracy. Additionally, XGBoost processes the data and stores the results before training, enabling their reuse in subsequent iterations to reduce computational complexity and facilitate parallel execution, thereby increasing efficiency.

LightGBM is a novel gradient boosting framework developed by Microsoft Research Asia in 2017 [40]. It is an enhanced version of GBDT that incorporates two key techniques: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). The core concept behind GOSS is that larger gradients contribute more to the information gain. The GOSS algorithm identifies samples with high gradients and randomly selects a subset from samples with small gradients. This approach effectively utilizes the samples during the training process, optimizing their impact on the model. On the other hand, EFB focuses on reducing the number of features by merging mutually exclusive ones. EFB consists of two algorithms: one for exclusive feature bundling, which combines related features, and another for merging feature bundles and assigning a value to the resulting bundle.

CatBoost is a new gradient descent algorithm that was presented by Prokhorenkova et al. in 2018 [41]. It is highly effective in predicting categorical features and is based on the utilization of binary decision trees as base predictors. This algorithm incorporates several techniques including permutation methods, one-hot-max-size encoding, greedy methods for new tree splits, and target-based statistics. These techniques are applied as follows: The dataset is randomly permuted into subsets, the labels are converted to integer numbers, and the categorical values are transformed into numerical representations. This combination of techniques enhances the effectiveness of CatBoost in handling categorical data and improves its predictive capabilities.

2.2 Hyperparameters

One concern when using hybrid learning techniques is the hyperparameters of the model, which affect the performance and accuracy of an ensemble model. Hyperparameters are parameters that are set prior to training a machine learning model, unlike model parameters which are learned from data during training. There are many hyperparameters for ensemble algorithms, which can be classified into many groups such as Accuracy, Speed, and Overfitting. In this research, each model of GBDT, XGBoost, LightGBM, and CatBoost will be performed and evaluated in combination with different values of typical hyperparameters to increase the reliability of the results [42, 43]. These key hyperparameters that will be considered in this paper include:

- The learning rate (lr), which determines the step size at each iteration with respect to the loss gradient function.
- The maximum depth (md), which is an integer that controls the maximum distance between the root node and a leaf node.
- The number of estimators (ne), which is the number of trees used in the model.

3 Proposed Method

3.1 Sliding Window Procedure

Time series data refers to a collection of observations on the values that a variable takes at different points in time, following a uniform time–frequency. It can be represented by the equation:

$$y_1,\ y_2,\ \ldots,\ y_m,\ \ldots,\ y_M \tag{1}$$

where m ranges from 1 to M, representing the number of observation values.

To incorporate time series data into an ensemble algorithm, the Sliding Window procedure has been utilized to extract both time series data and production data features. The Sliding Window procedure is illustrated in Fig. 1, where a window size of 7 has been employed [32].

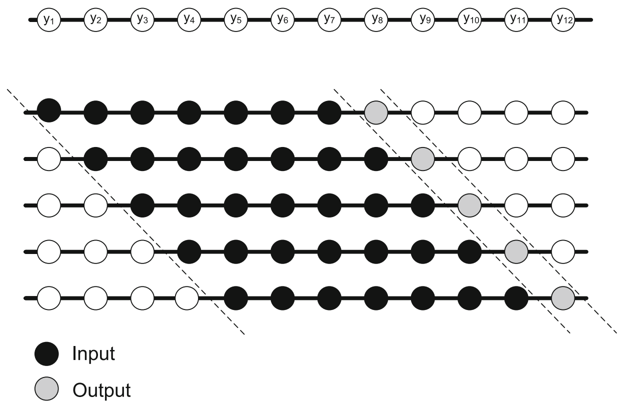


Fig. 1 The process of the Sliding Window procedure

Table 1 The Sliding Window procedure process

Dataset	Input				Output
Training	y_1	y_2	…	y_N	y_{N+1}
	…	…	…	…	…
	y_{M-H-N}	$y_{M-H-N+1}$	…	y_{M-H-1}	y_{M-H}
Testing	$y_{M-H-N+1}$	$y_{M-H-N+2}$	…	y_{M-H}	y_{M-H+1}
	…	…	…	…	…
	y_{M-N}	y_{M-N+1}	…	y_{M-1}	y_M

When working with a time series dataset of length M, the Sliding Window procedure is applied using a window size denoted as N. Subsequently, the dataset is divided into training and testing subsets, where the number of testing instances is denoted as H. The steps for constructing the dataset using the sliding window procedure are detailed in Table 1.

The training dataset is composed of an input sequence $X_{\text{train}} = \{y_1, \ldots, y_N;\ y_2, \ldots, y_{N+1};\ \ldots;\ y_{M-H-N}, \ldots, y_{M-H-1}\}$ and an output sequence $Y_{\text{train}} = \{y_{N+1}, \ldots y_{M-H}\}$. On the other hand, the testing dataset includes an input sequence $X_{\text{test}} = \{y_{M-H+1-N}, \ldots, y_{M-H};\ y_{M-H+2-N}, \ldots, y_{M-H+1};\ \ldots;\ Y_{M-N}, \ldots, y_{M-1}\}$ and an output sequence $Y_{\text{test}} = \{y_{M-H+1}, \ldots y_M\}$. Following the Sliding Window procedure mentioned earlier, the data is structured into input and output components. The training data is represented as $(X_{\text{train}}, Y_{\text{train}})$, while the testing data is represented as $(X_{test}, Y_{\text{test}})$. These training and testing datasets serve as the foundation for applying machine learning algorithm in time series forecasting.

3.2 Differencing Operator

The repetitive nature of time series refers to the regular or periodic patterns that occur in the data over time. These patterns can occur at regular intervals, such as hourly, daily, weekly, monthly, or yearly for electric load patterns. To address the repetitive nature and make time series data more amenable to analysis, the Differencing Operator is often applied. The Differencing Operator involves computing the difference between consecutive observations in the time series by subtracting the previous value from the current value. This captures the changes or fluctuations in the data, as shown by the equation below:

$$y(t) = y(t) - y(t-d) \tag{2}$$

where d is the order of differencing.

The algorithm flowchart of the Differencing Operator is shown in Fig. 2. By applying differencing, the operator helps remove the trend and seasonality from the data, making it stationary. This process allows for better modeling and prediction of the time series.

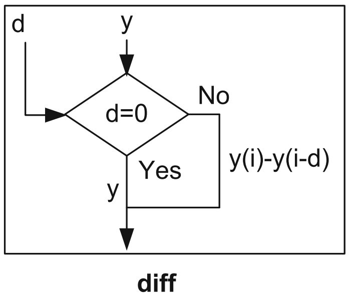


Fig. 2 The flow chart of the Differencing Operator

3.3 The Integration of Differencing Operator into the Sliding Window Procedures

Based on the presentation above, the Differencing Operator has the potential to significantly impact the forecast of time series data. Therefore, in this study, the author proposes the utilization of the Differencing Operator integrated into the Sliding Window Procedure for ensemble learning algorithms in the case of peak load forecasting, as illustrated in Fig. 3. Figure 3 depicts the training and testing process of the ensemble learning algorithm. First, ensemble algorithms are trained using $\{X_{train}, Y_{train}\}$ as input and output variables, which generates a regression model called $\boldsymbol{mdl_i^{(d)}}$. Secondly, this trained regression model, $\boldsymbol{mdl_i^{(d)}}$, produces the output variable $\widehat{Y}$ corresponding to X_{test} in the testing process. The error rate (such as MAPE) between the predict value $\widehat{Y}$ and real value Y_{test} values is used to evaluate the effectiveness of the ensemble algorithms.

