

## Article

# Short-Term Wind Power Prediction Based on a Modified Stacking Ensemble Learning Algorithm

Yankun Yang , Yuling Li, Lin Cheng  and Shiyong Yang \* 

College of Electrical Engineering, Zhejiang University, Hangzhou 310027, China

\* Correspondence: eesyang@zju.edu.cn; Tel.: +86-0571-87952498

**Abstract:** A high proportion of new energy has become a prominent feature of modern power systems. Due to the intermittency, volatility, and strong randomness in wind power generation, an accurate and reliable method for the prediction of wind power is required. This paper proposes a modified stacking ensemble learning method for short-term wind power predictions to reduce error and improve the generalization performance of traditional single networks in tackling the randomness of wind power. Firstly, the base learners including tree-based models and neural networks are improved based on the Bagging and Boosting algorithms, and a method for determining internal parameters and iterations is provided. Secondly, the linear integration and stacking integration models are combined to obtain deterministic prediction results. Since the modified stacking meta learner can change the weight, it will enhance the strengths of the base learners and optimize the integration of the model prediction to fit the second layer prediction, compared to traditional linear integration models. Finally, a numerical experiment showed that the modified stacking ensemble model had a decrease in MAPE from about 8.3% to 7.5% (an absolute decrease of 0.8%) compared to a single learner for the 15 min look-ahead tests. Changing variables such as the season and predicting the look-ahead time showed satisfactory improvement effects under all the evaluation criteria, and the superiority of the modified stacking ensemble learning method proposed in this paper regarding short-term wind power prediction performance was validated.

**Keywords:** wind power prediction; stacking; ensemble learning; bagging and boosting algorithms; fusion models



**Citation:** Yang, Y.; Li, Y.; Cheng, L.; Yang, S. Short-Term Wind Power Prediction Based on a Modified Stacking Ensemble Learning Algorithm. *Sustainability* **2024**, *16*, 5960. <https://doi.org/10.3390/su16145960>

Academic Editor: Mohamed A. Mohamed

Received: 15 April 2024

Revised: 19 June 2024

Accepted: 19 June 2024

Published: 12 July 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Wind power, as one of the oldest and most important energy sources on Earth today, has played a predominant role in the development of renewable energies due to its advantages such as huge resource reserves, renewability, no environmental pollution, wide distribution, short infrastructure cycle, and low installation, operation, and maintenance costs. For example, the development of wind power in China has advanced in both total quantity and growth rate. Nevertheless, wind power abandonment is still prominent [1]. Due to the intermittency, randomness, and volatility of wind speeds [2], wind power has low controllability and schedulability, bringing operational challenges to power systems. If wind power can be predicted to a certain extent in the sense of reducing uncertainty, wind power can better leverage its advantages of stability and cleanliness. In terms of safety, it can reduce power grid fluctuations and power grid losses. In terms of adaptability, it is convenient to adjust the scheduling plan and stabilize the operation control. In terms of economy, it can reduce the operating costs of the power system, promote new energy consumption, implement power transmission, and save power, which are of great economic significance [3].

The mathematical models used in wind power predictions are mainly artificial-intelligence-based methods, such as artificial neural networks (ANN) [4] and support vector machines (SVM) [5]. A series of ensemble learning methods, including Bagging and Boosting variants, have been introduced to improve the regression ability of deep

belief networks (DBN) and are applied to low-voltage load point predictions with a strong ability to approximate nonlinear mapping [6]. Artificial Neural Networks (ANNs), Long Short-Term Memory (LSTM), Gate Recurrent Units (GRUs), and eXtreme Gradient Boosting (XGboost) have been used to predict deterministic and probabilistic wind power generation and compare their performance using numerical weather predictions [7]. A novel adaptive ensemble data driven (AEDD) approach has been proposed for nonparametric probabilistic forecasting of electricity load by mining the uncertainty distribution from historical observations [8]. A backpropagation (BP) neural network with a sparrow search algorithm was proposed to improve the accuracy of wind power predictions [9]. Linear regression has been used as the meta learner of a second layer to construct a photovoltaic power prediction model with multiple stacking models embedded in machine learning algorithms [10].

The issue in existing wind power prediction models is that a single learner and traditional ensemble learning have a more limited role in wind power prediction error elimination. Due to the greater uncertainties of wind power versus load, more complex ensemble learning is needed to reduce errors and enhance the generalization ability. A stacking algorithm—which learns the model in two layers, saving the output of the first layer’s base learner as the input of the second layer’s meta learner, and training again to find the ensemble learning and deeper logic inside the wind power—can solve this problem to some extent [11].

A traditional stacking algorithm usually uses Random Forest, XGBoost, LightGBM, three tree-based learners, in the first layer; this method has been used in predictions of, e.g., the state of charge of a battery [12] and photovoltaic power predictions [10]. However, the tree-based model is a simplified version of a neural network, which can memorize more complex and probability-based rules. Previous studies have proven that the performance of neural networks can be better than that of the tree-based algorithm in both classification problems [13] and regression problems at the expense of more than 10 times the training time being used [14].

Therefore, the main contributions in this paper include the following:

- Five suitable base models were chosen for the following research, including two boosting algorithms and methods to determine internal parameters.
- A stacking algorithm with both tree-based models and neural networks combines the advantages of the two, both of which have been proposed for wind power predictions and to reduce errors therein.
- In addition, a modified stacking algorithm is proposed to overcome deficiencies, such that all outputs of the first layer have equal weights.

In summary, all the improvements make the errors of the proposed short-term wind power predictions smaller and meet the relevant national standards.

## 2. Model Evaluation Criteria

In the regression problem of machine learning, the most commonly used evaluation metrics are the mean absolute error (MAE), the root mean square error (RMSE), and the coefficient of determination ( $R^2$ ). In order to facilitate a comparison of the performance of the models with different numbers of samples, the normalized percentage metrics are used as much as possible for measuring. In view of this, this paper uses the mean absolute percentage error (MAPE), the normalized root means square error (NRMSE), and the coefficient of determination. In the following description,  $y(x_i)$  is the true value of the sample output corresponding to the  $i$ th input,  $\bar{y}$  is the average of the true value of the sample output,  $\hat{y}(x_i)$  is the predicted value of the sample output corresponding to the  $i$ th input,  $N$  is the number of samples,  $y_{\max}$  and  $y_{\min}$  are the maximum value and minimum value of the sample output.

The mean absolute error is the arithmetic mean of the absolute error between the actual value and the predicted value, reflecting the deviation between the predicted value

and the actual value, which is normalized to MAPE. The MAPE takes a value in the range of [0, 1]; the smaller this value, the better performance. Moreover,

$$MAPE = \frac{\frac{1}{N} \sum_{i=1}^N |y(x_i) - \hat{y}(x_i)|}{y_{\max} - y_{\min}} \times 100\%. \quad (1)$$

The root mean square error is the root of the mean of the squared sum of the difference between the predicted and actual values, that is, the standard deviation between the predicted and actual values, reflecting the degree of dispersion between the predicted and actual values. It is NRMSE after normalization whose value is in the range of [0, 1]; the smaller this value, the lower the variance, meaning better performance. Moreover,

$$NRMSE = \frac{\sqrt{\frac{1}{N} \sum_{i=1}^N (y(x_i) - \hat{y}(x_i))^2}}{y_{\max} - y_{\min}} \times 100\%. \quad (2)$$

The coefficient of determination is the ratio of the regression square and the total square in the regression; this reflects the regression equation for the predicted value of the fit of the metrics. It takes a value in the range of [0, 1]; the closer to 1, the better the fit. Moreover,

$$R^2 = 1 - \frac{\sum_{i=1}^N (y(x_i) - \hat{y}(x_i))^2}{\sum_{i=1}^N (y(x_i) - \bar{y})^2}. \quad (3)$$

### 3. A Modified Stacking Ensemble Learning Algorithm

Ensemble learning accomplishes learning tasks by building and combining multiple learners. The general approach is to first generate a set of base learners and then combine the results of multiple base learners with some strategy through ensemble learning, which often leads to a superior result over a single learner and becomes a strong learner. If the base learners are the same types of algorithms, the integration is then called homogeneous; otherwise, it is called heterogeneous. In terms of both a certain level of accuracy and diversity, the integration error rate decreases as the number of learners in the inheritance increases, and the integration result will theoretically be better than the optimal result in the base learner [15]. Common ensemble learning methods include Bagging, Boosting, and Stacking.

