WIND POWER PREDICTION BASED ON MACHINE LEARNING AND DEEP LEARNING MODELS

Report submitted to the SASTRA Deemed to be University as the requirement for the course

INT300 - MINI PROJECT

Submitted by

DANESHWAR B

(Reg. No.: 125015028**,** B. TECH INFORMATION TECHNOLOGY**)**

NITHISH SV

(Reg. No.: 125015077**,** B. TECH INFORMATION TECHNOLOGY**)**

ASHWIN S

(Reg. No.: 125015018**,** B. TECH INFORMATION TECHNOLOGY**)**

May 2024

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act, 1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

THANJAVUR | KUMBAKONAM | CHENNAI

SCHOOL OF COMPUTING

THANJAVUR – 613 401

Bonafide Certificate

This is to certify that the report titled "**Wind Power Prediction Based on Machine Learning and Deep Learning Models**" submitted as a requirement for the course, CSE300 / INT300 / ICT300: **MINI PROJECT** for B.Tech. is a bonafide record of the work done by **Mr. Ashwin S (Reg. No.125015018, B.Tech Information Technology),Mr. Daneshwar B (Reg. No.125015028, B.Tech Information Technology) ,Mr. Nithish SV (Reg. No.125015077, B.Tech Information Technology)** during the academic year 2023-24, in the School of Computing, under my supervision.

Signature of Project Supervisor : A. Emily Jenifer

Name with Affiliation : Emily Jenifer A, (AP III SOC)

Date : 20/04/2004

Mini Project *Viva voce* held on _____

Examiner 1 **Examiner 2**

ACKNOWLEDGEMENTS

We would like to thank our Honorable Chancellor **Prof. R. Sethuraman** for providing us with an opportunity and the necessary infrastructure for carrying out this project as a part of our curriculum.

We would like to thank our Honorable Vice-Chancellor **Dr. S. Vaidhyasubramaniam** and **Dr. S. Swaminathan**, Dean, Planning & Development, for the encouragement and strategic support at every step of our college life.

We extend our sincere thanks to **Dr. R. Chandramouli**, Registrar, SASTRA Deemed to be University for providing the opportunity to pursue this project.

We extend our heartfelt thanks to **Dr. V. S. Shankar Sriram**, Dean, School of Computing, **Dr. R. Muthaiah**, Associate Dean, Research, **Dr. K.Ramkumar**, Associate Dean, Academics, **Dr. D. Manivannan**, Associate Dean, Infrastructure, **Dr. R. Algeswaran**, Associate Dean, Students Welfare

Our guide Emily Jenifer A, (AP III SOC), School of Computing was the driving force behind this whole idea from the start. Her deep insight in the field and invaluable suggestions helped us in making progress throughout our project work. We also thank the project review panel members for their valuable comments and insights which made this project better.

We would like to extend our gratitude to all the teaching and non-teaching faculties of the School of Computing who have either directly or indirectly helped us in the completion of the project.

We gratefully acknowledge all the contributions and encouragement from my family and friends resulting in the successful completion of this project. We thank you all for providing us an opportunity to showcase our skills through project.

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

LSTM:	Long Short-Term Memory
RMSE:	Root Mean Square Error
MAE:	Mean Absolute Error
MBE:	Mean Bias Error
R^2:	Coefficient of Determination
NSE:	Nash Sutcliffe Efficiency
DNN:	Deep Neural Network
KNN:	K-Nearest Neighbor
RF:	Random Forest
PSO:	Particle Swarm Optimization
GRU:	Gated Recurrent Unit

NOTATIONS

- V'_n : Predicted wind power value at nth instance
- V_n : Actual wind power value at nth instance
- $\overline{V'}$: Arithmetic mean of predicted wind power values
- $\overline{V}$: Arithmetic mean of actual wind power values
- N : Total number of observations in the dataset
- n : Index for each observations in the dataset

Abstract

One of the sustainable ways to produce renewable energy is through wind power. With the primary objective of fostering sustainable growth, several nations have recently shifted to using renewable energy sources, such as wind and solar energy, to meet their future energy needs. This article builds a number of deep learning and machine learning models as foundation models in order to achieve the prediction of wind power generation. These regression models include the following: averaging model, random forest (RF) regressor, bagging regressor, gradient boosting (GB) regressor, deep neural network (DNN), k-nearest neighbor (KNN) regressor, Gated Recurrent Unit (GRU), and long short-term memory (LSTM), long short-term memory (LSTM) with PSO and Gated Recurrent Unit (GRU) with PSO. Preprocessing for statistics and data cleaning were also completed. This analysis uses a dataset with 50530 occurrences and four features. Five evaluation standards have been utilized to estimate the efficiency of the regression models, namely, Mean Absolute Error (MAE), Nash Sutcliffe performance (NSE), Mean Squared Error (MSE), coefficient of determination(R^2), Root Mean Squared Error (RMSE). The experimental findings showed that, in terms of wind power value prediction, the Gated Recurrent Unit (GRU) model produced the best results, with an R^2equal to 99.97%.

KEY WORDS: Neural network, Regressor, Feature extraction, Optimization

TABLE OF CONTENTS

CHAPTER 1

SUMMARY OF THE BASE PAPER

" Wind Power Prediction Based on Machine Learning and Deep Learning Models"
in Computers, Materials & Continua 2023, 74(1), 715-732., indexed in SCI-E[1].

1.1 CONTENT, NOVELTY OF THE BASE PAPER:

Based on the basic research, we investigate the use of deep learning and machine learning techniques for predicting wind power generation. Understanding the intricacies and difficulties related to the dynamics of offshore wind turbines and their surrounding environments, our research makes use of deep learning, regression, and neural networks techniques to improve prediction accuracy. We offer a thorough technique for predicting wind power through data pre-processing, cleaning, and evaluation utilizing metrics including MAE, NSE, MSE, R^2, and RMSE. The efficiency of the suggested strategy is demonstrated by a comparative examination of several regression models, such as DNN[2], KNN, LSTM, RF, bagging regressor, and gradient boosting, GRU, GRU with PSO, LSTM with PSO. All in all, these models seek to advance the field of renewable energy by offering a strong foundation for precise wind power prediction, supporting maintenance planning, profit maximization, and power grid resilience.

Proposed Model: First, we started combining PSO with an LSTM-based hybrid optimizer algorithm to maximize its potential. Three main layers comprised this architecture: five stacked hidden layers with rectified linear unit(ReLU) activation functions, input layers that represented the wind dataset's characteristics, and the basic DNN layers[2] with an Adam optimizer. But as deep learning is a dynamic field, we moved on to incorporate the Gated Recurrent Unit (GRU) model. The unique gating mechanisms of the GRU enable better long-term dependence capture and increase the precision of wind power prediction.

Overcoming Limitations: Three primary concerns in the prediction of wind power are feature extraction, non-linear dynamics, and data uncertainty. We have carefully constructed our hybrid approach, which is based on GRU, to get around these challenges. Improved feature extraction, data collection, and prediction accuracy are highlighted, especially for predicting the future that is essential for managing and developing renewable energy sources.

Better Prediction Accuracy: Over a variety of wind datasets, our GRU-based model has continuously shown improved prediction accuracy. The GRU's sophisticated architecture, which excels at capturing temporal dependencies, is responsible for its capacity to control wind power fluctuation and optimize network deployment management.

Our study's innovative use of the GRU model propels advancements in the renewable energy sector. The proposed architecture not only ensures a robust wind power prediction

system but also facilitates power grid stability, simplifies maintenance, and boosts revenue in renewable energy systems.