The pseudocode for the training process is shown in Fig. 4. The input data for this process is assigned to training data, The output of the training stage is a trained model, $\boldsymbol{mdl_i^{(d)}}$, where subscript i represents one case of the combination of hyperparameters, and superscript (d) refers to the differencing order d. The training process is performed as follows:

- The original data is differenced according to the order d, as defined in Eq. (2).
- Transforming data into input-target pairs: The input X_{train} and output Y_{train} for the training process are established using the Sliding Window procedure discussed earlier in Sect. 3.1.
- Defining the ensemble model: The ensemble algorithms, namely GBDT, XGBoost, LightGBM, and CatBoost models, are defined within the Python environment for this research. The corresponding libraries used are sklearn, xgboost, lightgbm, and catboost.
- The model training is conducted using the input-target data (X_{train} and Y_{train}), which correspond to the defined model from the previous step.

The pseudocode for the testing process is presented in Fig. 5. During the testing process, the input consists of the testing data, the trained model$\boldsymbol{mdl_i^{(d)}}$, and the training data used to invert the Differencing Operator. The testing process follows these main steps:

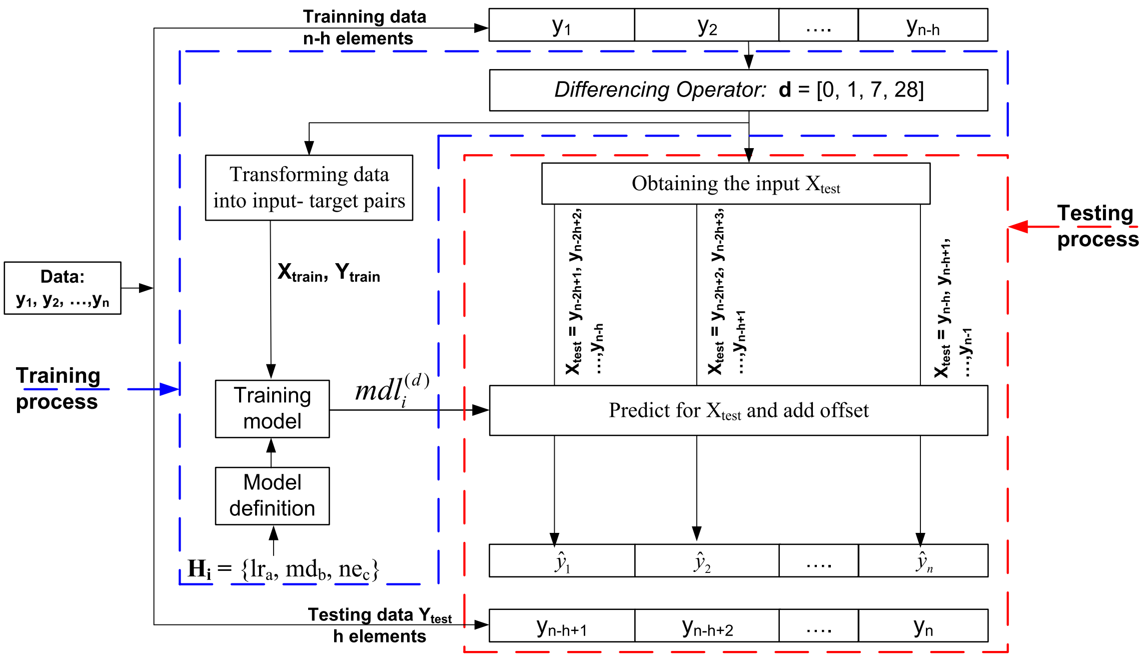

Trainning data
n-h elements
y₁
y₂
….
yₙ₋ₕ
Differencing Operator: d = [0, 1, 7, 28]
Transforming data into input- target pairs
Obtaining the input X_test
Testing process
Data: y₁, y₂, …,yₙ
X_train, Y_train
X_test = yₙ₋₂ₕ₊₁, yₙ₋₂ₕ₊₂, …,yₙ₋ₕ
X_test = yₙ₋₂ₕ₊₂, yₙ₋₂ₕ₊₃, …,yₙ₋ₕ₊₁
X_test = yₙ₋ₕ, yₙ₋ₕ₊₁, …,yₙ₋₁
Training process
Training model
mdl_i^(d)
Predict for X_test and add offset
Model definition
ŷ₁
ŷ₂
….
ŷₙ
H_i = {lr_a, md_b, ne_c}
Testing data Y_test
h elements
yₙ₋ₕ₊₁
yₙ₋ₕ₊₂
….
yₙ

Fig. 3 The training and testing process of the ensemble learning algorithm

Fig. 4 The pseudocode of training stage

```
Pseudocode of training process
Input:
- Training data [$y_1, y_2, \ldots, y_{n-h}$]
- $H_i$: combination of tuning hyperparameters, $H_i = \{lr_a, md_b, ne_c\}$
- d = [0, 1, 7, 28]
∀:
- a = 1: A, b= 1: B, c=1: C, where A, B, C are the number of elements of the hyperparameters lr, md, and ne surveyed in the study.
i = [1:I], where I = AxBxC is the total number of elements of the combination of hyperparameters lr, md, ne.

1:  Differencing Operator :
    If d > 0:
      training data [j]= training data [j]– training data [j-d]
    where j =d+1: n-h
2:  Transform data into input – target:
    Transform training data into input $X_{train}$ and taget $Y_{train}$.
3:  Define the ensemble model:
    $mdl_i^{(d)}$ = GradientBoostingRegressor (learning_rate=$lr_a$, max_depth=$md_b$, n_estimators=$ne_c$)
    $mdl_i^{(d)}$ = XGBRegressor (learning_rate=$lr_a$, max_depth=$md_b$, n_estimators=$ne_c$)
    $mdl_i^{(d)}$ = LGBMRegressor (learning_rate=$lr_a$, max_depth=$md_b$, n_estimators=$ne_c$)
    $mdl_i^{(d)}$ = CatBoostRegressor (learning_rate=$lr_a$, max_depth=$md_b$, n_estimators=$ne_c$)
4:  Training model
    $mdl_i^{(d)}$.fit($X_{train}$, $Y_{train}$)

Output: $mdl_i^{(d)}$
```

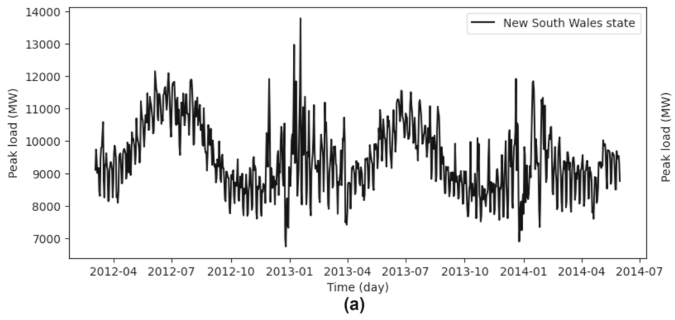
Fig. 5 The pseudocode of testing stage

```
Pseudocode of testing stage
Input:
- training data: [$y_1, y_2, \ldots, y_{n-h}$]
- testing data: [$y_{n-h+1}, y_{n-h+2}, \ldots, y_n$]
- $mdl_i^{(d)}$: model from training stage
1:  For t =1 : h
a:  Difference data and calculate offset
    rolling = training data
    rolling [j]= rolling [j]– rolling [j-d]
    ∀ j=d+1: n-h
    Offset = rolling [-d]
b:  Make the input $X_{test}$ for prediction.
    The input $X_{test}$ is generated by applying the Sliding Window procedure to rolling data.
c:  Predict for $X_{test}$ and add offset
    $\hat{y}(t)$= $mdl_i^{(d)}$.predict($X_{test}$)
    $\hat{y}(t)$ = $\hat{y}(t)$ + Offset

d:  Add actual observation to rolling data for the next prediction value
    rolling data: [$y_1, y_2, \ldots, y_{n-h}, y_{n-h+t}$]

2:  Calculate error rate
    Calculate error rate MAPE between the testing values [$y_{n-h+1}, y_{n-h+2}, \ldots, y_n$] and the prediction values [$\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$].
Output: $MAPE_i^{(d)}$
```

- Obtaining rolling data and differencing offset: The training data is used to obtain the rolling data and determine the differencing offset.
- Obtaining the input X_{test}: The input X_{test} is obtained from the rolling data, using the Sliding Window procedure.
- Obtaining the first predicted value $\widehat{y}_1$: Using the model $mdl_i^{(d)}$ and the X_{test}, the initial predicted value $\widehat{y}_1$ is caculated. It is then adjusted by the differencing offset.
- Updating the rolling data and repeating the process: The rolling data is updated with the actual observation, and the process is repeated for the remaining predicted values $\widehat{y}_i$, i = 2, …h.