#### 3.1. Bagging

Bagging, also known as Bootstrap aggregation, comprises parallel ensemble learning, where individual learners do not have strong dependencies on each other and can be generated simultaneously [16]. Randomly selected  $m$  data in original dataset  $M$  are trained to generate an input space to output space learner. Sampling is repeated  $S$  times, and a total of  $S$  new datasets are selected to train  $S$  learners, respectively. Bagging is sampling with put-back; the number of samples in each subset must be the same as the original number of samples, while duplicates are allowed. Finally, the results of these  $S$  learners are averaged using a simple voting method.

#### 3.2. Boosting

Boosting is an algorithm that can boost a weak learner into a strong one; it is an example of the sequential ensemble learning. There are strong dependencies between individual learners that must be generated serially. Its general method is the same as that of Bagging, but with two main improvements [17].

Firstly, each training sample in the original dataset has a different selection probability. Bagging takes equal probability sampling, and the  $S$  samples are completely independent

of each other. In boosting, only in the first sampling instant, an equal probability random sampling is used; after that, according to the learning results of each instant, the probability of each sample being selected into the new sample set is adjusted by the re-weighting method, and each round needs to detect whether the currently generated learner meets the basic standards of accuracy. In case of unsatisfactory results, the current learner is discarded, and the learning process stops.

Secondly, the combination methods of the results of the individual bases are more diverse. Boosting algorithms combine the results of each base model more rationally, applying methods including weighted averaging, median selection of the results of the error function, and matrix calculation instead of simple averaging (as in bagging).

These two changes make boosting pay more attention to the samples that are difficult to learn, increasing the training accuracy of the samples with large errors and reducing the calculation of the samples that are easy to identify, which ultimately improves the accuracy of the model. In this paper, BEM (Basic Ensemble Method) Boosting and AdaBoost.RT (Adaptive Boosting) are used to construct the base learner.

### 3.2.1. Procedure of BEM Boosting

The training of BEM Boosting [6] includes the following six steps.

1. Initialize sample probabilities in original data set  $D_O$  so that their initial selection probabilities are equal and the probability of the  $i$ th sample is determined as:

$$p_i^0 = 1/N, i = 1, 2, \dots, N, \quad (4)$$

where  $N$  is the number of the datum in the sample set.

2. Compose a new input space based on the above probability with putative sampling, and use the corresponding output space as the target to train the learner.
3. From the simulated raw data of the above learner, calculate absolute error AE between the simulated and actual values:

$$AE_i = |y(x_i) - \hat{y}(x_i)|, i = 1, 2, \dots, N. \quad (5)$$

4. Split the data into two categories of large and small errors based on the absolute error, with the split point set to an adjustable  $\varphi$ . Count the data with errors larger than the split point and calculate their errors  $\varepsilon$  for subsequent integrations:

$$N_{upper} = \sum_{i=1}^N 1\{AE_i > \varphi\}, \quad (6)$$

$$\varepsilon = \left\{ \sum AE_i \mid AE_i > \varphi \right\}. \quad (7)$$

5. Since boosting pays more attention to the samples that are difficult to learn and have large errors, the selection probability of the samples with large errors is increased, while the selection probability of the samples with small errors is decreased. The  $m$ th probability is updated:

$$p_i^{m+1} = \frac{p_i^m}{Z} \times \begin{cases} \frac{N}{N_{upper}}, & AE_i > \varphi \\ \frac{N_{upper}}{N}, & AE_i \leq \varphi \end{cases}, i = 1, 2, \dots, N, \quad (8)$$

where normalization constant  $Z = \sum p_i^{m+1}$ .

6. Repeat steps 2–5 several times, with the first  $T$  times serving to meet the accuracy of the basic conditions of the combination of learners and the remaining iterations being implemented because the beginning of the  $T + 1$  times learners are all abandoned.

Here,  $T$  is the final number of iterations. Combine the initial learner and these  $T$  learners in the following way:

$$\hat{y}_{Ensemble} = \frac{\sum_{t=1}^T \log(\varepsilon_t^{-1}) \times \hat{y}^{(t)}}{\sum_{t=1}^T \log(\varepsilon_t^{-1})}. \quad (9)$$

### 3.2.2. Procedure of AdaBoost.RT

AdaBoost.RT is similar to BEM Boosting with only a slight difference in the classification error definition, i.e., the average relative error is used to update the mean absolute error in Equation (5):

$$ARE_i = \frac{|y(x_i) - \hat{y}(x_i)|}{y(x_i)} \times 100\%, i = 1, 2, \dots, N. \quad (10)$$

The subsequent split point  $\varphi$  should also correspond with the mean relative error percentage; the rest is the same as with BEM Boosting.

### 3.3. Generate the Base Learner

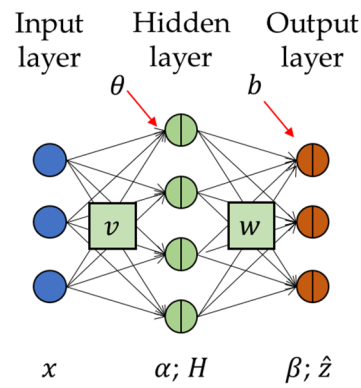
Ensemble learning uses some learners as base learners in order to get better results; it can have a strong nonlinear fitting ability in wind power predictions. Ensemble learning mainly consists of bagging and boosting algorithms, of which Random Forest (RF) is the most widely used of the bagging algorithms. XGBoost and LightGBM are boosting algorithms which show better performance in terms of dealing with the overfitting runs and time effects. Therefore, five models, i.e., RF, XGBoost, LightGBM, BP BEM Boosting, and BP AdaBoost.RT are combined as base learners in the stacking algorithm.

Random Forest is a classical integrated learning model based on bagging; it further introduces a random attribute selection in the training of decision trees [18]. The randomness of each training is ensured by two alterations, data random sampling and feature random sampling, to construct different prediction trees with both sample perturbation and attribute perturbation. Ultimately, the prediction results are obtained by voting or averaging, which further improves the generalization performance of the ensemble learning with an increase in the degree of variances.

XGBoost (eXtreme Gradient Boosting) is a boosting algorithm, which is an improvement of the Gradient Boosting Decision Tree algorithm (GBDT); it uses Newton's method when solving the extreme value of the loss function [19]. Expanding the Taylor expansion of the loss function to the second order and adding a regularization term in the loss function, it considers the accuracy and complexity of the model. It has a parallel operation structure with the advantages of being easy to implement and achieving high accuracy. It is one of the most commonly used decision tree models.

LightGBM (Light Gradient Boosting Machine) is a kind of gradient boosting framework. Essentially, it is still a decision tree, used as a weak learner model [20]. LightGBM has the advantages of efficient parallel computing, faster training speed, and lower memory use. It is suitable for dealing with large-scale data, and as such, it has a wide range of applications.

Backpropagation (BP) neural networks are multilayer networks and the most successful neural network learning algorithms to date [15]. Without loss of generality, they are widely used in ensemble learning. A BP network is structured into three layers: the input layer, hidden layer, and output layer. The hidden layer and output layer are used to operate on the data, while the output layer outputs the results. The training is divided into two processes, forward propagation and back propagation, as shown in Figure 1.



**Figure 1.** Schematic diagram of BP.

The forward propagation process allows the data to enter the network and directly picks the values of the neurons in each layer multiplied by the corresponding weights + bias variables as inputs, which then go to activation function Sigmoid to produce the output. Let the input be  $x$  and the predicted target output be  $z$ .