This fundamental work is distinctive because it offers a brand-new hybrid optimizer for the prediction of wind power. In order to overcome the inherent difficulties and constraints of the current wind power prediction techniques, we first merged LSTM with PSO. To improve prediction accuracy even more, After learning about the special benefits and capabilities of the Gated Recurrent Unit (GRU) paradigm, we independently investigated its implementation.

This novel technique overcomes the shortcomings of earlier methods while significantly increasing precision in predictions. The hybrid optimizer offers a stable and effective model for wind power prediction because of its capacity to handle the intricacies involved in wind energy dynamics.

Dataset Name	Number of Instances	Number of Features
T1.csv	50531	5

Table 1.1: Dataset

No.	Characteristics
I.	Date/Time
II.	LV ActivePower (kW)
III.	Wind Speed (m/s)
IV.	Theoretical_Power_Curve (KWh)
V.	Wind Direction (°)

Table 1.2: Features of Dataset

1.2 ADDRESSED RESEARCH AND PROPOSED SOLUTION:

This foundation paper investigates the challenging task of applying deep learning and machine learning approaches for wind power prediction, with a focus on offshore areas. Acknowledging the constraints and intricacies present in conventional techniques for predicting wind power, we present a novel hybrid model. Our proposed model combines optimization techniques, notably Particle Swarm Optimization (PSO), with Long Short-Term Memory (LSTM) networks to improve assessment metrics and prediction accuracy.

We furthermore implemented a Gated Recurrent Unit (GRU) model in order to investigate the possibilities of various neural network designs. Because of its ability to detect long-range dependencies and adapt to shifting input sequences, particularly in scenarios where temporal dynamics are crucial, the GRU is an excellent choice for wind power prediction.

This study evaluates the proposed hybrid model's performance according to certain evaluation criteria, such as Root Mean Squared Error (RMSE), Coefficient of Determination (R^2), Mean Absolute Error (MAE), Nash Sutcliffe Efficiency (NSE), and Mean Squared Error (MSE). The proposed hybrid strategy outperforms existing deep learning and machine learning models in terms of improving prediction accuracy overall, regulating variability in wind power, and simplifying network deployment management.

This little project's main goal is to advance the technologies related to renewable energy sources by offering a stable and practical framework for wind power prediction. The suggested hybrid model can assist in improving power grid stability, optimizing maintenance schedules, and maximizing profit in renewable energy systems by precisely forecasting wind power. This would help promote the shift to a more environmentally friendly and sustainable energy infrastructure.

1.3 ARCHITECTURE, ALGORITHM PROPOSED AND ITS CORRECTNESS:

Framework: The Fig 1.1 depicts the framework of the proposed wind power prediction

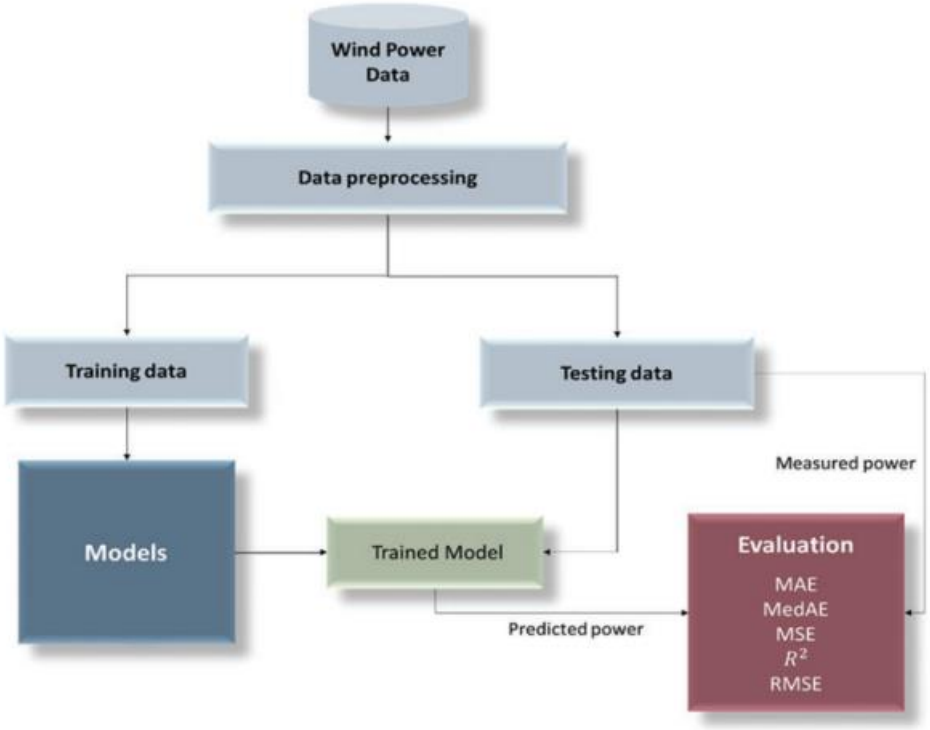


Fig. 1.1. Framework of the proposed wind power prediction[1]

The suggested hybrid optimizer technique integrates Long Short-Term Memory (LSTM) and Particle Swarm Optimization (PSO). An independent model was developed and studied in order to gain a deeper understanding of the Gated Recurrent Unit (GRU) model's performance in wind power prediction. The model begins by processing input features selected from the wind dataset by traveling through the first three fundamental layers of the DNN. After that, these features are routed using five stacked hidden layers that utilize the Adam optimizer and ReLU activation functions. Subsequently, the LSTM layer is employed to evaluate the time series data and derive temporal associations from the wind power data. The PSO optimization paradigm is applied to enhance the model's performance and enhance the outputs of the evaluation criteria.

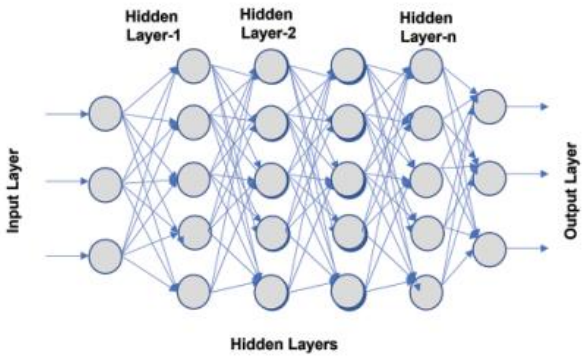


Fig. 1.2. Deep Neural Architecture with multiple layers[1]