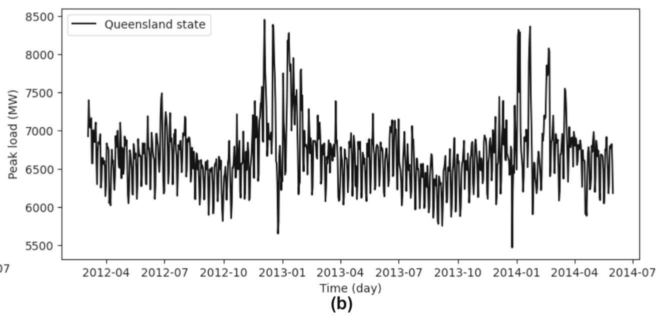


Fig. 6 The daily peak load of New South Wales and Queensland

Table 2 The descriptive statistics of data

Data	Descriptive statistics (MW)						
	Mean	Std	Min	25%	50%	75%	max
NSQ	9453	1051	6747	8696	9322	10,116	13,787
QL	6670	425	5473	6400	6647	6863	8453

The output of the testing process is the error rate, which is calculated based on the real values [$y_{\text{n-h+1}}, y_{\text{n-h+2}}, \ldots, y_{\text{n}}$] and the predicted values [$\widehat{y}_1, \widehat{y}_2, \ldots, \widehat{y}_n$]. In this paper, the mean absolute percentage error (MAPE) is calculated to evaluate forecasting accuracy. The MAPE error rate is expressed by the following formula [44, 45]:

$$\text{MAPE} = \frac{1}{h}\sum_{i=1}^{h}\left|\frac{y_{n-h+i} - \widehat{y}_i}{y_{n-h+i}}\right| \quad (3)$$

Note:

To evaluate the effectiveness of the Differencing Operator based on the Sliding Window Procedure for ensemble algorithms, it is necessary to analyze the performance of these algorithms according to the differencing order d. This is the reason why there is a superscript (d) in the model $\mathbf{mdl}_i^{(d)}$*, and the error rate* $\mathbf{MAPE}_i^{(d)}$ *in Figs. 3, 4, and 5 as presented above.*

Additionally, to enhance the reliability of the results, it is suggested to combine different values of hyperparameters for each ensemble algorithm. That explains why there is the input $H_i = \{lr_a, md_b, ne_c\}$ *in the training process, as well as the subscript (i) in the variable* $\mathbf{mdl}_i^{(d)}$ *and the error rate* $\mathbf{MAPE}_i^{(d)}$.

Thus, based on the procedure outlined in Fig. 3 and the integrated pseudocode in Figs. 4 and 5, the error rate of each ensemble model can be determined by considering specific values of differencing order (d). This allows for the evaluation of the effectiveness of integrating Differencing Operator into the Sliding Window Procedures based on ensemble algorithms.

4 Experimental Study

4.1 Experimental setup

In this study, the author recommends utilizing the daily peak load data of New South Wales (NSW) and Queensland (QL), Australia, for both training and testing. Figure 6 depicts the peak load graph of these two states from March 4, 2012 to May 31, 2014, along with their corresponding characteristics listed in Table 2. The training phase utilizes data from March 4, 2012 to May 3, 2014, while the testing phase encompasses the period from May 4, 2014 to May 31, 2014, covering a duration of 28 days.

To enhance the reliability of the proposed method, it is crucial to explore multiple cases for each ensemble algorithm. In this study, the author suggests simultaneously investigating different combinations of significant common hyperparameters for the GBD, XGBoost, LightBoost, and CatBoost models. These hyperparameters include the learning rate (lr), maximum depth (md), and number of estimators (ne), as discussed in Sects. 2 and 3. The range and the number of survey participants for these hyperparameters are presented in Table 3 below. The total number of combinations for the lr, md, and ne hyperparameters is 2000 cases.

In the present work, the focus is on forecasting daily peak loads. For this purpose, several differencing values are proposed, including $d = 0$ (no differencing), $d = 1$ (first differencing), $d = 7$ (weekly seasonal differencing), and $d = 28$ (monthly seasonal differencing). Additionally, for the experimental application of the proposed algorithm to peak load data, three window sizes have been established:

Table 3 The ranges for hyperparameters

Hyperparameter	Range			Number of elements
	Min	Max	Step	
Learning rate (lr)	0.01	0.2	0.01	A = 20
Max depth (md)	1	10	1	B = 10
Number of estimators (ne)	100	550	50	C = 10

- Window size = 1, which uses the data taken from the previous day for forecasting.
- Window size = 7, which uses the data taken from the previous week.
- Window size = 28, which considers a typical month's data, specifically, from four preceding weeks.

After executing the program and analyzing the results from the mentioned window sizes, the obtained outcomes were quite similar across the board. However, the window size of 7 proved to be the most effective one. Notably, this finding is helpful for the paper focus that is devoted to clarifying the impact of the Differencing Operator. As a result, the window size of 7 was chosen for further study in this research.

The experiments were implemented using the Scikit-learn, math, Matplotlib, and other libraries, as well as the XGBoost, LightGBM, and CatBoost libraries in the Python environment on the Google Colab platform. The runtime type in Colab is TPU with high RAM.

4.2 Evaluation of Error Rates

Figure 7 displays a boxplot of the error rate (MAPE) between the predicted value ($\widehat{Y}$) and the actual value (Ytest) for different values of differencing order d (0, 1, 7, 28). The result corresponds to the GBDT, XGBoost, LightBoost, and CatBoost models for the New South Wales and Queensland data cases.

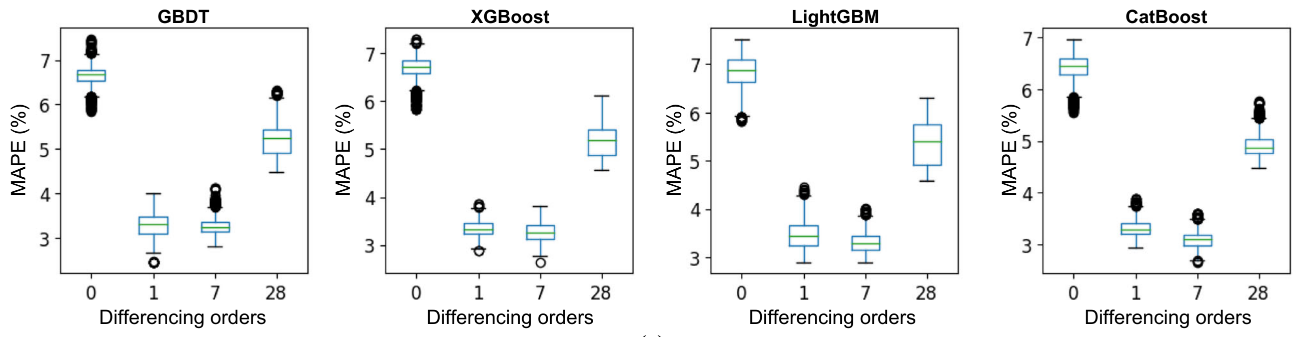
Table 4 presents statistics for each set of differencing order d (d = 0, 1, 7, 28) shown in Fig. 7. For each set, five statistical values are provided, including minimum, 25th percentile, 50th percentile, 75th percentile, and maximum. For example, in the upper-left subfigure of Fig. 7 (GBDT model, New South Wales data), a differencing order d = 0 yields statistical values of 5.84 (minimum), 6.54 (25th percentile), 6.68 (50th percentile), 6.79 (75th percentile), and 7.47 (maximum). Similarly, for the last subfigure on the bottom right (CatBoost model, Queensland data), a differencing order d = 28 gives statistic values of 3.21 (minimum), 3.51 (25th percentile), 3.59 (50th percentile), 3.68 (75th percentile), and 4.32 (maximum).