The output of the hidden layer is:

$$H_h = f(\alpha_h) = f\left(\sum_{i=1}^d v_{ih}x_i - \theta_h\right), \quad (11)$$

The output of the output layer is:

$$\hat{z}_j = f(\beta_j) = f\left(\sum_{h=1}^q w_{hj}H_h - b_j\right), \quad (12)$$

where  $d$ ,  $q$ , and  $l$  are the numbers of neurons in the input, hidden, and output layers, respectively;  $v_{ih}$  is the weight between the  $i$ th neuron in the input layer to the  $h$ th neuron in the hidden layer;  $w_{hj}$  is the weight between  $h$ th neuron in the hidden layer to the  $j$ th neuron in output layer;  $\theta_h$ ,  $b_j$  are the  $h$ th bias of the hidden layer and the  $j$ th bias of the output layer, respectively; and  $\alpha_h$ ,  $\beta_j$  are the  $h$ th input of the hidden layer and the  $j$ th input of the output layer;  $f$  is the activation function.

The backpropagation process uses the gradient descent method and the objective function is the sum of error squares:

$$(v, w, \theta, b) = \underset{(v, w, \theta, b)}{\operatorname{argmin}} E = \underset{(v, w, \theta, b)}{\operatorname{argmin}} \frac{1}{2} \sum_{j=1}^l (\hat{z}_j - z_j)^2. \quad (13)$$

BP network training results are obtained after updating the parameters. Since the network has good nonlinear mapping ability and strong migration ability, it can be used as the base network of a boosting algorithm to enhance wind power predictions. BP BEM Boosting and BP AdaBoost.RT are algorithms that use a BP network and enhance its performance with two boosting methods. The above five learners are heterogeneous learners, with tree and multi-layer network structures, including ensemble learning of bagging and boosting theories. They provide respective prediction advantages to complement each other as the base learners of wind power predictions.

### 3.4. Linear Weighted Model Integration

Due to the uncertainty and stochasticity of wind power, a single base learner needs to be improved. Also, it is necessary to integrate the base learner with ensemble learning to improve the model's immunity to interference and generalization performance, so as to enable it to cope with more situations. Tuning the parameters of the three networks, i.e., RF, XGBoost, and LightGBM, as well as two kinds of improvements to the BP network, i.e.,

BP BEM Boosting and BP AdaBoost.RT, yields the basic requirements for predictions and is used with base learners to construct an ensemble learning integrated model.

The average and stacking methods were chosen for model integrations in this paper [13]. Five different kinds of average integrated models are shown below. In Equations (14)–(18),  $n$  represents the number of base learners.

### 7. Equal Weighted Integrated Model

The equal weight integrated model is the simplest, taking the arithmetic mean directly as the final result:

$$\hat{y}_{Liner\_Weighted1} = \frac{1}{n} \sum_{i=1}^n \hat{y}_i, \quad (14)$$

### 8. Weighted Integrated Model

Compared to the equal weight integrated model, the weighted integrated model considers the superiority or inferiority of the prediction effect of each base learner, and the weight coefficients are only related to the model prediction value:

$$\hat{y}_{Liner\_Weighted2} = \sum_{i=1}^n \omega_i \hat{y}_i, \omega_i = \frac{\sum_{j=1}^m \hat{y}_i(x_j)}{\sum_{j=1}^m \sum_{i=1}^n \hat{y}_i(x_j)}, \quad (15)$$

### 9. Sum of Squared Error Weighted Integrated Model

Similar to the above except that the weights are assigned differently, and the inverse of the sum of squared errors is used for weight calculations:

$$\hat{y}_{Liner\_Weighted3} = \sum_{i=1}^n \omega_i \hat{y}_i, \omega_i = \frac{1/SSE_i}{\sum_{i=1}^n (1/SSE_i)} = \frac{1/\sum_{j=1}^m (\hat{y}_i(x_j) - y_i(x_j))^2}{\sum_{i=1}^n \left[ 1/\sum_{j=1}^m (\hat{y}_i(x_j) - y_i(x_j))^2 \right]}, \quad (16)$$

### 10. Root Mean Squared Error Weighted Integrated Model

The inverse of the (standardized) root mean square error is used for weighting calculations:

$$\hat{y}_{Liner\_Weighted4} = \sum_{i=1}^n \omega_i \hat{y}_i, \omega_i = \frac{1/NRMSE_i}{\sum_{i=1}^n (1/NRMSE_i)}, \quad (17)$$

### 11. Coefficient of Determination Weighted Integrated Model

Weights are calculated using the coefficient of determination:

$$\hat{y}_{Liner\_Weighted5} = \sum_{i=1}^n \omega_i \hat{y}_i, \omega_i = \frac{(R^2)_i}{\sum_{i=1}^n (R^2)_i}. \quad (18)$$

The combination logic of the above five models is the same, using a linear combination of the base learner prediction results and the corresponding weights. These weights have a high percentage of learner predictions with small errors and high accuracy, except that the reference standards for measuring the error in the different weighted models are different. All the weights sum to 1 to guarantee the integration results.

### 3.5. Stacking

Stacking, like bagging and boosting, is a typical algorithm for ensemble learning to integrate multiple models [21]. While bagging focuses on reducing the variance of the

model and boosting focuses on reducing the bias of the model, stacking is able to reduce the bias and variance of the model in a comprehensive way. Stacking has a more complex contracture and improves the performance of the model.

Stacking is the process of constructing a new input variable from the predictions of several models and using it to train a new model to obtain the final output. It is generally divided into two layers. The single learner used for the first layer of training is called the base learner (level-0 model), and the inductive and integrated learner used for the second layer of training is called the meta-learner (level-1 model). Several base models are trained and used to predict the results based on the inputs and outputs of the original data from the first level hypothesis space. This step has already been done in Section 3.4. and we assume the output values are  $\hat{y}_1, \hat{y}_2, \hat{y}_3, \hat{y}_4, \hat{y}_5$ . The prediction is done using a K-fold cross-validation method [22], with  $k$  taken as 5, which effectively prevents overfitting. By arranging the outputs of these base learners as a new input vector (matrix), the outputs are still the corresponding real outputs, while the training meta learner is constructed as a new hypothesis space:

$$h_{Stacking} : v(x_i) \rightarrow y(x_i), v(x_i) = \{\hat{y}_1(x_i), \hat{y}_2(x_i), \hat{y}_3(x_i), \hat{y}_4(x_i), \hat{y}_5(x_i)\}. \quad (19)$$

Then, training the meta learner will yield the final stacking prediction.

Since the base learners have already gone through one round of predictions, the output should be closer to the true value, although still biased. Each dimension of the meta-learner's input variables is close to the output, and the second layer of training brings these predictions closer to the real outputs. This is how stacking can reduce model errors and increase stability.

Modified Stacking: The weight of each dimension in input vector  $v$  is the same, but the performance of each base learner is different in the actual prediction, so it is desirable to increase the weight of the base learner with better performance in the second layer and decrease the weight with poor performance. Therefore, the new hypothesis space for modified stacking is:

$$h_{Modified\_Stacking} : v'(x_i) \rightarrow y(x_i), v'(x_i) = \{\omega_1 \hat{y}_1(x_i), \omega_2 \hat{y}_2(x_i), \omega_3 \hat{y}_3(x_i), \omega_4 \hat{y}_4(x_i), \omega_5 \hat{y}_5(x_i)\} \quad (20)$$

where  $\omega_i$  is the weight corresponding to the best performing model in the linear weighted fusion model above. The process is exactly the same as with stacking. A flow chart is shown in Figure 2.