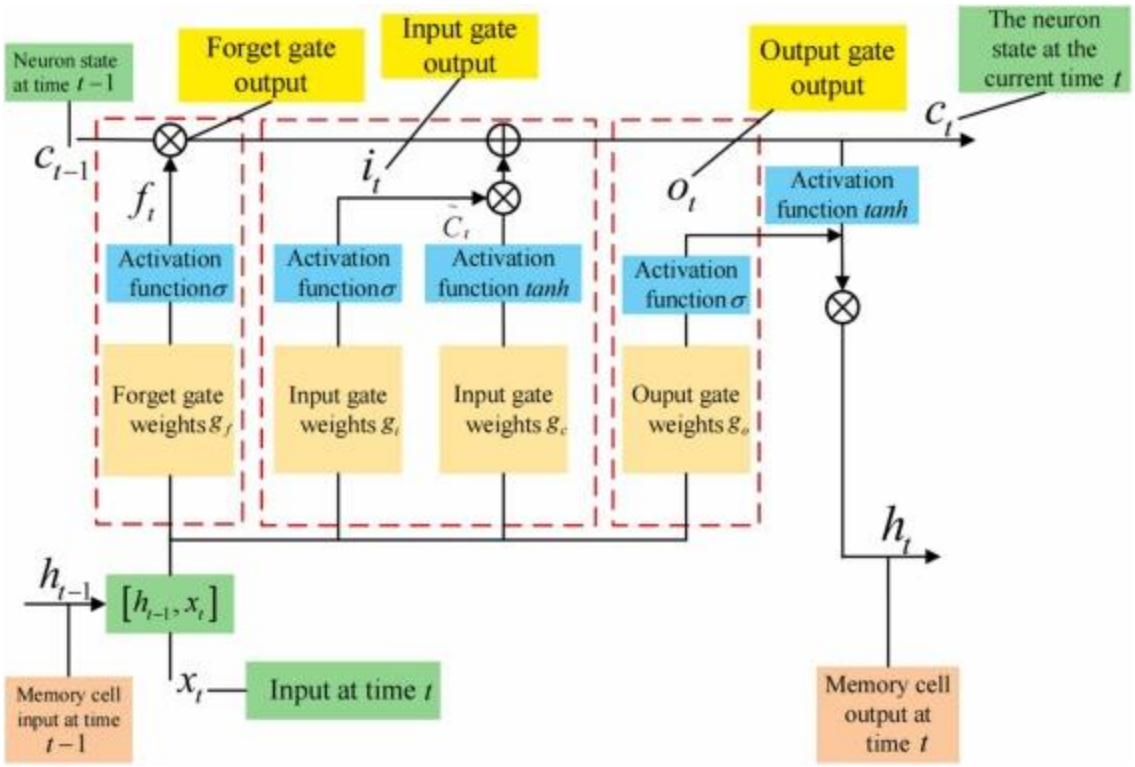


Fig. 1.3. The architecture of LSTM cell[3]

The correctness of the proposed hybrid optimizer algorithm with LSTM and the independently developed GRU model is evaluated using a wide range of evaluation criteria, including Mean Absolute Error (MAE), Nash Sutcliffe Efficiency (NSE), Mean Squared Error (MSE), Coefficient of Determination (R^2), and Root Mean Squared Error (RMSE).

Cho et al. [5] first proposed GRU as a more condensed and easier-to-implement hidden unit that was modeled after the LSTM unit. Without discrete memory cells, GRUs [6] have reset and update gates that adaptively govern how much each hidden unit remembers or forgets during training. This indicates that, based on the gating mechanisms' activity frequency, each hidden unit can adaptively record dependencies across a range of time scales. For

instance, frequent reset gate activity will be used to identify short-term dependencies, and frequent update gate activity will be used to identify long-term dependencies [5,7].

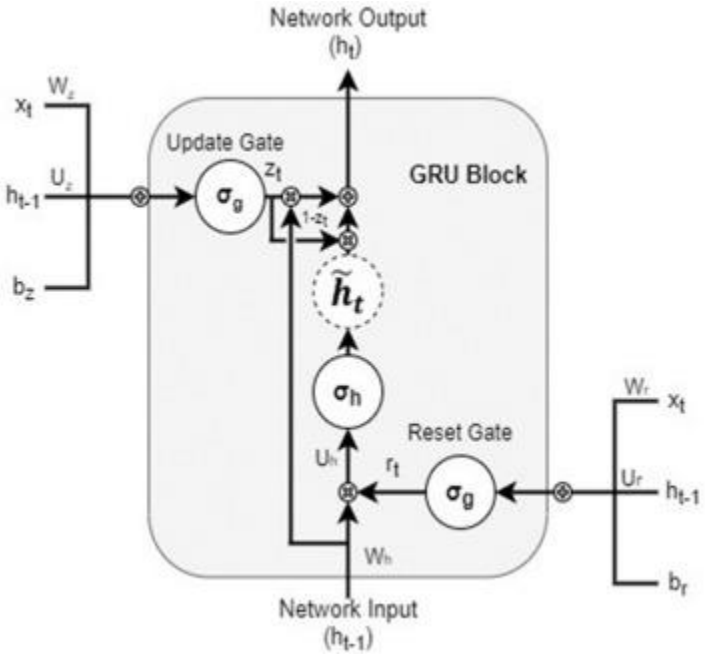


Fig. 1.4. The architecture of GRU cell.[4]

According to experimental findings, the suggested LSTM-PSO model performs better in terms of prediction accuracy than both deep learning and conventional machine learning models, especially when used with an unbalanced wind dataset. Comparing the suggested hybrid model and the GRU model to other wind power prediction models now in use further validates their efficacy in managing wind power variability, simplifying network deployment management, and improving overall prediction accuracy.

CHAPTER 2

MERITS AND DEMERITS OF THE PAPER

2.1 MERITS:

Innovative Modeling: In the wind power prediction, an optimized Long Short-Term Memory(LSTM) model and Particle Swarm Optimization (PSO) are introduced. A different GRU model has also been created to investigate its potential in wind power prediction. This integration overcomes the shortcomings of traditional methods and improves predictions by utilizing cutting-edge optimization algorithms.

Superior Prediction Accuracy: The proposed LSTM model predicts wind power amounts with remarkable precision.The independent GRU model also demonstrates promising results, indicating that deep learning architectures may have more applications in wind power prediction. The models' potential to maximize energy production and management in wind farms is suggested by their ability to predict wind power values with reliability.

Robust Evaluation Metrics: The study employs an extensive range of assessment measures, such as RMSE, MAE, NSE, and R^2, to assess the efficacy of the proposed prediction technique. This thorough assessment framework supports trust in the model's usefulness by guaranteeing the validity and dependability of the model's predictions.

Future Expansion Potential: In addition to reporting its findings, the paper makes recommendations for further research and development. The model's scalability and adaptability to changing energy prediction issues are indicated by recommendations to expand to new prediction tasks, investigate multilingual implementations, and integrate a variety of datasets.

2.2 DEMERITS:

Ambiguity in Prediction Definition: A precise and succinct description of wind power prediction would be beneficial to the investigation. For the purpose of creating effective prediction algorithms and guaranteeing reliable and consistent predictions, a thorough understanding is essential.

Limited Dataset Description: The wind power dataset used for training and evaluation is not thoroughly explained in the publication. The dataset's representativeness, diversity, and quality are inextricably linked to the model's ability to make predictions. Enhancing the evaluation of the model's generalizability and reliability would require a more comprehensive explanation of the dataset.

Language and Geographic Restrictions: The study predominantly focuses on wind power prediction without addressing its applicability across various geographical locations and languages. Given the global nature of wind energy production, this limitation could hinder the model's broader applicability and effectiveness.

Limited Discussion on Constraints: Potential limitations are mentioned in passing throughout the work, but no in-depth analysis or discussion is done. Further insights into the model's shortcomings and difficulties will be helpful in directing future developments in wind power prediction work.