An in-depth analysis of Figs. 7 and Table 4 reveals that the application of the Differencing Operator to the input data (d = 1, 7, 28) leads to significantly better results, with a drastic reduction in prediction error compared to using the original data (d = 0). Specifically, when examining the GBDT model and New South Wales data, Fig. 7 and Table 4 show that using the original data yields excessively high forecast error values (minimum: 5.84, 25th percentile: 6.54, 50th percentile: 6.68, 75th percentile: 6.79, maximum: 7.47), whereas cases with d = 1 (minimum: 2.45, 25th percentile: 3.11, 50th percentile: 3.30, 75th percentile: 3.49, maximum: 4.01), d = 7 (minimum: 2.82, 25th percentile: 3.13, 50th percentile: 3.23, 75th percentile: 3.36, maximum: 4.11), and d = 28 (minimum: 4.47, 25th percentile: 4.92, 50th percentile: 5.23, 75th percentile: 5.43, maximum: 6.31) demonstrate significantly improved results. Similar trends are observed in all other cases. Moreover, a comparison of the error values across different Differencing Operator cases (d = 1, 7, 28) highlights that the most optimal results are achieved when d = 7.

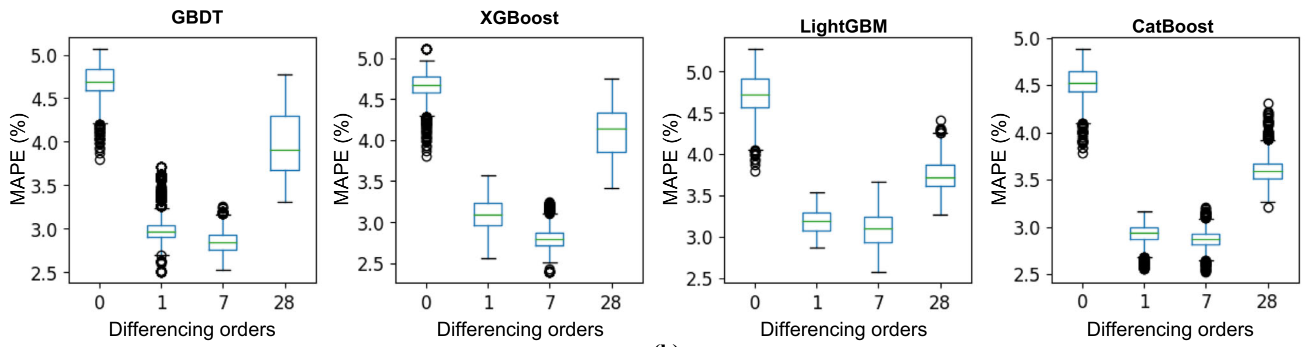
To accurately evaluate the impact of the Differencing Operator, the next step focuses on calculating the ratio of the error rate between the differencing cases (d = 1, 7, 28) and the original data case (d = 0). Figure 8 displays a boxplot of the error rate ratio for the GBDT, XGBoost, LightGBM, and CatBoost models applied to the New South Wales and Queensland data. The statistical values for each column in Fig. 8 are summarized in Table 5.

The results in Figs. 8 and Table 5 clearly demonstrate the fluctuation range of the error rate ratio for both the New South Wales and Queensland data. For the New South Wales data, the ratio of the error rate ranges from 0.36 to 0.68 for the minimum statistic, 0.46 to 0.75 for the 25th percentile, 0.48 to 0.78 for the median (50th percentile), 0.50 to 0.81 for the 75th percentile, and 0.59 to 0.93 for the maximum statistic. Similarly, for the Queensland data, the ratio ranges from 0.51 to 0.73 for the minimum statistic, 0.58 to 0.85 for the 25th percentile, 0.60 to 0.88 for the median, 0.62 to 0.92 for the 75th percentile, and 0.80 to 1.16 for the maximum statistic.

For the New South Wales data, all error ratios of the Differencing Operator (d = 1, 7, 28) to the original data (d = 0) are less than 1. However, in the Queensland dataset, there are instances where the ratio 28/0 (d = 28/d = 0) exceeds 1, as detailed in Tables 6 below. Table 6 reveals 10 instances in the GBDT model, 12 instances in the XGBoost model, 5 instances in the LightGBM model, and 14 instances in the



(a)



(b)

Fig. 7 The error rates for differencing orders of 0, 1, 7, and 28: (**a**) New South Wales, (**b**) Queensland

Table 4 The descriptive statistics of error rate

Algorithms	d	New South Wales					Queensland				
		Min	25%	50%	75%	Max	Min	25%	50%	75%	Max
GBDT	0	5.84	6.54	6.68	6.79	7.47	3.80	4.59	4.69	4.84	5.07
	1	2.45	3.11	3.30	3.49	4.01	2.51	2.90	2.97	3.04	3.71
	7	2.82	3.13	3.23	3.36	4.11	2.52	2.76	2.84	2.93	3.26
	28	4.47	4.92	5.23	5.43	6.31	3.31	3.68	3.90	4.30	4.77
XGBoost	0	5.82	6.59	6.72	6.84	7.30	3.80	4.58	4.67	4.78	5.11
	1	2.89	3.24	3.32	3.45	3.86	2.55	2.96	3.09	3.23	3.57
	7	2.64	3.12	3.26	3.41	3.82	2.39	2.72	2.79	2.87	3.25
	28	4.57	4.89	5.19	5.40	6.11	3.42	3.86	4.14	4.33	4.75
LightGBM	0	5.82	6.63	6.89	7.10	7.52	3.80	4.57	4.71	4.91	5.27
	1	2.90	3.25	3.44	3.66	4.45	2.87	3.07	3.19	3.29	3.54
	7	2.89	3.16	3.29	3.45	4.00	2.58	2.93	3.10	3.23	3.66
	28	4.58	4.91	5.39	5.75	6.31	3.26	3.61	3.72	3.86	4.41
CatBoost	0	5.55	6.29	6.45	6.59	6.97	3.79	4.43	4.53	4.66	4.89
	1	2.93	3.19	3.29	3.41	3.88	2.56	2.88	2.95	3.00	3.17
	7	2.65	2.99	3.09	3.19	3.59	2.53	2.82	2.87	2.93	3.21
	28	4.47	4.76	4.87	5.03	5.77	3.21	3.51	3.59	3.68	4.32