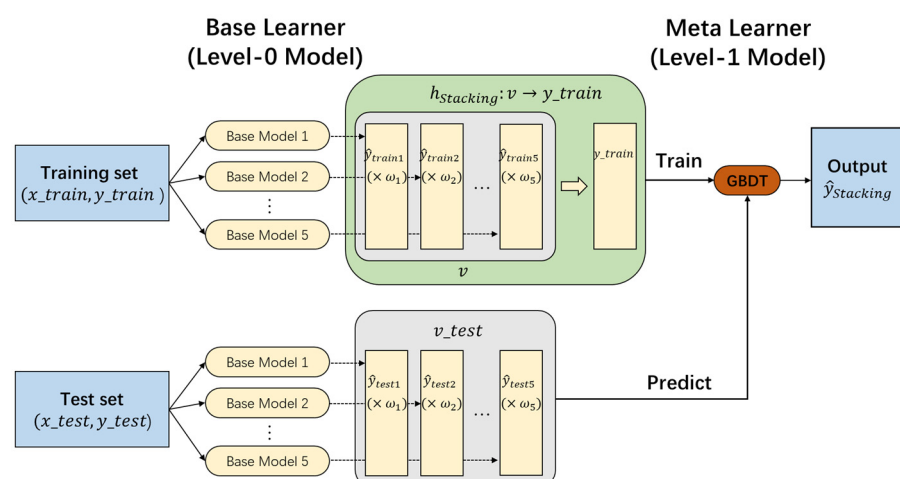


Figure 2. Flow chart of (modified) stacking.

Stacking has the following considerations for model selection: For base learners, heterogeneous learners with good performance and large differences in algorithmic principles should be elected, so that each learner can benefit from its respective advantages for dif-



ferent inputs and outputs, and then integrate them with the idea of ensemble learning to complement each other's strengths. For meta learners, it should be stable and not duplicated with that of the base learner, so as to avoid the same data passing through the base learner and meta learner, resulting in a large number of overfittings. The five base learners have met the above requirements, and the meta learner can use a simple Gradient Boosting Decision Tree algorithm (GBDT), which has a fast running time and stable results.

## 4. Case Study

### 4.1. Data Collection

In order to verify the performance of the improved stacking model proposed in this paper, actual wind power NWP data from a wind farm in China for one year were used as a case study. The data included wind speed and direction at 10 m, 30 m, and 50 m of the wind tower, the temperature and humidity of the wind farm at different moments, as well as the corresponding wind power. The data were measured from 1 January to 31 December with a time interval of 15 min. We divided this period into four quarters in our analysis. The first two months of each quarter were used for training and the first 20 days of the latter month were used for testing, i.e., to make sure the ratio of the training and testing sets was 3:1. Below, if not otherwise specified, the wind power was taken as the predicted output, and the rest of the physical quantities were taken as the predicted inputs, so that the inputs of a certain moment could be used to predict the wind power of the next moment (i.e., 15 minutes later). MAPE was used as the primary metric of model performance, while NRMSE and  $R^2$  were used as the secondary reference indicators. All models were implemented in Jupyter Notebook under the Python framework.

### 4.2. Base Learner

Determining the internal parameters of the base learner provides a significant improvement in its performance. Without loss of generality, the test was performed using the 15-min ahead numerical example for the second quarter data from this wind farm. Random Forest, XGBoost, and LightGBM can all be trained with the Grid Search algorithm [23]. The number of iterations and internal parameters of BP BEM Boosting and BP AdaBoost.RT need to be calculated manually. The following will take BP BEM Boosting as an example to illustrate the training of the boosting algorithm.

Figure 3 plots the MAPE of the BP BEM Boosting model as a function of the number of iterations. It can be seen that the MAPE decreases significantly at the beginning, reaches its lowest at iteration 6, and remains essentially flat thereafter. If the result of one iteration is smaller than that of the following two, the iteration terminates, i.e.,

$$\begin{aligned} \text{if } : \text{MAPE}(i) < \text{MAPE}(i+1) \& \text{MAPE}(i) < \text{MAPE}(i+2) \\ \text{Iteration} &= \min\{i\} \end{aligned} \quad (21)$$

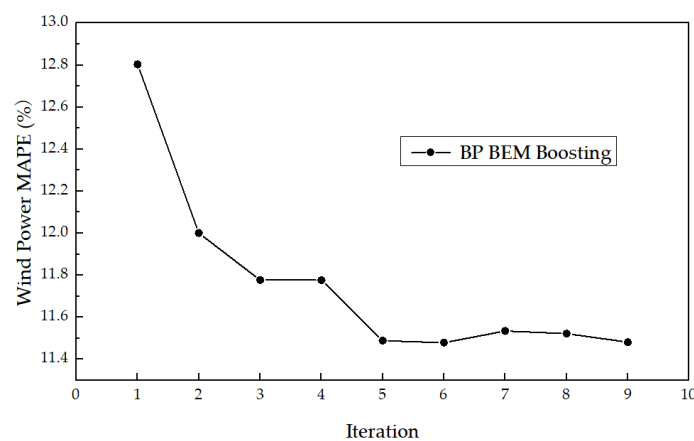
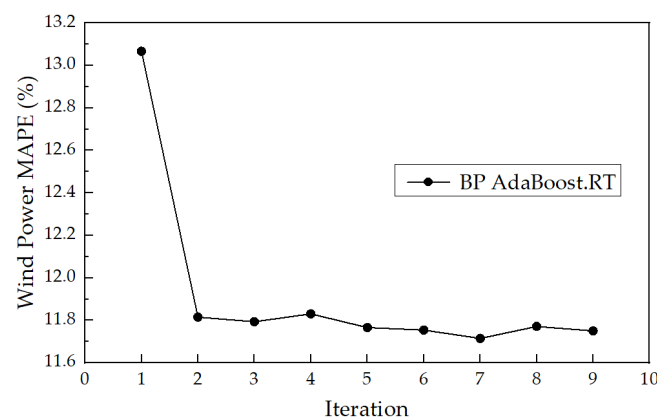


Figure 3. MAPE of different iterations in BP BEM Boosting.

As a result, the number of iterations for the subsequent integration was selected as 6. The optimal result (at the sixth iteration) and the original BP model (at the first iteration) were compared, and the error decreased by about 4.35%.

This selection also suggests that the seventh iteration, after increasing the chosen possibility of data with high prediction errors, led to a greater overall error in the results. Weak learners were those that did not meet the ensemble learning requirements, i.e., predicting correctly at a rate higher than 50%. Mixing in a poor learner in an otherwise better set of learners obviously reduces the model's performance.

Similarly, the same operation was applied to BP AdaBoost.RT and the optimal model was obtained when the number of iterations was 7. Figure 4 shows an image of MAPE of different iterations in BP AdaBoost.RT.



**Figure 4.** MAPE of different iterations in BP AdaBoost.RT.

The internal parameter  $\varphi$  is independent of iterations, which decides the dividing of different samples. Through extensive experimental testing,  $\varphi = 0.4$  was taken for BP BEM Boosting and  $\varphi = 0.8$  was taken for BP AdaBoost.RT to achieve the best performance, respectively.

The results of these five base models are compared in Table 1.

**Table 1.** Performance Comparison of Base Models.

Model	MAPE	NRMSE	R <sup>2</sup>
RF	8.303%	11.659%	0.764211072
XGBoost	8.316%	11.697%	0.762671824
LightGBM	8.315%	11.565%	0.768005054
BP BEM Boosting	8.315%	12.513%	0.728408502
BP AdaBoost.RT	8.613%	12.172%	0.743009887

To make sure the predicted wind power was closer to the real wind power, we chose MAPE as the main criterion and NRMSE and R<sup>2</sup> as secondary criteria. From Table 1, it can be seen that within the five base learners, Random Forest was the best and BP AdaBoost.RT was the worst. Since the basic requirements of heterogeneity and good performance of the base learners were satisfied, they could be used for the construction of the subsequent linear weighted integration model and stacking integration model.

#### 4.3. Meta Learner (Integration Model)

Both the Linear Weighted Integration Model and the (Modified) Stacking Integration Model were trained and tested according to the process described above. The main test variables were different seasons (quarters) and prediction look-ahead time, and the test contained a training set and test set performance metrics [24].