S.No	Title of the base paper	Author	Year	Method
1.	Wind power prediction based on high-frequency SCADA data along with isolation forest and deep learning neural networks.[2]	Z. Lin, X. Liu and M. Collu	2020	condition monitoring (CM), supervisory control and data acquisition (SCADA) data, Gaussian Process (GP), isolate forest (IF).
2.	Efficient Wind Power Prediction Using Machine Learning Methods: A Comparative Study.[8]	Abdulelah Alkesaiberi, Fouzi Harrou	2022	Bayesian optimization (BO) ,Gaussian process regression (GPR), Support Vector Regression (SVR) with different kernels, and ensemble learning (ES) models
3.	A Novel Deep Learning Approach for Wind Power Forecasting Based on WD-LSTM Model.[9]	B. Liu, S. Zhao, X.Yu, L. Zhang , Q.Wang,	2020	Wavelet Decomposition (WD) and Long Short-Term Memory neural network (LSTM)
4.	Short-term forecasting and uncertainty analysis of wind turbine power based on long short-term memory network and Gaussian mixture model.[10]	Jinhua Zhang, Jie Yan	2019	Gaussian mixture model (GMM), long short-term memory network (LSTM), radial basis function (RBF),deep belief network (DBN),back propagation neural networks (BPNN), and Elman neural network (ELMAN).
5.	Prediction of wind speed using a new Grey-extreme learning machine hybrid algorithm: A case study[11]	Mojtaba Qolipour, Ali Mostafaeipour	2019	Extreme Learning Machine (ELM).
6.	Wind turbine power curve modeling using radial basis function neural networks and tabu search.[12]	D.Karamichailidou, V. Kaloutsa and A. Alexandridis,	2021	artificial neural networks (ANNs), radial basis function (RBF), non-symmetric fuzzy means (NSFM).
7.	Wind power forecasting based on daily wind speed data using machine learning algorithms.[13]	H. Demolli, A. S. Dokuz, A. Ecemis andM. Gokcek	2019	Least Absolute Shrinkage Selector Operator (LASSO), K Nearest Neighbor (kNN), eXtreme Gradient Boost (XGBoost), Random Forest (RF), and Support Vector Regression (SVR) algorithms
8.	LSTM-EFG for wind power forecasting based on sequential correlation features.[14]	R. Yu, J. Gao, M. Yu,W. Lu, T. Xu et al.,	2019	Long Short-Term Memory-enhanced forget-gate network (LSTM-EFG)

Table 2.1 Literature Survey

2.3 EVALUVATION CRITERIA:

The evaluation of the wind power prediction models is conducted using a comprehensive set of metrics outlined in Table 3. These metrics include:

Root Mean Square Error (RMSE): calculates the average error size between values that are anticipated and those that are observed.

Mean Absolute Error (MAE): arithmetizes the absolute difference between the values that were predicted and those that were seen.

Mean Bias Error (MBE): shows the mean variation between the values that were anticipated and those that were observed.

Coefficient of Determination (R^2): gives an indication of how well the observed values match the projected values.

Nash Sutcliffe Efficiency (NSE): Evaluates the model's ability to predict values relative to observed data.

Metric	**Value**
RMSE	$\sqrt{\frac{1}{N}}\sum_{n=1}^{N}\left[\widehat{V}_n - V_n\right]^2$
MAE	$\frac{1}{N}\sum_{n=1}^{N}\left\|\widehat{V}_n - V_n\right\|$
MBE	$\frac{1}{N}\sum_{n=1}^{N}\left(\widehat{V}_n - V_n\right)$
R^2	$1 - \frac{\sum_{n=1}^{N}\left(V_n - \widehat{V}_n\right)^2}{\sum_{n=1}^{N}((\sum_{n=1}^{N} V_n) - V_n)^2}$
NSE	$1 - \frac{\sum_{n=1}^{N}\left(V_n - \widehat{V}_n\right)^2}{\sum_{n=1}^{N}(V_n - \overline{\widehat{V}}_n)^2}$

Table 2.2 Evaluation Criteria

CHAPTER 3

SOURCE CODE:

EDA:

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from windrose import WindroseAxes
from statsmodels.graphics.tsaplots import plot_pacf, plot_acf
%matplotlib inline
raw_data = pd.read_csv('T1.csv')
raw_data.head()
#create a loss column
raw_data['Loss'] = raw_data['Theoretical_power'] - raw_data['Power']
raw_data.head(3)
# create an hourly,daily,weekly,monthly data frame resampled by the mean
hourly = pd.DataFrame()
daily = pd.DataFrame()
weekly = pd.DataFrame()
monthly = pd.DataFrame()
for col in raw_data.columns:
    weekly[col] = raw_data[col].resample('W').mean()
for col in raw_data.columns:
    monthly[col] = raw_data[col].resample('M').mean()
for col in raw_data.columns:
    daily[col] = raw_data[col].resample('D').mean()
for col in raw_data.columns:
    hourly[col] = raw_data[col].resample('H').mean()
hourly['Power'].plot(figsize=(15,5))
plt.show()
```

```
# create a function for a categorical column
def direction(x):
    if x > 348.75 or x<11.25: return 'N'
    if x < 33.75: return 'NNE'
    if x < 56.25: return 'NE'
    if x < 78.75: return 'ENE'
    if x < 101.25: return 'E'
    if x < 123.75: return 'ESE'
    if x < 146.25: return 'SE'
    if x < 168.75: return 'SSE'
    if x < 191.25: return 'S'
    if x < 213.75: return 'SSW'
    if x < 236.25: return 'SW'
    if x < 258.75: return 'WSW'
    if x < 281.25: return 'W'
    if x < 303.75: return 'WNW'
    if x < 326.25: return 'NW'
    else: return 'NNW'
#create a categorical column for the direction of wind
daily['Direction'] = daily['Wind_direction'].apply(direction)
daily.head(3)
#check number of data points in raw_data where the wind speed is above 3.3 and power is less than zero
#create a dataframe where times of maintenance are not included
raw_data_nm = raw_data[~((raw_data['Power']<=0) & (raw_data['Wind_speed'] > 3.3))]
raw_data_nm
#function to create x,y component of wind direction
def x_y_component(wind_direction, wind_speed):
    #convert to radians
    #Convert degrees to x,y components
    radians = (wind_direction * np.pi)/180
    # give the x, y compenents
```

```
    x = wind_speed * np.cos(radians)
    y = wind_speed * np.sin(radians)
    return x,y
# create two extra columns in raw_data_nm for x,y compnenents of wind direction
raw_data_nm['x_com'], raw_data_nm['y_com'] =
x_y_component(raw_data_nm['Wind_direction'],raw_data_nm['Wind_speed'])
raw_data_nm.head(3)
#ACF : ckecks to see if the previos time step has an impact on the next time step
#PCF : see which lag has an impact on the next time step.
plot_acf(hourly_nm['Power'], lags=30)
plt.savefig('acf.png')
plt.show()
plt.figsize=(20,5)
plot_pacf(hourly_nm['Power'], lags=30)
plt.savefig('pacf.png')
plt.show()
hourly_nm['T_1'] = hourly_nm['Power'].shift(1)
hourly_nm = hourly_nm.dropna()
hourly_nm.head(3)
#Dataframe after completing EDA
import tensorflow as tf
import pandas as pd
import numpy as np
import random
import os
df = pd.read_csv("hourly_nm.csv")
df
#Normalize the dataset using MinMaxScaler
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

```
df[["Power","Wind_speed","Theoretical_power","Loss","x_com","y_com","T_1"]] =
scaler.fit_transform(df[["Power","Wind_speed","Theoretical_power","Loss","x_com","y_
com","T_1"]])

X = df.iloc[:,2:].values

y = df.iloc[:,1].values

#Train and test split

X_train = X[:8351]

y_train = y[:8351]

X_test = X[8351:]

y_test = y[8351:]

#evaluation criteria NSE ,MBE

def nse(predictions, targets):

    return ((np.sum((targets-predictions)**2))/(np.sum((targetsnp.mean(predictions))**2)))

def mbe(predictions,targets):

    return (np.mean(predictions - targets))
```