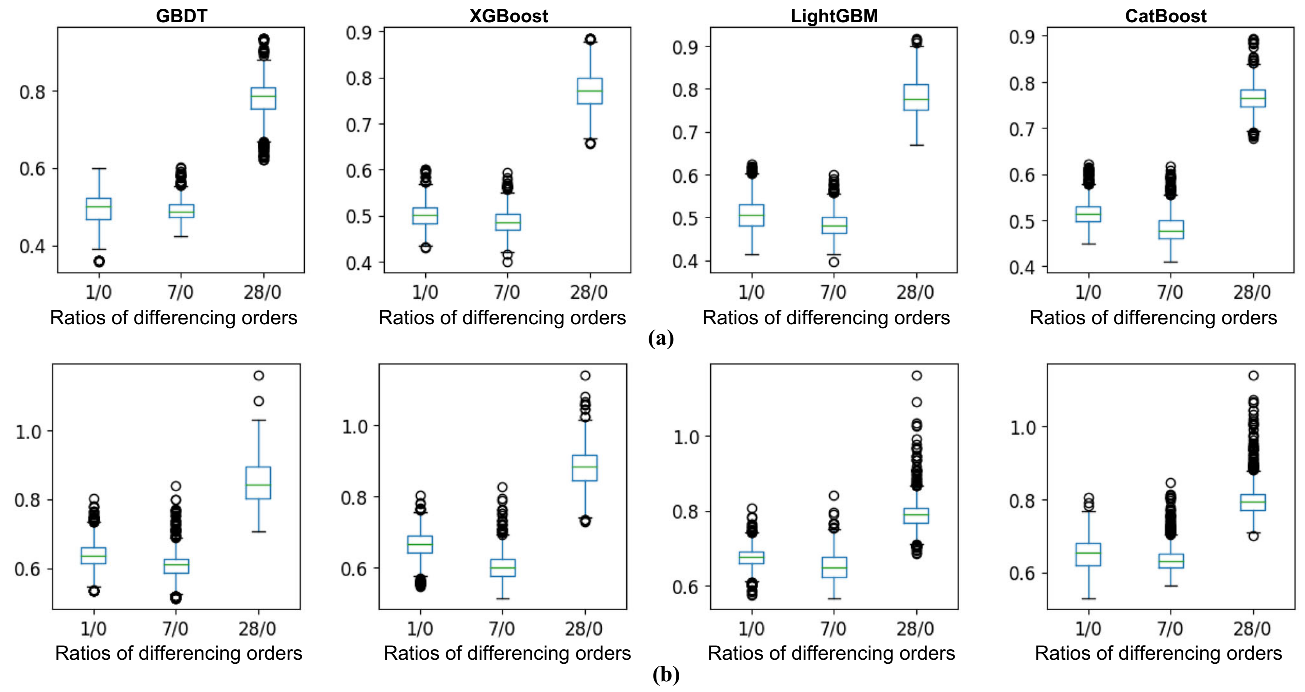


Fig. 8 The error ratio between differencing order d of 1, 7, 28 and 0: (**a**) New South Wales, (**b**) Queensland

Table 5 The descriptive statistics of error rate

Algorithms	Ratios	New South Wales					Queensland				
		Min	25%	50%	75%	Max	Min	25%	50%	75%	Max
GBDT	1/0	0.36	0.47	0.50	0.52	0.60	0.54	0.61	0.64	0.66	0.80
	7/0	0.42	0.47	0.49	0.50	0.60	0.51	0.59	0.61	0.63	0.84
	28/0	0.62	0.75	0.78	0.81	0.93	0.71	0.80	0.84	0.89	1.16
XGBoost	1/0	0.43	0.48	0.50	0.52	0.60	0.55	0.64	0.67	0.69	0.80
	7/0	0.40	0.47	0.49	0.50	0.59	0.52	0.58	0.60	0.62	0.83
	28/0	0.66	0.74	0.77	0.80	0.89	0.73	0.85	0.88	0.92	1.14
LightGBM	1/0	0.41	0.48	0.51	0.53	0.62	0.58	0.66	0.68	0.69	0.81
	7/0	0.40	0.46	0.48	0.50	0.60	0.57	0.62	0.65	0.68	0.84
	28/0	0.67	0.75	0.78	0.81	0.92	0.69	0.77	0.79	0.81	1.16
CatBoost	1/0	0.45	0.50	0.51	0.53	0.62	0.53	0.62	0.65	0.68	0.81
	7/0	0.41	0.46	0.48	0.50	0.62	0.56	0.61	0.63	0.65	0.85
	28/0	0.68	0.75	0.76	0.78	0.89	0.70	0.77	0.79	0.81	1.14

CatBoost model where the ratios are greater than 1. Considering the total of 2000 combinations of hyperparameters (lr, md, and ne) for each model, the number of cases where the ratio exceeds 1 is exceptionally small. This indicates that the utilization of the Differencing Operator (d = 1, 7, 28) can effectively enhance the precision of the forecasting process for ensemble algorithms. The data analysis also demonstrates that the differencing order of 7 may result in the smallest error ratio compared to the differencing orders of 1 or 28 for most values of the min, 25th, 50th, 75th, and max statistics.

In conclusion, the results confirm that the utilization of the Differencing Operator ($d = 1, 7, 28$) has a positive impact on reducing errors and improving the accuracy of the forecasting process for ensemble algorithms. The analysis also suggests that a differencing order of 7 tends to yield the smallest error ratio compared to orders of 1 or 28 for various statistical values.

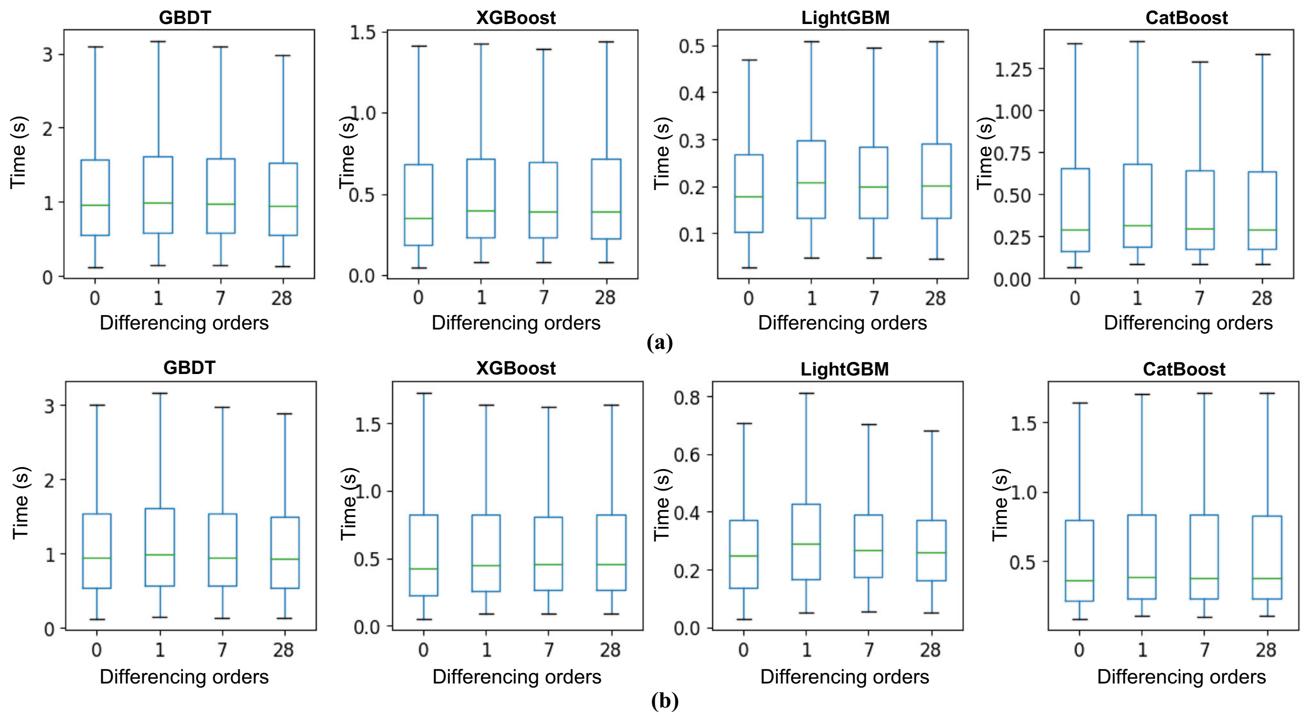


Fig. 9 The execution time for differencing orders of 0, 1, 7, and 28: (**a**) New South Wales, (**b**) Queensland

4.3 Evaluation of Execution Time

Figure 9 illustrates a boxplot representing the execution time for different differencing orders ($d = 0, 1, 7, 28$) corresponding to the GBDT, XGBoost, LightGBM, and CatBoost models applied to the New South Wales and Queensland data cases. Table 7 presents the statistical values for each set of differencing orders ($d = 0, 1, 7, 28$) as shown in Fig. 9.