#### 4.3.1. Wind Power Prediction Results at Different Times

The training set used 60 days (about 2 months) of data with 5760 points, while the test set used 20 days of data with 1920 points. Deterministic wind power predictions were done in four quarters.

1. The first quarter deterministic prediction. In this phase, the historical NWP data from January to February were used, and the first 20 days of March were tested.
2. The second quarter deterministic prediction. In this phase, the historical NWP data from April to May were used, and the first 20 days of June were tested.
3. The third quarter deterministic prediction. In this phase, the historical NWP data from July to August were used, and the first 20 days of September were tested.
4. The last quarter deterministic prediction. In this phase, the historical NWP data from October to November were used, and the first 20 days of December were tested.

Table 2 shows a performance comparison of deterministic predictions in different seasons, including the training dataset and test dataset. Comparing the training and test sets of the same quarter and model, it was clearly established that the test set error was generally larger than that of the training set.

**Table 2.** Performance Comparison of Deterministic Predictions in Different Seasons.

Season	Model	Training Dataset Metrics			Test Dataset Metrics		
		MAPE	NRMSE	R <sup>2</sup>	MAPE	NRMSE	R <sup>2</sup>
1st Quarter	RF	3.720%	7.180%	0.943799246	6.824%	10.692%	0.876278386
	XGBoost	3.542%	6.600%	0.952517148	6.958%	10.970%	0.869752545
	LightGBM	4.175%	7.601%	0.937008462	6.905%	10.715%	0.875734685
	BP BEM Boosting	5.158%	11.107%	0.865503138	6.976%	10.715%	0.875731071
	BP AdaBoost.RT	5.154%	11.103%	0.865595447	7.500%	11.963%	0.845121147
	Linear Weighted 1	4.063%	7.889%	0.932155762	6.335%	9.996%	0.891858533
	Linear Weighted 2	4.066%	7.895%	0.932040174	6.336%	9.999%	0.891790414
	Linear Weighted 3	3.869%	7.299%	0.941922394	6.489%	10.235%	0.886613431
	Linear Weighted 4	3.952%	7.543%	0.937969772	6.399%	10.100%	0.889587824
	Linear Weighted 5	4.040%	7.814%	0.933430257	6.344%	10.012%	0.891509564
	Stacking	2.831%	5.459%	0.967507691	6.017%	9.598%	0.904283235
Modified Stacking	2.854%	5.450%	0.967612491	5.967%	9.503%	0.906589526	
2nd Quarter	RF	7.944%	11.847%	0.822613381	8.303%	11.659%	0.764211072
	XGBoost	8.138%	11.937%	0.819909272	8.316%	11.697%	0.762671824
	LightGBM	8.656%	12.541%	0.801238329	8.315%	11.565%	0.768005054
	BP BEM Boosting	10.628%	17.103%	0.630305072	8.315%	12.513%	0.728408502
	BP AdaBoost.RT	11.038%	17.208%	0.625762387	8.613%	12.172%	0.743009887
	Linear Weighted 1	8.402%	12.638%	0.798147646	7.840%	11.128%	0.785186501
	Linear Weighted 2	8.463%	12.752%	0.794477499	7.829%	11.124%	0.785359731
	Linear Weighted 3	8.146%	12.129%	0.814089328	7.946%	11.207%	0.782117116
	Linear Weighted 4	8.241%	12.330%	0.807860135	7.889%	11.155%	0.784130852
	Linear Weighted 5	8.278%	12.399%	0.805698832	7.876%	11.146%	0.784506212
	Stacking	6.833%	10.224%	0.867889374	7.559%	10.656%	0.794307533
Modified Stacking	6.798%	10.212%	0.868206889	7.494%	10.581%	0.797350030	
3rd Quarter	RF	7.944%	8.786%	0.870844715	9.576%	13.041%	0.776532352
	XGBoost	8.138%	8.886%	0.867889090	9.682%	13.236%	0.769792381
	LightGBM	8.656%	9.413%	0.851758807	9.677%	13.038%	0.776647279
	BP BEM Boosting	10.628%	9.982%	0.833305919	9.761%	13.447%	0.762397117
	BP AdaBoost.RT	11.038%	10.162%	0.827246353	9.793%	13.220%	0.770365891
	Linear Weighted 1	6.057%	8.929%	0.866629720	9.284%	12.573%	0.792304657
	Linear Weighted 2	6.059%	8.924%	0.866777689	9.285%	12.572%	0.792310802
	Linear Weighted 3	6.057%	8.892%	0.867721061	9.299%	12.599%	0.791421221
	Linear Weighted 4	6.056%	8.909%	0.867208690	9.290%	12.585%	0.791904049
	Linear Weighted 5	6.057%	8.922%	0.866838530	9.286%	12.577%	0.792169706
	Stacking	5.078%	7.400%	0.908396294	8.636%	11.502%	0.800468944
Modified Stacking	5.056%	7.400%	0.908379363	8.602%	11.428%	0.803092836	

Table 2. Cont.

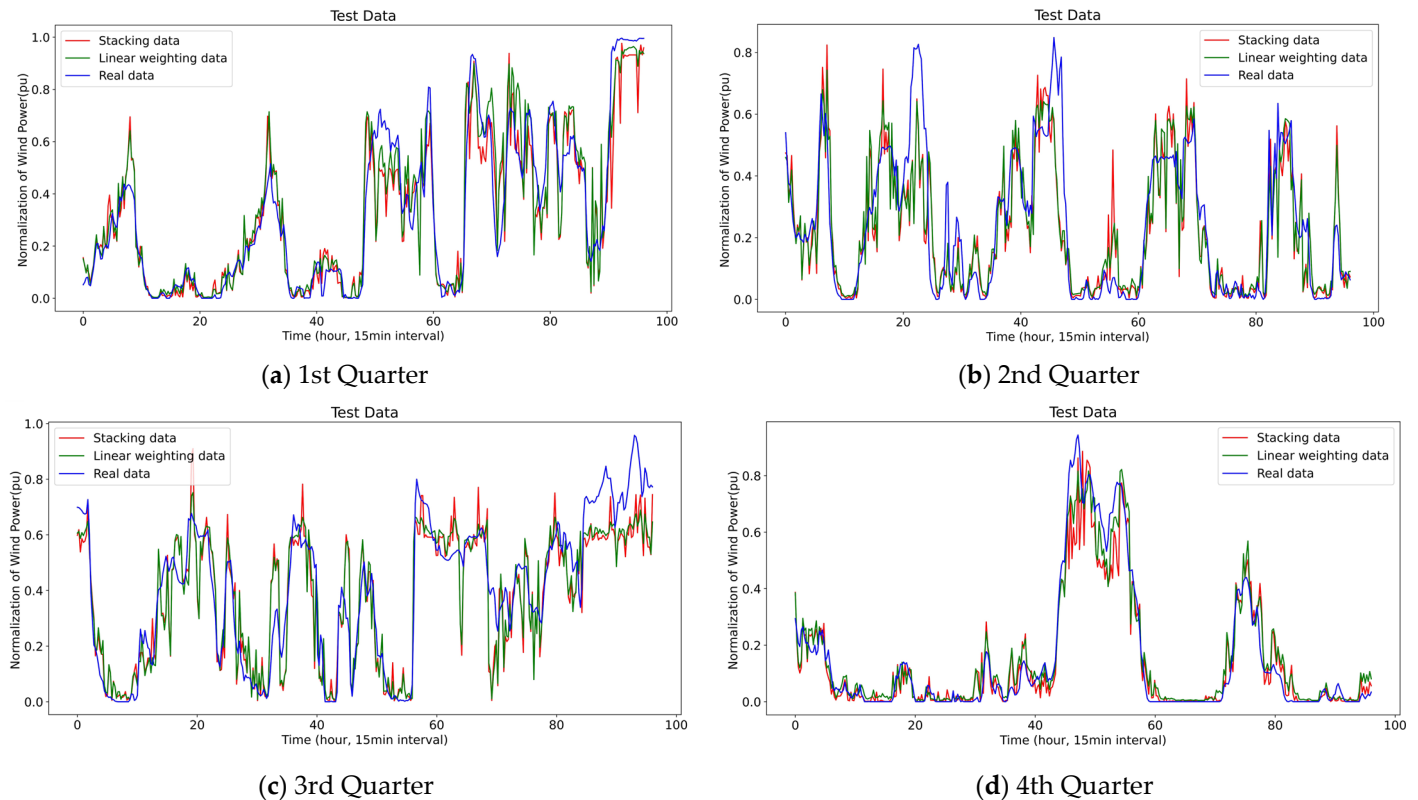
Season	Model	Training Dataset Metrics			Test Dataset Metrics		
		MAPE	NRMSE	R <sup>2</sup>	MAPE	NRMSE	R <sup>2</sup>
4th Quarter	RF	4.790%	8.149%	0.920345949	2.572%	4.775%	0.876127312
	XGBoost	4.852%	8.081%	0.921679310	2.601%	4.794%	0.875137442
	LightGBM	5.474%	8.908%	0.904827315	2.766%	4.595%	0.885309730
	BP BEM Boosting	4.938%	9.552%	0.890562007	2.220%	4.116%	0.907979616
	BP AdaBoost.RT	6.169%	10.887%	0.857850149	5.516%	7.910%	0.660125525
	Linear Weighted 1	4.997%	8.433%	0.914707161	2.902%	4.597%	0.885217680
	Linear Weighted 2	5.002%	8.446%	0.914435299	2.918%	4.610%	0.884567662
	Linear Weighted 3	4.935%	8.283%	0.917719618	2.757%	4.507%	0.889631651
	Linear Weighted 4	4.962%	8.349%	0.916384685	2.821%	4.542%	0.887931725
	Linear Weighted 5	4.988%	8.411%	0.915141855	2.880%	4.580%	0.886060534
	Stacking	3.684%	6.340%	0.951788957	2.455%	4.270%	0.900361683
	Modified Stacking	3.675%	6.335%	0.951867104	2.445%	4.218%	0.903223258