Gradient Boosting Regressor:

```
from sklearn.metrics import mean_absolute_error,r2_score,mean_squared_error

from sklearn.ensemble import GradientBoostingRegressor

gb_regressor = GradientBoostingRegressor(n_estimators=200,
learning_rate=0.1,random_state=42)

gb_regressor.fit(X_train, y_train)

y_pred_gbr = gb_regressor.predict(X_test)

mae_gbr = mean_absolute_error(y_test, y_pred_gbr)

r2_gbr = r2_score(y_test,y_pred_gbr)*100

nse_gbr = nse(y_pred_gbr,y_test)

nse_gbr = "{:.1e}".format(nse_gbr)

mbe_gbr = mbe(y_pred_gbr,y_test)

rmse_gbr = np.sqrt(mean_squared_error(y_test, y_pred_gbr))

metrics_gbr = [mae_gbr,nse_gbr,mbe_gbr,r2_gbr,rmse_gbr]

table_metrics_gbr = pd.DataFrame(metrics_gbr)
```

```
table_metrics_gbr.index = ["MAE","NSE","MBE","R2","RMSE"]

table_metrics_gbr.columns = ["Gradient Boosting Model"]

table_metrics_gbr

import matplotlib.pyplot as plt

plt.figure(figsize=(7,5))

plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='black',
linestyle='solid', lw=3, label='Predicted')

plt.scatter(y_test, y_pred_gbr, color='dodgerblue', marker='o', label='Actual')

plt.title('GB Regressor Model')

plt.legend()

plt.show()
```

Random Forest Regressor:

```
from sklearn.ensemble import RandomForestRegressor

rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)

rf_regressor.fit(X_train, y_train)

y_pred_rf = rf_regressor.predict(X_test)

mae_rf = mean_absolute_error(y_test, y_pred_rf)

r2_rf = r2_score(y_test,y_pred_rf)*100

nse_rf = nse(y_pred_rf,y_test)

nse_rf = "{:.1e}".format(nse_rf)

mbe_rf = mbe(y_pred_rf,y_test)

rmse_rf = np.sqrt(mean_squared_error(y_test, y_pred_rf))

metrics_rf = [mae_rf,nse_rf,mbe_rf,r2_rf,rmse_rf]

table_metrics_rf = pd.DataFrame(metrics_rf)

table_metrics_rf.index = ["MAE","NSE","MBE","R2","RMSE"]

table_metrics_rf.columns = ["Random Forest Model"]

table_metrics_rf

plt.figure(figsize=(7,5))

plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='black',
linestyle='solid', lw=3, label='Predicted')
```

```
plt.scatter(y_test, y_pred_rf, color='dodgerblue', marker='o', label='Actual')
plt.title('Random Forest Regressor Model')
plt.legend()
plt.show()
```

Bagging Regressor:

```
from sklearn.ensemble import BaggingRegressor
from sklearn.tree import DecisionTreeRegressor
base_regressor = DecisionTreeRegressor()
bagging_regressor = BaggingRegressor(base_estimator=base_regressor, n_estimators=10, random_state=42)
bagging_regressor.fit(X_train, y_train)
y_pred_br = bagging_regressor.predict(X_test)
mae_br = mean_absolute_error(y_test, y_pred_br)
r2_br = r2_score(y_test,y_pred_br)*100
nse_br = nse(y_pred_br,y_test)
nse_br = "{:.1e}".format(nse_br)
mbe_br = mbe(y_pred_br,y_test)
rmse_br = np.sqrt(mean_squared_error(y_test, y_pred_br))
metrics_br = [mae_br,nse_br,mbe_br,r2_br,rmse_br]
table_metrics_br = pd.DataFrame(metrics_br)
table_metrics_br.index = ["MAE","NSE","MBE","R2","RMSE"]
table_metrics_br.columns = ["Bagging Regressor Model"]
table_metrics_br
plt.figure(figsize=(7,5))
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='black', linestyle='solid', lw=3, label='Predicted')
plt.scatter(y_test, y_pred_br, color='dodgerblue', marker='o', label='Actual')
plt.title('Bagging Regressor Model')
plt.legend()
plt.show()
```

K Nearest Neighbor Regressor:

```
from sklearn.neighbors import KNeighborsRegressor
knn_regressor = KNeighborsRegressor(n_neighbors=5,weights="distance")
knn_regressor.fit(X_train, y_train)
y_pred_knn = knn_regressor.predict(X_test)
mse_knn = mean_squared_error(y_test, y_pred_knn)
mae_knn = mean_absolute_error(y_test, y_pred_knn)
r2_knn = r2_score(y_test,y_pred_knn)*100
nse_knn = nse(y_pred_knn,y_test)
nse_knn = "{:.1e}".format(nse_knn)
mbe_knn = mbe(y_pred_knn,y_test)
rmse_knn = np.sqrt(mean_squared_error(y_test, y_pred_knn))
metrics_knn = [mae_knn,nse_knn,mbe_knn,r2_knn,rmse_knn]
table_metrics_knn = pd.DataFrame(metrics_knn)
table_metrics_knn.index = ["MAE","NSE","MBE","R2","RMSE"]
table_metrics_knn.columns = ["KNN Model"]
table_metrics_knn
plt.figure(figsize=(7,5))
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='black',
linestyle='solid', lw=3, label='Predicted')
plt.scatter(y_test, y_pred_knn, color='dodgerblue', marker='o', label='Actual')
plt.title('KNN Regressor Model')
plt.legend()
plt.show()
```

Averaging Model:

```
y_pred_avg = (y_pred_rf + y_pred_br + y_pred_gbr) / 3
mse_avg = mean_squared_error(y_test, y_pred_avg)
mae_avg = mean_absolute_error(y_test, y_pred_avg)
```

```
r2_avg = r2_score(y_test,y_pred_avg)*100
nse_avg = nse(y_pred_avg,y_test)
nse_avg = "{:.1e}".format(nse_avg)
mbe_avg = mbe(y_pred_avg,y_test)
rmse_avg = np.sqrt(mean_squared_error(y_test, y_pred_avg))
metrics_avg = [mae_avg,nse_avg,mbe_avg,r2_avg,rmse_avg]
table_metrics_avg = pd.DataFrame(metrics_avg)
table_metrics_avg.index = ["MAE","NSE","MBE","R2","RMSE"]
table_metrics_avg.columns = ["Averaging Model"]
table_metrics_avg
plt.figure(figsize=(7,5))
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='black',
linestyle='solid', lw=3, label='Predicted')
plt.scatter(y_test, y_pred_avg, color='dodgerblue', marker='o', label='Actual')
plt.title('Averaging Model')
plt.legend()
plt.show()
```

Deep Neural Network:

```
seed = 42
os.environ['PYTHONHASHSEED'] = str(seed)
np.random.seed(seed)
random.seed(seed)
tf.random.set_seed(seed)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
model_dnn = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(32, activation='relu'),
    Dense(16, activation='relu'),
```

```
    Dense(1)
])
optimizer = Adam(lr=0.0001)
model_dnn.compile(optimizer=optimizer,loss="mean_squared_error")
history_dnn = model_dnn.fit(X_train, y_train, epochs=50,
batch_size=64,validation_split=0.85)
y_pred_dnn = model_dnn.predict(X_test)
mse_dnn = mean_squared_error(y_test, y_pred_dnn)
mae_dnn = mean_absolute_error(y_test, y_pred_dnn)
r2_dnn = r2_score(y_test,y_pred_dnn)*100
nse_dnn = nse(y_pred_dnn,y_test)
nse_dnn = "{:.1e}".format(nse_dnn)
mbe_dnn = mbe(y_pred_dnn,y_test)
rmse_dnn = np.sqrt(mean_squared_error(y_test, y_pred_dnn))
metrics_dnn = [mae_dnn,nse_dnn,mbe_dnn,r2_dnn,rmse_dnn]
model_dnn.summary()
plt.plot(history_dnn.history['loss'], label='Training Loss')
plt.plot(history_dnn.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
plt.plot(history_dnn.history['loss'])
plt.title('Mean Squared Error vs Epochs')
plt.xlabel('Epoch')
plt.ylabel('Mean Squared Error')
plt.legend()
plt.show()
table_metrics_dnn = pd.DataFrame(metrics_dnn)
table_metrics_dnn.index = ["MAE","NSE","MBE","R2","RMSE"]
```

```
table_metrics_dnn.columns = ["DNN Model"]

table_metrics_dnn

plt.figure(figsize=(7,5))

plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='black',
linestyle='solid', lw=3, label='Predicted')

plt.scatter(y_test, y_pred_dnn, color='dodgerblue', marker='o', label='Actual')

plt.title('DNN Model')

plt.legend()

plt.show()
```