A detailed analysis of Fig. 9 and Table 7 reveals that the application of the Differencing Operator ($d = 1, 7, 28$) does not significantly increase the execution time of the program compared to the case where the original data ($d = 0$) is used. For example, let's consider the GBDT network with the New South Wales dataset. The results presented in Fig. 9 and Table 7 show that the execution time statistic values for the original data case ($d = 0$) are as follows: minimum: 0.12 s, 25th percentile: 0.55 s, 50th percentile: 0.95 s, 75th percentile: 1.57 s, and maximum: 3.14 s. These values remain largely unchanged when compared to the cases of d = 1 (minimum: 0.14, 25th percentile: 0.58, 50th percentile: 0.99, 75th percentile: 1.62, and maximum: 3.19), d = 7 (minimum: 0.14, 25th percentile: 0.57, 50th percentile: 0.98, 75th percentile: 1.59, and maximum: 3.15), and d = 28 (minimum: 0.13, 25th percentile: 0.55, 50th percentile: 0.94, 75th percentile: 1.53, and maximum: 3.13). And all other cases have similar results.

In addition, Fig. 10 presents a boxplot illustrating the execution time ratios between the Differencing Operator ($d = 1$, 7, 28) and the original data case ($d = 0$). The corresponding statistical values for each column in Fig. 10 are summarized in Table 8. For instance, in the 50th percentile statistical case (median values), the error ratio of the execution time with the Differencing Operator to that with the original data ranges from [0.99 to 1.20] for all New South Wales data cases. Similarly, for the Queensland data, the ratio fluctuates within the range of [0.99–1.18]. These findings indicate that there is no significant difference in the execution time when considering differencing orders of 1, 7, or 28 for the GBDT, XGBoost, LightGBM, and CatBoost models, respectively. Figures 10 and Table 8 consistently demonstrate that the execution time remains largely unchanged when the Differencing Operator is applied.

5 Conclusion

In this study, the author suggests the combination of the input data Differencing Operator with the Sliding Window procedure for ensemble learning algorithms. The objective was to assess the error rate and execution time for the GBDT, XGBoost, LightGBM, and CatBoost models in the forecasting process. Extensive exploration of hyperparameter combinations, such as learning rate, max depth, and number of estimations, was conducted to evaluate the effectiveness of the proposed approach. The results clearly demonstrated that implementing the input data difference approach ($d =$

Table 6 The list of ratios 28/0 greater than 1 for the Queensland data

No	Hyperparameters			Error			Model
	lr	md	ne	d = 0	d = 28	(d = 0)/(d = 28)	
1	0.01	1	100	3.7953	4.4034	1.1602	GBDT
2	0.01	1	150	3.8668	4.2029	1.0869	
3	0.01	1	200	3.9316	4.0228	1.0232	
4	0.01	2	100	3.9105	4.0277	1.0300	
5	0.01	3	100	3.9805	4.0216	1.0103	
6	0.01	5	100	4.0435	4.0515	1.0020	
7	0.01	8	100	4.1447	4.2740	1.0312	
8	0.01	9	100	4.2177	4.3230	1.0250	
9	0.01	10	100	4.2584	4.3789	1.0283	
10	0.02	1	100	3.9334	4.0198	1.0220	
1	0.01	1	100	3.7986	4.3297	1.1398	XGBoost
2	0.01	1	150	3.8684	4.1784	1.0801	
3	0.01	1	200	3.9284	4.1040	1.0447	
4	0.01	1	250	3.9780	4.0337	1.0140	
5	0.01	2	100	3.9098	4.1231	1.0545	
6	0.01	3	100	3.9773	4.2364	1.0651	
7	0.01	4	100	4.0218	4.2643	1.0603	
8	0.01	5	100	4.0430	4.2680	1.0557	
9	0.01	6	100	4.1000	4.1924	1.0225	
10	0.01	7	100	4.1834	4.1926	1.0022	
11	0.01	10	100	4.2971	4.3942	1.0226	
12	0.02	1	100	3.9303	4.1058	1.0446	
1	0.01	1	100	3.7962	4.4104	1.1618	LightGBM
2	0.01	1	150	3.8691	4.2150	1.0894	
3	0.01	1	200	3.9309	4.0367	1.0269	
4	0.01	2	100	3.9079	4.0373	1.0331	
5	0.02	1	100	3.9339	4.0296	1.0243	
1	0.01	1	100	3.7874	4.3185	1.1402	Catboost
2	0.01	1	150	3.8411	4.0992	1.0672	
3	0.01	1	200	3.9077	3.9326	1.0064	
4	0.01	2	100	3.8387	4.0052	1.0434	
5	0.01	3	100	3.8823	4.0093	1.0327	
6	0.01	4	100	3.8954	3.9550	1.0153	
7	0.01	5	100	3.9074	3.9374	1.0077	
8	0.01	6	100	3.9086	3.9484	1.0102	
9	0.01	7	100	3.9111	3.9899	1.0201	
10	0.01	8	100	3.9051	4.0765	1.0439	
11	0.01	9	100	3.9119	4.1620	1.0639	
12	0.01	10	100	3.9029	4.1876	1.0729	
13	0.01	10	150	4.0044	4.0107	1.0016	
14	0.02	1	100	3.9149	3.9219	1.0018	

Table 7 The descriptive statistics of execution time

Algorithms	d	New South Wales						Queensland			
		Min	25%	50%	75%	Max	Min	25%	50%	75%	Max
GBDT	0	0.12	0.55	0.95	1.57	3.14	0.12	0.54	0.94	1.53	3.06
	1	0.14	0.58	0.99	1.62	3.19	0.14	0.57	0.98	1.61	3.47
	7	0.14	0.57	0.98	1.59	3.15	0.14	0.56	0.95	1.54	2.97
	28	0.13	0.55	0.94	1.53	3.13	0.13	0.54	0.93	1.49	2.93
XGBoost	0	0.05	0.19	0.35	0.69	7.19	0.05	0.23	0.42	0.83	9.43
	1	0.08	0.23	0.40	0.71	4.80	0.09	0.26	0.45	0.82	5.07
	7	0.08	0.23	0.39	0.70	4.73	0.09	0.26	0.46	0.81	4.97
	28	0.08	0.23	0.39	0.71	4.73	0.09	0.27	0.45	0.82	5.24
LightGBM	0	0.03	0.10	0.18	0.27	0.47	0.03	0.14	0.25	0.37	0.76
	1	0.05	0.13	0.21	0.30	0.51	0.05	0.17	0.29	0.43	0.81
	7	0.05	0.13	0.20	0.28	0.53	0.05	0.17	0.27	0.39	0.70
	28	0.05	0.13	0.20	0.29	0.54	0.05	0.16	0.26	0.37	0.74
CatBoost	0	0.07	0.16	0.29	0.66	3.65	0.08	0.21	0.36	0.80	4.48
	1	0.09	0.19	0.31	0.68	3.80	0.10	0.23	0.38	0.83	4.39
	7	0.08	0.17	0.29	0.64	3.36	0.10	0.23	0.38	0.83	4.46
	28	0.08	0.17	0.29	0.64	3.43	0.10	0.23	0.38	0.83	4.38