Without loss of generality, the following illustration still takes the second quarter of Table 2 as an example to compare the performance of different learners and the integration results in the same quarter. The five linearly weighted fusion models had an error of about 7.9% in the test set and were within 0.15% of each other, revealing that there was not much difference in the performance of these linear weighted models. Compared with the five base learners, the error of the MAPE decreased from about 8.3% to 7.9%, which is a relative decrease of 4.8% and an absolute decrease of 0.4%, and all the linear weighted integration models outperformed the base learners in terms of all three metrics. This fully illustrates the basic principle of ensemble learning, which can be achieved by combining weak learners to develop strong learners, allowing heterogeneous learners to function in the face of data points that they are good at, and other learners to average to reduce the errors at the points where their own predictions are poor. Since all learners yielded good predictions at most of the points and poor prediction at some of the points, the above averaging operation was able to complement the strengths of each procedure and improve the performance and generalization ability. In addition, Linear Weighted 2~Linear Weighted 5 were unequally weighted, which enabled learners with small errors to take up more weight and learners with large errors to reduce their weights, reducing the integration error even more. However, the five linear weighted models were constructed with similar ideas, and there was not much difference in terms of performance between the internals.

The ensemble learning stacking integration model yielded a decrease in MAPE from about 8.3% to 7.5% (an absolute decrease 0.8%) compared to the base learner, and a relative decrease of 9.0% and 9.8% for stacking and modified stacking. Similar to the analysis above, (modified) stacking reduced the model error more compared to simple linear weighting; all three metrics reflected this observation. In summary, the model performance for both the training and test sets may be ordered as follows: modified stacking > stacking > linear weighted > base learners. The reason that stacking outperformed linear weighting was that the second layer of the algorithm was trained to optimally combine the model predictions to form a new set of predictions, instead of simply assigning a fixed weight to each variable. The modified stacking model showed slightly improved performance compared to stacking, indicating that the idea was valid but could be improved [20,25–30].

From the four quarterly comparisons, it was observed that the errors in the 2nd and 3rd quarters were larger than those in 1st and 4th quarters, mainly due to the fact that the average values of the real wind power in the former were larger than that in the latter. The wind farm was characterized by a distinct monsoon climate, with high wind speeds in summer and low wind speeds in winter. The wind power often equaled 0 when the wind speed was low, thus reducing the uncertainty and making it easier to predict with smaller overall errors. This was also the reason for the anomalous test set error being less than the training set error in the 4th quarter, i.e., the real wind speed was too small in the sampled December test set.

The deterministic prediction results of 15-min-ahead on typical days in different seasons at this wind farm are plotted in Figure 5a–d. To facilitate the observation, a total of 384 points for only 4 days in each season are plotted with the optimal results of the stacking integration model, the optimal results in the linear weighted integration model, and the real data images, respectively.



**Figure 5.** Deterministic prediction results of integration models compared with real data with 15 min look-ahead time from four typical days (384 points) in different quarters: (a) 1st Quarter, (b) 2nd Quarter, (c) 3rd Quarter, (d) 4th Quarter.

#### 4.3.2. Wind Power Prediction Results of Different Look-Ahead Times

Without a loss of generality, for the purpose of comparing the performance selection of seasons with higher wind speeds, the 2nd quarter data were used to make forecasts of 15-min-ahead, 30-min-ahead, and 1-h-ahead. The 15-min-ahead forecasts are given in the 2nd Quarter part of Table 2; the other results are shown in Table 3.

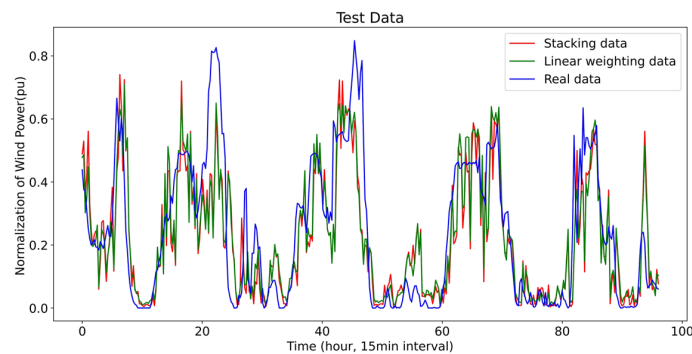
As the prediction look-ahead time increased, the error increased for all models, due to the overall increase in uncertainty caused by the physical boost of the prediction look-ahead time. Also, comparing modified stacking with the base learners, predicting 30 min and 1 h ahead relatively reduced the MAPE by about 9.3% and 8.6%, respectively (absolutely 0.83% and 0.94%) [7]. This shows that the model still worked when the base learners were slightly underperforming. The rest of the conclusions are basically the same as above, and a performance comparison still yielded the following order: modified stacking > stacking > linear weighted > base learners. It is worth noting that at 30 min and 1 h ahead predictions, the two BP Boosting models showed a significant increase in the error within the training set, and there was a disparity in the performance compared to the other base learners. This suggests that after increasing the prediction time further, the two BP learners may correctly predict less often than expected, and it may appear that the integration error of the five base learners is larger than the integration result of the three base learners.

The deterministic prediction results of 30 min and 1 h ahead on a typical day of the 2nd Quarter of this wind farm are plotted in Figure 6a,b. A total of 384 points from four

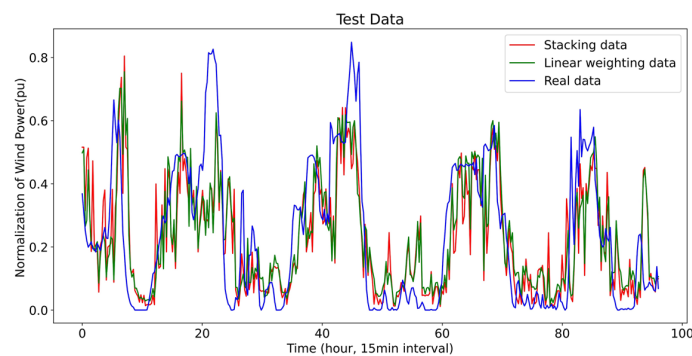
days were still taken to plot the optimal results of the modified stacking integration model, the optimal results in the linear weighted integration model, and the real data images, respectively. The deterministic prediction images for 15-min-ahead are given in Figure 5b.