Long Short Term Memory (LSTM) Model:

```
from tensorflow.keras.layers import LSTM

from keras import regularizers

model_lstm = Sequential()

model_lstm.add(LSTM(64,activation='relu',input_shape=(X_train.shape[1], 1)))

model_lstm.add(Dense(32,activation='relu'))

model_lstm.add(Dense(16,activation='relu'))

model_lstm.add(Dense(1))

optimizer_lstm = Adam(lr=0.0001)

model_lstm.compile(optimizer=optimizer_lstm, loss='mean_squared_error')

history_lstm = model_lstm.fit(X_train, y_train, epochs=50,
batch_size=64,validation_split=0.85)

y_pred_lstm = model_lstm.predict(X_test)

mse_lstm = mean_squared_error(y_test, y_pred_lstm)

mae_lstm = mean_absolute_error(y_test, y_pred_lstm)

r2_lstm = r2_score(y_test, y_pred_lstm) * 100

nse_lstm = nse(y_pred_lstm, y_test)

nse_lstm = "{:.1e}".format(nse_lstm)

mbe_lstm = mbe(y_pred_lstm, y_test)

rmse_lstm = np.sqrt(mean_squared_error(y_test, y_pred_lstm))

'"print("Mean Absolute Error:", mae_lstm)
```

```
print("Root Mean Squared Error:", rmse_lstm)
print("R2 Score:", r2_lstm)
print("NSE:", nse_lstm)
print("MBE:", mbe_lstm)'"
metrics_lstm = [mae_lstm,nse_lstm,mbe_lstm,r2_lstm,rmse_lstm]
model_lstm.summary()
plt.plot(history_lstm.history['loss'], label='Training Loss')
plt.plot(history_lstm.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
plt.plot(history_lstm.history['loss'])
plt.title('Mean Squared Error vs Epochs')
plt.xlabel('Epoch')
plt.ylabel('Mean Squared Error')
plt.legend()
plt.show()
table_metrics_lstm = pd.DataFrame(metrics_lstm)
table_metrics_lstm.index = ["MAE","NSE","MBE","R2","RMSE"]
table_metrics_lstm.columns = ["LSTM Model"]
table_metrics_lstm
plt.figure(figsize=(7,5))
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='black',
linestyle='solid', lw=3, label='Actual')
plt.scatter(y_test, y_pred_lstm, color='dodgerblue', marker='o', label='Predicted')
plt.title('LSTM Model')
plt.legend()
plt.show()
```

Long Short Term Memory (LSTM) Model with PSO:

```
seed = 42
os.environ['PYTHONHASHSEED'] = str(seed)
np.random.seed(seed)
random.seed(seed)
tf.random.set_seed(seed)
def lstm_pso_optimization(params):
    model = keras.Sequential([
        keras.layers.LSTM(int(params[0]), activation='relu',
input_shape=(X_train.shape[1],1)),
        keras.layers.Dropout(int(params[1])),
        keras.layers.Dense(int(params[2]), activation='relu'),
        keras.layers.Dropout(int(params[3])),
        keras.layers.Dense(int(params[4]), activation='relu'),
        keras.layers.Dense(1),
    ])
    model.compile(optimizer=keras.optimizers.Adam(0.001), loss='mean_squared_error')
    model.fit(X_train, y_train, epochs=50, batch_size=64)
    y_pred = model.predict(X_test)
    r2 = r2_score(y_test, y_pred.round())
    return r2
from pyswarm import pso
import keras
lstm_lb = [8,0.1,8,0.1,8]
lstm_ub = [32,0.5,32,0.5,32]
best_params_lstm, _ =
pso(lstm_pso_optimization,lstm_lb,lstm_ub,swarmsize=10,maxiter=5)
lstm_neurons_layer1, lstm_dropout1, lstm_neurons_layer2, lstm_dropout2,
lstm_neurons_layer3 = best_params_lstm
from keras.layers import Dropout
model_lstm_pso = Sequential()
```

```
model_lstm_pso.add(LSTM(units=int(lstm_neurons_layer1), activation='relu',
input_shape=(X_train.shape[1], 1), recurrent_dropout=0.2))

model_lstm_pso.add(Dropout(int(lstm_dropout1)))

model_lstm_pso.add(Dense(units=int(lstm_neurons_layer2), activation='relu'))

model_lstm_pso.add(Dropout(int(lstm_dropout2)))

model_lstm_pso.add(Dense(units=int(lstm_neurons_layer3), activation='relu'))

model_lstm_pso.add(Dense(1))

optimizer_lstm_pso = Adam(lr=0.001)

model_lstm_pso.compile(optimizer=optimizer_lstm_pso, loss='mean_squared_error')

history_lstm_pso = model_lstm_pso.fit(X_train, y_train, epochs=100, batch_size=32,
validation_split=0.2)

y_pred_lstm_pso = model_lstm_pso.predict(X_test)

mse_lstm_pso = mean_squared_error(y_test, y_pred_lstm_pso)

mae_lstm_pso = mean_absolute_error(y_test, y_pred_lstm_pso)

r2_lstm_pso = r2_score(y_test, y_pred_lstm_pso) * 100

nse_lstm_pso = nse(y_pred_lstm_pso, y_test)

nse_lstm_pso = "{:.1e}".format(nse_lstm_pso)

mbe_lstm_pso = mbe(y_pred_lstm_pso, y_test)

rmse_lstm_pso = np.sqrt(mean_squared_error(y_test, y_pred_lstm_pso))

metrics_lstm_pso =
[mae_lstm_pso,nse_lstm_pso,mbe_lstm_pso,r2_lstm_pso,rmse_lstm_pso]

model_lstm_pso.summary()

plt.plot(history_lstm_pso.history['loss'], label='Training Loss')

plt.plot(history_lstm_pso.history['val_loss'], label='Validation Loss')

plt.title('Training and Validation Loss')

plt.xlabel('Epoch')

plt.ylabel('Loss')

plt.legend()

plt.show()

plt.plot(history_lstm_pso.history['loss'])

plt.title('Mean Squared Error vs Epochs')
```

```
plt.xlabel('Epoch')
plt.ylabel('Mean Squared Error')
plt.legend()
plt.show()
table_metrics_lstm_pso = pd.DataFrame(metrics_lstm_pso)
table_metrics_lstm_pso.index = ["MAE","NSE","MBE","R2","RMSE"]
table_metrics_lstm_pso.columns = ["PSO-SFS LSTM"]
table_metrics_lstm_pso
plt.figure(figsize=(7,5))
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='black',
linestyle='solid', lw=3, label='Actual')
plt.scatter(y_test, y_pred_lstm_pso, color='dodgerblue', marker='o', label='Predicted')
plt.title('LSTM Model With PSO Optimizer')
plt.legend()
plt.show()
```