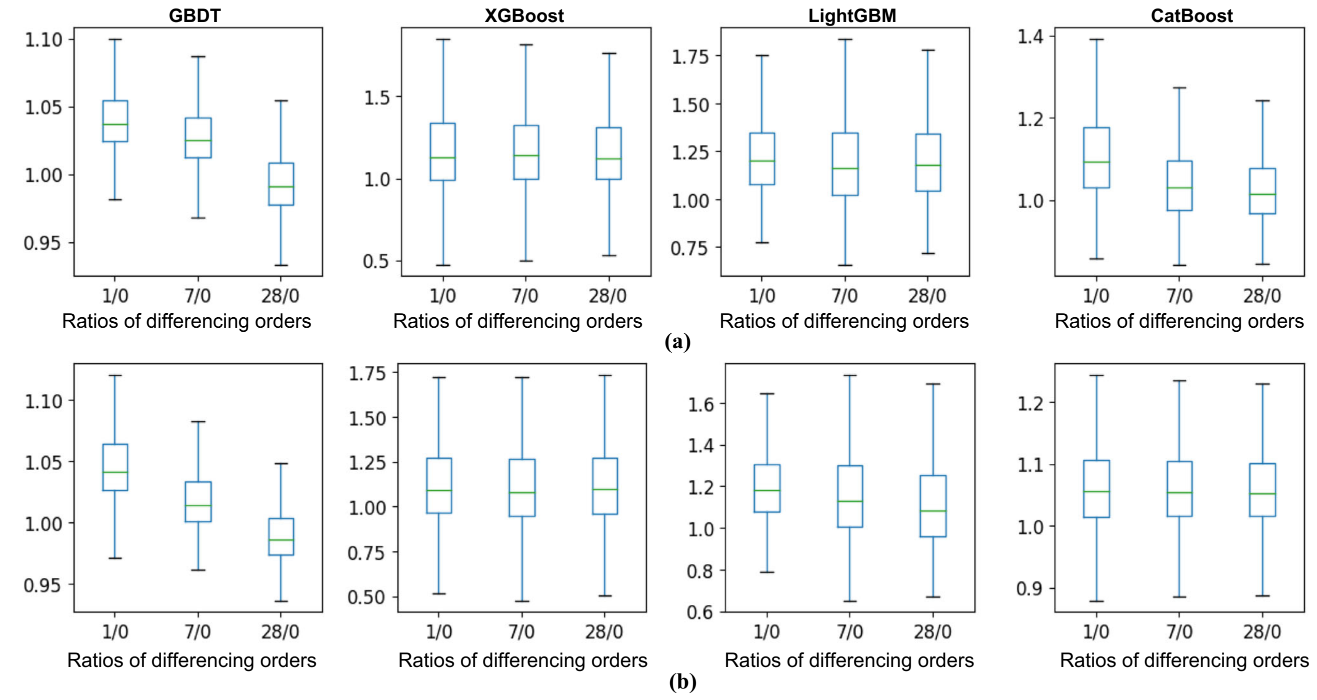


Fig. 10 The time ratio between differencing order of 1, 7, 28 and 0: (**a**) New South Wales, (**b**) Queensland

Table 8 The descriptive statistics of time ratio

Algorithms	Ratios	New South Wales					Queensland				
		Min	25%	50%	75%	Max	Min	25%	50%	75%	Max
GBD	1/0	0.96	1.02	1.04	1.05	1.20	0.97	1.03	1.04	1.06	1.55
	7/0	0.90	1.01	1.02	1.04	1.20	0.89	1.00	1.01	1.03	1.17
	28/0	0.92	0.98	0.99	1.01	1.17	0.91	0.97	0.99	1.00	1.16
XGBoost	1/0	0.03	0.99	1.13	1.34	11.33	0.03	0.97	1.09	1.27	11.80
	7/0	0.03	0.99	1.14	1.33	17.43	0.03	0.95	1.08	1.27	12.32
	28/0	0.03	1.00	1.12	1.31	13.14	0.03	0.96	1.10	1.27	17.00
LightGBM	1/0	0.47	1.08	1.20	1.35	2.19	0.37	1.08	1.18	1.31	2.20
	7/0	0.48	1.02	1.16	1.35	2.11	0.41	1.01	1.13	1.30	2.19
	28/0	0.48	1.05	1.18	1.34	2.21	0.35	0.96	1.09	1.25	2.10
CatBoost	1/0	0.66	1.03	1.09	1.18	1.59	0.16	1.01	1.06	1.11	1.46
	7/0	0.66	0.98	1.03	1.10	1.33	0.16	1.02	1.05	1.10	1.43
	28/0	0.61	0.97	1.02	1.08	1.37	0.16	1.02	1.05	1.10	1.51

1, 7, 28) led to a significant reduction in prediction error. Furthermore, it was observed that the execution time only experienced a slight increase when employing the data partitioning approach. In conclusion, the integration of the Differencing Operator into the Sliding Window Procedure for ensemble learning presents a promising solution to address technical challenges, particularly in the domain of peak load forecasting. This result lays the foundation for the author to further develop the proposed algorithm toward various machine learning models, particularly deep learning models. Additionally, it enables the extension of the algorithm's application in diverse types of time series data, such as financial and weather data. Moreover, exploring its effectiveness in real-time data processing or under different operational conditions presents an important challenge.

References

1. Singh, AK.; Ibraheem, Khatoon S.; Muazzam, M.; Chaturvedi, DK.: Load forecasting techniques and methodologies: A review. ICPCES 2012 - 2012 2nd Int Conf Power, Control Embed Syst. 2012;(March 2015). https://doi.org/10.1109/ICPCES.2012.6508132.
2. Zhang, Y.; Wen, H.; Wu, Q.; Ai, Q.: Optimal adaptive prediction intervals for electricity load forecasting in distribution systems via reinforcement learning. IEEE Trans. Smart Grid. **14**(4), 3259–3270 (2023). https://doi.org/10.1109/TSG.2022.3226423
3. Guo, W.; Che, L.; Shahidehpour, M.; Wan, X.: Machine-Learning based methods in short-term load forecasting. Electr. J. **34**(1), 106884 (2021). https://doi.org/10.1016/j.tej.2020.106884
4. Wu, D.; Lin, W.: Efficient residential electric load forecasting via transfer learning and graph neural networks. IEEE Trans. Smart Grid. **14**(3), 2423–2431 (2023). https://doi.org/10.1109/TSG.2022.3208211
5. Zawadali, M.; Nasmussakib Khan Shabbir, M.; Sifatulalam Chowdhury, M.; Ghosh, A;, Liang, X.: Regression models of critical parameters affecting peak load demand forecasting. Can Conf Electr Comput Eng. 2018;2018-May:31–34. https://doi.org/10.1109/CCECE.2018.8447786.
6. Hsu, C.C.; Chen, C.Y.: Regional load forecasting in Taiwan—applications of artificial neural networks. Energy Convers. Manag. **44**(12), 1941–1949 (2003). https://doi.org/10.1016/S0196-8904(02)00225-X
7. Kazemzadeh, M.R.; Amjadian, A.; Amraee, T.: A hybrid data mining driven algorithm for long term electric peak load and energy demand forecasting. Energy **204**, 117948 (2020). https://doi.org/10.1016/j.energy.2020.117948
8. Mado, I.; Soeprijanto, A.; Suhartono, S.: Applying of double seasonal ARIMA model for electrical power demand forecasting at PT PLN Gresik Indonesia. Int. J. Electr. Comput. Eng. (IJECE). **8**(6), 4892 (2018)
9. Lim, P.Y.; Nayar, C.V.: Solar irradiance and load demand forecasting based on single exponential smoothing method. Int. J. Eng. Technol. **4**(4), 451 (2012)
10. Ji, P.; Xiong, D.; Wang, P.; Chen, J.: A study on exponential smoothing model for load forecasting. Asia-Pacific Power Energy Eng. Conf., APPEEC. **1**, 1–4 (2012)
11. Yue, LYL.; Zhang, YZY.; Xie, HXH.; Zhong, QZQ.: The fuzzy logic clustering neural network approach for middle and long term load forecasting. 2007 IEEE International Conference on Grey Systems and Intelligent Services. 963–7 (2007)
12. Nazarko, J.: The fuzzy regression approach to peak load estimation in power distribution systems. IEEE Trans. Power Syst. **14**(3), 809–814 (1999). https://doi.org/10.1109/59.780890
13. Raza, M.Q.; Khosravi, A.: A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings. Renew. Sustain. Energy Rev. **50**, 1352–1372 (2015). https://doi.org/10.1016/j.rser.2015.04.065
14. Arvanitidis, A.I.; Bargiotas, D.; Daskalopulu, A.; Laitsos, V.M.; Tsoukalas, L.H.: Enhanced short-term load forecasting using artificial neural networks. Energies **14**(22), 1–14 (2021)
15. Baliyan, A.; Gaurav, K.; Kumar Mishra, S.: A review of short term load forecasting using artificial neural network models. Procedia Comput. Sci. **48**, 121–125 (2015). https://doi.org/10.1016/j.procs.2015.04.160