**Table 3.** Performance Comparison of Deterministic Predictions with Different Look-ahead Times.

Look-Ahead Time	Model	Training Dataset Metrics			Test Dataset Metrics		
		MAPE	NRMSE	R <sup>2</sup>	MAPE	NRMSE	R <sup>2</sup>
15 min				See above			
30 min	RF	9.015%	13.065%	0.784251850	9.061%	12.615%	0.723847075
	XGBoost	9.212%	13.227%	0.778874960	9.050%	12.642%	0.722655063
	LightGBM	9.750%	13.833%	0.758151130	9.050%	12.482%	0.729659837
	BP BEM Boosting	11.359%	17.946%	0.592946634	8.994%	13.236%	0.696015216
	BP AdaBoost.RT	11.466%	17.760%	0.601351939	9.368%	13.372%	0.689747241
	Linear Weighted 1	9.354%	13.823%	0.758491032	8.613%	12.140%	0.744277930
	Linear Weighted 2	9.377%	13.868%	0.756911813	8.611%	12.139%	0.744325672
	Linear Weighted 3	9.203%	13.441%	0.771669886	8.682%	12.184%	0.742420939
	Linear Weighted 4	9.261%	13.601%	0.766189456	8.642%	12.153%	0.743712171
	Linear Weighted 5	9.270%	13.623%	0.765442534	8.639%	12.150%	0.743822392
1 h	Stacking	7.744%	11.329%	0.837794039	8.192%	11.724%	0.759075358
	Modified Stacking	7.724%	11.324%	0.837933722	8.160%	11.710%	0.759707105
	RF	11.269%	15.683%	0.689119575	11.186%	15.229%	0.597820244
	XGBoost	11.587%	16.018%	0.675682783	11.009%	15.096%	0.604793105
	LightGBM	12.111%	16.596%	0.651864864	11.142%	15.081%	0.605570133
	BP BEM Boosting	13.507%	20.974%	0.443999575	10.993%	16.290%	0.539794949
	BP AdaBoost.RT	14.875%	21.766%	0.401172809	12.445%	17.327%	0.479350627
	Linear Weighted 1	11.720%	16.638%	0.650101009	10.595%	14.706%	0.624942026
	Linear Weighted 2	11.775%	16.732%	0.646156699	10.608%	14.716%	0.624432778
	Linear Weighted 3	11.542%	16.216%	0.667615956	10.660%	14.711%	0.624721131
Linear Weighted 4	11.613%	16.395%	0.660265527	10.619%	14.696%	0.625497369	
Linear Weighted 5	11.564%	16.279%	0.665040871	10.640%	14.704%	0.625083714	
Stacking	9.696%	13.640%	0.764829815	10.063%	14.222%	0.627542378	
Modified Stacking	9.647%	13.583%	0.766817397	10.053%	14.187%	0.629403895	



(a) 30 min look-ahead



(b) 1 h look-ahead

**Figure 6.** Deterministic prediction results of the integration models compared with real data of different look-ahead times for four typical days (384 points) in the 2nd Quarter: (a) 30 min look-ahead, (b) 1 h look-ahead (15 min look-ahead time is shown in Figure 5b).

For subsequent improvements, firstly, observing the performance of each base learner when increasing the prediction look-ahead time (preferably such that the difference between their errors is not too large) and removing clearly inferior models will be useful in fusion models. Secondly, continuing to enrich the type and number of base learners for similar models with different parameters could achieve better basic accuracy. The combination of the two above enhancements could reduce the prediction errors and improve the model's generalization performance.

## 5. Conclusions

Ensemble learning is an effective method to reduce the uncertainty and increase the reliability of wind power predictions. Combining bagging, boosting, and stacking in ensemble learning, this paper proposes an ensemble learning integration model based on modified stacking. The modified stacking ensemble model showed a decrease in MAPE from about 8.3% to 7.5% (absolute decrease 0.8%) compared to a single learner for 15 min look-ahead tests. Comparing modified stacking with the base learners, predicting 30 min and 1 h ahead relatively reduced the MAPE by about 9.3% and 8.6%, respectively (absolutely 0.83% and 0.94%). The idea of bagging and boosting was used to train the base learners for short-term wind power predictions of wind farms. After training, bagging and boosting were able to improve the heterogeneous learners, with good performance and large differences. Meta learners can optimize the results via different base learning on the basis of linear weighted models that take advantage of strengths and complement the weaknesses, which improved the model's performance in all three different metrics. Changing the season and prediction look-ahead time, this model could predict deterministic wind power at least for about 1 h-ahead time, still with the best reliability. These observations prove that this method combines the strengths of both tree-based learners and neural networks, and that it is more effective than normal integration strategies. Base learners can be subsequently changed and applied in modified stacking to cope with longer look-ahead time predictions.

According to China's national wind power standards, the prediction error should not exceed  $\pm 15\%$  of real wind power in short-term (1–7 days ahead) predictions. In this research, we are trying to narrow the gap between practical situations and this standard and to increase the reliability of predictions and raise standards.

**Author Contributions:** Conceptualization, methodology, software, validation, formal analysis, writing—original draft preparation Y.Y.; investigation, resources, data curation, writing—review and editing, visualization, L.C.; supervision, Y.L. and S.Y. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Data is unavailable due to privacy.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Nomenclature

Variables	Meaning
MAPE	Mean absolute percentage error
NRMSE	Normalized root mean square error
$R^2$	Coefficient of determination
$y(x_i)$	True value of wind power corresponding the $i$ th input
$\bar{y}$	Average of the true value of wind power
$\hat{y}(x_i)$	Predicted value of wind power corresponding the $i$ th input
$N$	Number of samples
$y_{\max}$	Maximum value of wind power

$y_{\min}$	Minimum value of wind power
$D_O$	Original data set
AE	Absolute error
$\varphi$	An adjustable parameter to split point set
$N_{upper}$	Number of data with errors larger than the split point
$\varepsilon$	Total errors
$p_i^m$	Probability of the $m$ th selection and the $i$ th item
$Z$	Normalization constant for probability
$T$	Final number of iterations
ARE	Average relative error
$H_h$	$h$ th output of the hidden layer
$\hat{z}_j$	Prediction of the $j$ th output of the output layer
$d$	Number of neurons in input layers
$q$	Number of neurons in hidden layers
$l$	Number of neurons in output layers
$v_{ih}$	Weight between the $i$ th neuron in the input layer to the $h$ th neuron in the hidden layer
$\theta_h$	$h$ th bias of the hidden layer
$w_{hj}$	Weight between $h$ th neuron in the hidden layer to the $j$ th neuron in output layer
$b_j$	$j$ th bias of the output layer
$\alpha_h$	$h$ th input of the hidden layer
$\beta_j$	$j$ th input of the output layer
$f$	Activation function
$\omega_i$	Weight coefficients of the $i$ th model
SSE	Sum of squared errors
$h_{Stacking}$	Hypothesis space for Stacking prediction
$h_{Modified\_Stacking}$	Hypothesis space for Modified Stacking prediction
$v(x_i)$	Input vector (matrix) of $i$ th input in Stacking prediction
$v'(x_i)$	Input vector (matrix) of $i$ th input in Modified Stacking prediction