GRU Model:

```
seed = 57
os.environ['PYTHONHASHSEED'] = str(seed)
np.random.seed(seed)
random.seed(seed)
tf.random.set_seed(seed)
from keras.layers import GRU
model_gru = Sequential()
model_gru.add(GRU(units=32, activation='relu', input_shape=(X_train.shape[1], 1)))
model_gru.add(Dense(units=16, activation='relu'))
model_gru.add(Dense(units=8, activation='relu'))
model_gru.add(Dense(1))
optimizer_gru = Adam(lr=0.001)
model_gru.compile(optimizer=optimizer_gru, loss='mean_squared_error')
history_gru = model_gru.fit(X_train, y_train, epochs=50, batch_size=64,
validation_split=0.2)
```

```
y_pred_gru = model_gru.predict(X_test)
mse_gru = mean_squared_error(y_test, y_pred_gru)
mae_gru = mean_absolute_error(y_test, y_pred_gru)
r2_gru = r2_score(y_test, y_pred_gru) * 100
nse_gru = nse(y_pred_gru, y_test)
nse_gru = "{:.1e}".format(nse_gru)
mbe_gru = mbe(y_pred_gru, y_test)
rmse_gru = np.sqrt(mean_squared_error(y_test, y_pred_gru))
metrics_gru = [mae_gru,nse_gru,mbe_gru,r2_gru,rmse_gru]
model_gru.summary()
plt.plot(history_gru.history['loss'], label='Training Loss')
plt.plot(history_gru.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
plt.plot(history_gru.history['loss'])
plt.title('Mean Squared Error vs Epochs')
plt.xlabel('Epoch')
plt.ylabel('Mean Squared Error')
plt.legend()
plt.show()
table_metrics_gru = pd.DataFrame(metrics_gru)
table_metrics_gru.index = ["MAE","NSE","MBE","R2","RMSE"]
table_metrics_gru.columns = ["PSO GRU"]
table_metrics_gru
plt.figure(figsize=(7,5))
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='black',
linestyle='solid', lw=3, label='Actual')
plt.scatter(y_test, y_pred_gru, color='dodgerblue', marker='o', label='Predicted')
```

```
plt.title('PSO GRU MODEL')
plt.legend()
plt.show()
```

GRU MODEL WITH PSO:

```
seed = 42
os.environ['PYTHONHASHSEED'] = str(seed)
np.random.seed(seed)
random.seed(seed)
tf.random.set_seed(seed)
def pso_optimization(params):
    model = keras.Sequential([
        keras.layers.GRU(int(params[0]), activation='relu',
input_shape=(X_train.shape[1],1)),
        keras.layers.Dense(int(params[1]), activation='relu'),
        keras.layers.Dense(int(params[2]), activation='relu'),
        keras.layers.Dense(1, activation='sigmoid'),
    ])
    model.compile(optimizer=keras.optimizers.Adam(0.001), loss='mean_squared_error')
    model.fit(X_train, y_train, epochs=50, batch_size=64)
    y_pred = model.predict(X_test)
    r2 = r2_score(y_test, y_pred.round())
    return r2
from pyswarm import pso
import keras
lb = [8,8,8]
ub = [32,32,32]
best_params, _ = pso(pso_optimization,lb,ub,swarmsize=10,maxiter=5)
neurons_layer1, neurons_layer2, neurons_layer3 = best_params
from keras.layers import GRU
model_gru_pso = Sequential()
```

```
model_gru_pso.add(GRU(units=int(neurons_layer1), activation='relu',
input_shape=(X_train.shape[1], 1)))

model_gru_pso.add(Dense(units=int(neurons_layer2), activation='relu'))

model_gru_pso.add(Dense(units=int(neurons_layer2), activation='relu'))

model_gru_pso.add(Dense(1))

optimizer_gru_pso = Adam(lr=0.001)

model_gru_pso.compile(optimizer=optimizer_gru_pso, loss='mean_squared_error')

history_gru_pso = model_gru_pso.fit(X_train, y_train, epochs=50, batch_size=64,
validation_split=0.2)

y_pred_gru_pso = model_gru_pso.predict(X_test)

mse_gru_pso = mean_squared_error(y_test, y_pred_gru_pso)

mae_gru_pso = mean_absolute_error(y_test, y_pred_gru_pso)

r2_gru_pso = r2_score(y_test, y_pred_gru_pso) * 100

nse_gru_pso = nse(y_pred_gru_pso, y_test)

nse_gru_pso = "{:.1e}".format(nse_gru_pso)

mbe_gru_pso = mbe(y_pred_gru_pso, y_test)

rmse_gru_pso = np.sqrt(mean_squared_error(y_test, y_pred_gru_pso))

metrics_gru_pso = [mae_gru_pso,nse_gru_pso,mbe_gru_pso,r2_gru_pso,rmse_gru_pso]

model_gru_pso.summary()

plt.plot(history_gru_pso.history['loss'], label='Training Loss')

plt.plot(history_gru_pso.history['val_loss'], label='Validation Loss')

plt.title('Training and Validation Loss')

plt.xlabel('Epoch')

plt.ylabel('Loss')

plt.legend()

plt.show()

plt.plot(history_gru_pso.history['loss'])

plt.title('Mean Squared Error vs Epochs')

plt.xlabel('Epoch')

plt.ylabel('Mean Squared Error')

plt.legend()
```

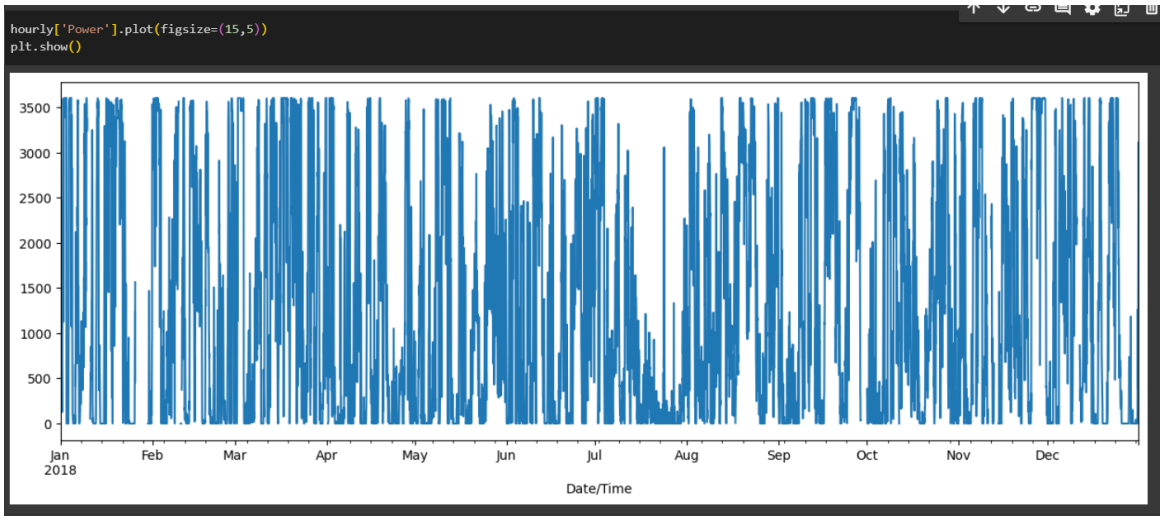
```
plt.show()

table_metrics_gru_pso = pd.DataFrame(metrics_gru_pso)

table_metrics_gru_pso.index = ["MAE","NSE","MBE","R2","RMSE"]

table_metrics_gru_pso.columns = ["PSO GRU"]

table_metrics_gru_pso

plt.figure(figsize=(7,5))

plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='black',
linestyle='solid', lw=3, label='Actual')

plt.scatter(y_test, y_pred_gru_pso, color='dodgerblue', marker='o', label='Predicted')

plt.title('PSO GRU MODEL')

plt.legend()

plt.show()

Metric =
pd.DataFrame(columns=["MAE","NSE","MBE","R2","RMSE"],index=['DNN','KNN','L
STM','Averaging','RF Regressor','Bagging Regressor','Gradient Boosting Regrssor','PSO
LSTM','GRU','PSO
GRU'],data=[metrics_dnn,metrics_knn,metrics_lstm,metrics_avg,metrics_rf,metrics_br,m
etrics_gbr,metrics_lstm_pso,metrics_gru,metrics_gru_pso])

Metric
```