16. Arnob, S.S.; Arefin, A.I.M.S.; Saber, A.Y.; Mamun, K.A.: Energy demand forecasting and optimizing electric systems for developing countries. IEEE Access **11**, 39751–39775 (2023)
17. Ceperic, E.; Ceperic, V.; Baric, A.: A strategy for short-term load forecasting by support vector regression machines. IEEE Trans. Power Syst. **4**, 4356–4364 (2013). https://doi.org/10.1109/TPWRS.2013.2269803
18. Wu, L.; Shahidehpour, M.: A hybrid model for integrated day-ahead electricity price and load forecasting in smart grid. IET Gener. Transm. Distrib. **8**(12), 1937–1950 (2014)
19. Laouafi, A.; Mordjaoui, M.; Laouafi, F.; Boukelia, T.E.: Daily peak electricity demand forecasting based on an adaptive hybrid two-stage methodology. Int. J. Electr. Power Energy Syst. **77**, 136–144 (2016). https://doi.org/10.1016/j.ijepes.2015.11.046
20. Campbell PRJ. A hybrid modelling technique for load forecasting. 2007 IEEE Canada Electrical Power Conference, EPC 2007.;435–9 (2007)
21. Wang, L.; Mao, S.; Wilamowski, B.M.; Nelms, R.M.: Ensemble learning for load forecasting. IEEE Trans. Green Commun. Netw. **4**(2), 616–628 (2020). https://doi.org/10.1109/TGCN.2020.2987304
22. Guo, H.; Tang, L.; Peng, Y.: Ensemble deep learning method for short-term load forecasting. Proc - 14th Int Conf Mob Ad-Hoc Sens Networks, MSN 2018.;(DI):86–90. (2018) https://doi.org/10.1109/MSN.2018.00021.
23. Chen, B.; Lin, R.; Zou, H.: A short term load periodic prediction model based on GBDT. Int Conf Commun Technol Proceedings, ICCT. 2019-:1402–1406. (2019) https://doi.org/10.1109/ICCT.2018.8600009.
24. Liu, S.; Cui, Y.; Ma, Y.; Liu, P.: Short-term load forecasting based on gbdt combinatorial optimization. 2nd IEEE Conf Energy Internet Energy Syst Integr EI2 2018 - Proc. Published online (2018):1-5. https://doi.org/10.1109/EI2.2018.8582108
25. Liao, X.; Cao, N.; Li, M.; Kang, X.: Research on short-term load forecasting using XGBoost based on similar days. Proc - 2019 Int Conf Intell Transp Big Data Smart City, ICITBS 2019. Published online (2019):675-678. https://doi.org/10.1109/ICITBS.2019.00167
26. Tran, N.T.; Tran, T.T.G.; Nguyen, T.A.; Lam, M.B.: A new grid search algorithm based on XGBoost model for load forecasting. Bull Electr. Eng. Inf. **12**(4), 1857–1866 (2023). https://doi.org/10.11591/eei.v12i4.5016
27. Hao, M.; Tian, Y.; Gao, J.; Wang, Y.; Tian, R.: Classification of short-term loads of enterprises using LightGBM. IOP Conf Ser Earth Environ Sci. 2020;526(1). https://doi.org/10.1088/1755-1315/526/1/012179.
28. Wang, Y.; Chen, J.; Chen, X., et al.: Short-Term Load Forecasting for Industrial Customers Based on TCN-LightGBM. IEEE Trans. Power Syst. **8950**, 1–1 (2020). https://doi.org/10.1109/tpwrs.2020.3028133
29. Yang, X.; Chen, Z.: A hybrid short-term load forecasting model based on CatBoost and LSTM. 2021 IEEE 6th Int Conf Intell Comput Signal Process ICSP 2021. 2021;(Icsp):328–332. https://doi.org/10.1109/ICSP51882.2021.9408768.
30. Zhang, C.; Chen, Z.; Zhou, J.: Research on short-term load forecasting using K-means clustering and catboost integrating time series features. Chinese Control Conf CCC. 2020;2020-July(July 2016):6099–6104. https://doi.org/10.23919/CCC50068.2020.9188856.
31. Semmelmann, L.; Henni, S.; Weinhardt, C.: Load forecasting for energy communities: a novel LSTM-XGBoost hybrid model based on smart meter data. Energy Inf. **5**, 1–21 (2022). https://doi.org/10.1186/s42162-022-00212-9
32. Park, S.; Jung, S.; Jung, S.; Rho, S.; Hwang, E.: Sliding window-based LightGBM model for electric load forecasting using anomaly repair. J. Supercomput. **0123456789**, 27–30 (2021). https://doi.org/10.1007/s11227-021-03787-4
33. Khwaja, A.S.; Anpalagan, A.; Naeem, M.; Venkatesh, B.: Joint bagged-boosted artificial neural networks: Using ensemble machine learning to improve short-term electricity load forecasting. Electr. Power Syst. Res. **2020**(179), 106080 (2019)
34. Zhao, X.; Xia, N.; Xu, Y.; Huang, X.; Li, M.: Mapping population distribution based on XGBoost using multisource data. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. **14**, 11567–11580 (2021). https://doi.org/10.1109/JSTARS.2021.3125197
35. Liang, W.; Luo, S.; Zhao, G.; Wu, H.: Predicting hard rock pillar stability using GBDT, XGBoost, and LightGBM algorithms. Mathematics **8**(5), 1–17 (2020). https://doi.org/10.3390/MATH8050765
36. Ben, J.S.; Mefteh-Wali, S.; Viviani, J.L.: Forecasting gold price with the XGBoost algorithm and SHAP interaction values. Ann. Oper. Res. (2022). https://doi.org/10.1007/s10479-021-04187-w
37. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Ann. Stat. **29**(5), 1189–1232 (2001). https://doi.org/10.1214/aos/1013203451
38. Friedman, J.H.: Stochastic gradient boosting. Comput. Stat. Data Anal. **38**(4), 367–378 (2002). https://doi.org/10.1016/S0167-9473(01)00065-2
39. Obiora, CN.; Ali. A.; Hasan AN.: Implementing extreme gradient boosting (xgboost) algorithm in predicting solar irradiance. 2021 IEEE PES/IAS PowerAfrica, PowerAfrica 2021. (2021);1–5
40. Machado, MR.; Karray, S.; De Sousa, IT.: LightGBM: An effective decision tree gradient boosting method to predict customer loyalty in the finance industry. 14th international conference on computer science and education, ICCSE 2019. (2019);(Iccse):1111–6.
41. Hancock, J.T.; Khoshgoftaar, T.M.: CatBoost for big data: an interdisciplinary review. J. Big Data. **7**(1), 94 (2020)
42. Wang, L.; Wu, J.; Zhang, W.; Wang, L.; Cui, W.: Efficient seismic stability analysis of embankment slopes subjected to water level changes using gradient boosting algorithms. Front. Earth Sci. **9**, 1–9 (2021)
43. Yin, L.; Ma, P.; Deng, Z.: Jlgbmloc—a novel high-precision indoor localization method based on lightgbm. Sensors (2021). https://doi.org/10.3390/s21082722
44. Yu, C.N.; Mirowski, P.; Ho, T.K.: A sparse coding approach to household electricity demand forecasting in smart grids. IEEE Trans. Smart Grid. **8**(2), 738–748 (2017). https://doi.org/10.1109/TSG.2015.2513900
45. Hoverstad, B.A.; Tidemann, A.; Langseth, H.; Ozturk, P.: Short-term load forecasting with seasonal decomposition using evolution for parametershould be a spacetuning. IEEE Trans. Smart Grid (2015). https://doi.org/10.1109/TSG.2015.2395822