## References

- China Electricity Council. Analysis and Forecast Report on the National Power Supply and Demand Situation in 2023–2024. Available online: <https://cec.org.cn/detail/index.html?3-330280> (accessed on 24 May 2024).
- Wan, C.; Cui, W.; Song, Y. Probabilistic Forecasting for Power Systems with Renewable Energy Sources: Basic Concepts and Mathematical Principles. *Proc. CSEE* **2021**, *41*, 6493–6509.
- Wan, C.; Lin, J.; Wang, J.; Song, Y.; Dong, Z.Y. Direct quantile regression for nonparametric probabilistic forecasting of wind power generation. *IEEE Trans. Power Syst.* **2016**, *32*, 2767–2778. [CrossRef]
- Park, D.C.; El-Sharkawi, M.; Marks, R.; Atlas, L.; Damborg, M. Electric load forecasting using an artificial neural network. *IEEE Trans. Power Syst.* **1991**, *6*, 442–449. [CrossRef]
- Fan, S.; Chen, L. Short-term load forecasting based on an adaptive hybrid method. *IEEE Trans. Power Syst.* **2006**, *21*, 392–401. [CrossRef]
- Cao, Z.; Wan, C.; Zhang, Z.; Li, F.; Song, Y. Hybrid Ensemble Deep Learning for Deterministic and Probabilistic Low-Voltage Load Forecasting. *IEEE Trans. Power Syst.* **2020**, *35*, 1881–1897. [CrossRef]
- Wu, Y.-K.; Wu, S.-H.; Huang, C.-L.; Hong, J.-S.; Chang, H.-L. Deterministic and Probabilistic Wind Power Forecasts by Considering Various Atmospheric Models and Feature Engineering Approaches. In Proceedings of the 2022 IEEE/IAS 58th Industrial and Commercial Power Systems Technical Conference (I&CPS), Las Vegas, NV, USA, 2–5 May 2022; pp. 1–10.
- Wan, C.; Cao, Z.; Lee, W.-J.; Song, Y.; Ju, P. An Adaptive Ensemble Data Driven Approach for Nonparametric Probabilistic Forecasting of Electricity Load. *IEEE Trans. Smart Grid* **2021**, *12*, 5396–5408. [CrossRef]
- Li, Z.; Ma, P.; Wang, X.; Xu, J.; Wan, X. An improved BP neural network method for Wind Power Prediction. In Proceedings of the 2022 IEEE 5th International Electrical and Energy Conference (CIEEC), Nanjing, China, 27–29 May 2022; pp. 653–658.
- Yang, R.; Sun, Z.; Lei, X. Photovoltaic Power Prediction Based on Stacking Model Fusion. *Comput. Syst. Appl.* **2020**, *29*, 36–45.
- Cao, Y.; Liu, G.; Luo, D.; Bavirisetti, D.P.; Xiao, G. Multi-timescale photovoltaic power forecasting using an improved Stacking ensemble algorithm based LSTM-Informer mode. *Energy* **2023**, *283*, 128669. [CrossRef]
- Teng, X. *Research on SOC Prediction of New Energy Vehicle Power Battery Based on Stacking Algorithm*; Chongqing Technology and Business University: Chongqing, China, 2022.
- Yashwanth, B.; Jaisharma, K. An Automated Users Profile Classification in Instagram to Improve the Precision Using an Integrated Neural Network Model and Decision Tree Algorithm. In Proceedings of the 2023 6th International Conference on Contemporary Computing and Informatics (IC3I), Gautam Buddha Nagar, India, 14–16 September 2023; pp. 2553–2557.



14. Okeleye, S.A.; Thiruvengadam, A.; Perhinschi, M.G.; Carder, D. Data-driven machine learning model of a Selective Catalytic Reduction on Filter (SCRf) in a heavy-duty diesel engine: A comparison of Artificial Neural Network with Tree-based algorithms. *Energy* **2024**, *290*, 130117. [[CrossRef](#)]
15. Zhou, Z. *Machine Learning*; Tsinghua University Press: Beijing, China, 2016.
16. Pramudita, R.; Solikin; Safitri, N. Optimization Analysis of Neural Network Algorithms Using Bagging Techniques on Classification of Date Fruit Types. In Proceedings of the 2022 Seventh International Conference on Informatics and Computing (ICIC), Denpasar, Bali, Indonesia, 8–9 December 2022; pp. 1–4.
17. Assaad, M.; Romuald, B.; Cardot, H. A New Boosting Algorithm for Improved Time-Series Forecasting with Recurrent Neural Networks. *Inf. Fusion* **2008**, *9*, 41–55. [[CrossRef](#)]
18. Sotnikov, D.; Lyly, M.; Salmi, T. Prediction of 2G HTS Tape Quench Behavior by Random Forest Model Trained on 2-D FEM Simulations. *IEEE Trans. Appl. Supercond.* **2023**, *33*, 1–5. [[CrossRef](#)]
19. Badola, S.; Mishra, V.N.; Parkash, S. Landslide susceptibility mapping using XGBoost machine learning method. In Proceedings of the 2023 International Conference on Machine Intelligence for GeoAnalytics and Remote Sensing (MIGARS), Hyderabad, India, 27–29 January 2023; pp. 1–4.
20. Kumar, D.; Abhinav, R.; Pindoriya, N. An Ensemble Model for Short-Term Wind Power Forecasting using Deep Learning and Gradient Boosting Algorithms. In Proceedings of the 2020 21st National Power Systems Conference (NPSC), Gandhinagar, India, 17–19 December 2020; pp. 1–6.
21. Breiman, L. Stacked regressions. *Mach. Learn.* **1996**, *24*, 49–64. [[CrossRef](#)]
22. Dietterich, T. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Comput.* **1998**, *10*, 1895–1923. [[CrossRef](#)] [[PubMed](#)]
23. Thanh, T.; Van, L.; Binh, L.M. Effects of Data Standardization on Hyperparameter Optimization with the Grid Search Algorithm Based on Deep Learning: A Case Study of Electric Load Forecasting. *Adv. Technol. Innov.* **2022**, *7*, 258.
24. Zhou, Q.; Ma, Y.; Lv, Q.; Zhang, R.; Wang, W.; Yang, S. Short-Term Interval Prediction of Wind Power Based on KELM and a Universal Tabu Search Algorithm. *Sustainability* **2022**, *14*, 10779. [[CrossRef](#)]
25. Zhao, Y.; Hu, Q.; Srinivasan, D.; Wang, Z. Data-driven correction approach to refine power curve of wind farm under wind curtailment. *IEEE Trans. Sustain. Energy* **2018**, *9*, 95–105. [[CrossRef](#)]
26. Hu, Y.; Qiao, Y.; Liu, J.; Zhu, H. Adaptive confidence boundary modeling of wind turbine power curve using SACADA data and its application. *IEEE Trans. Sustain. Energy* **2018**, *10*, 1330–1341. [[CrossRef](#)]
27. Xing, Z.; Zhi, Y.; Hao, R.-H.; Yan, H.-W.; Qing, C. Wind Speed Forecasting Model Based on Extreme Learning Machines and Complete Ensemble Empirical Mode Decomposition. In Proceedings of the 2020 5th Asia Conference on Power and Electrical Engineering (ACPEE), Chengdu, China, 4–7 June 2020; pp. 159–163.
28. Pan, G.; Zhang, H.; Ju, W.; Yang, W.; Qin, C.; Pei, L.; Sun, Y.; Wang, R. A Prediction Method for Ultra Short-Term Wind Power Prediction Basing on Long Short -Term Memory Network and Extreme Learning Machine. In Proceedings of the 2020 Chinese Automation Congress (CAC), Shanghai, China, 6–8 November 2020; pp. 7608–7612.
29. Jizmundo, R.E.; Maltezo, R.J.F.; Villanueva, F.G.; Pacis, M.C. A Long-Term Wind Power Prediction using Support Vector Regression and Ensemble Boosted Tree Algorithm (SVR-EBTA). In Proceedings of the 2023 15th International Conference on Computer and Automation Engineering (ICCAE), Sydney, Australia, 3–5 March 2023; pp. 149–152.
30. Arshad, J.; Zameer, A.; Khan, A. Wind Power Prediction Using Genetic Programming Based Ensemble of Artificial Neural Networks (GPeANN). In Proceedings of the 2014 12th International Conference on Frontiers of Information Technology, Islamabad, Pakistan, 17–19 December 2014; pp. 257–262.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.