CHAPTER 4

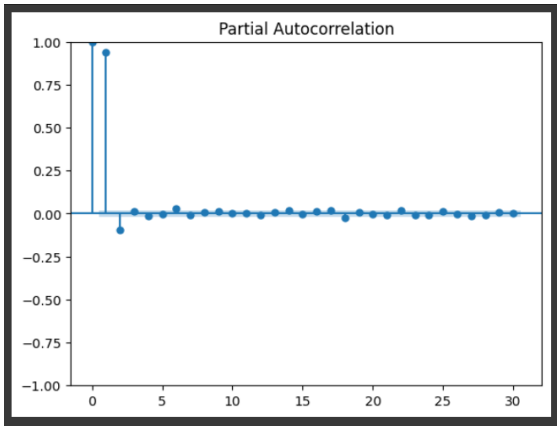
SNAPSHOTS

EDA:

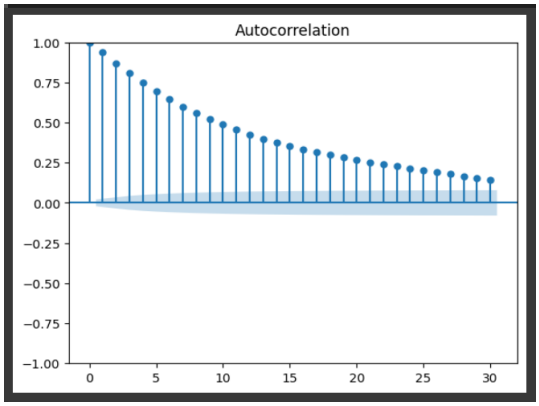
- **Visualization of power on hourly basis**



- **Autocorrelation:**



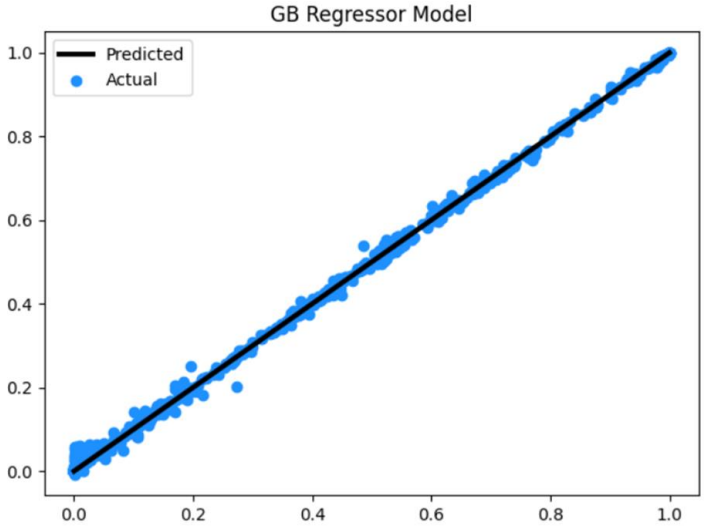
- **Partial Autocorrelation:**



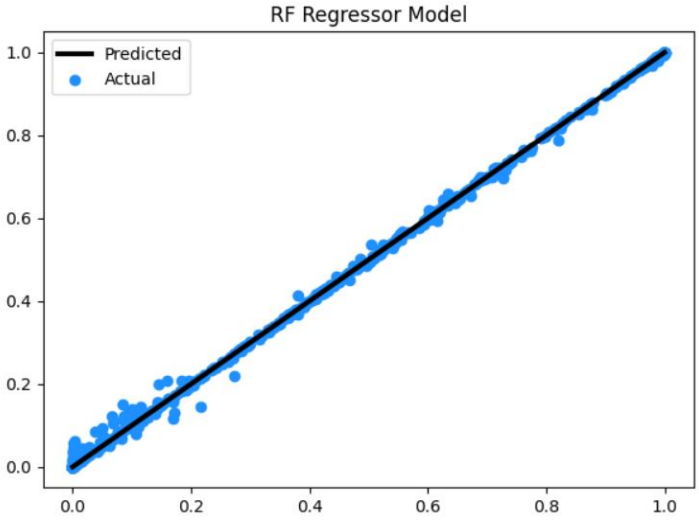
	Count	Mean	Std	Min	25%	50%	75%	Max
Power	50530	1307.68	1312.45	2.4714	50.677	825.83	2482.50	3618.73
Wind_speed	50530	7.55795	4.22716	0	4.2013	7.1045	10.3000	25.2060
Theor_power	50530	1492.17	1368.01	0	161.32	1063.77	2964.97	3600
Wind_dir	50530	123.687	93.4437	0	49.315	73.7129	201.696	359.997

Table 4.1 Statistical analysis of the wind dataset features

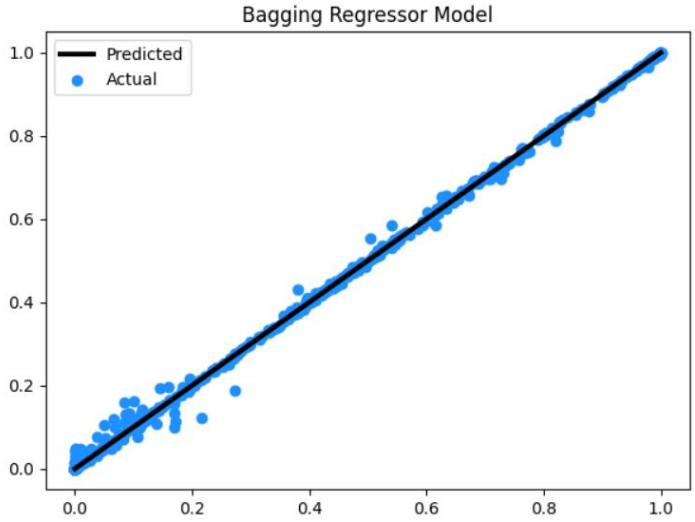
Gradient Boosting Regressor:



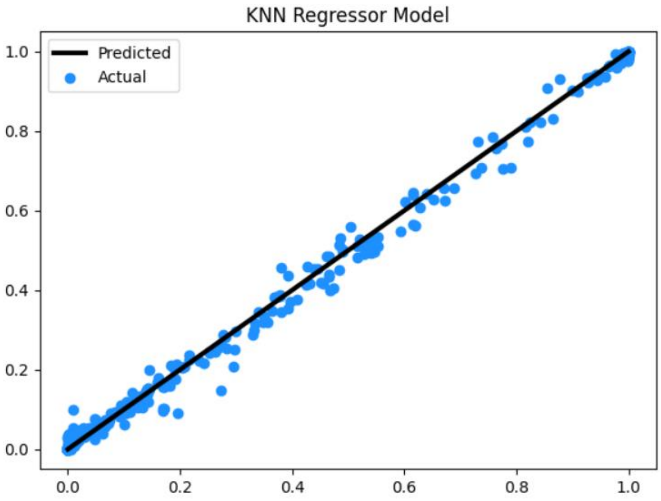
Random Forest Regressor:



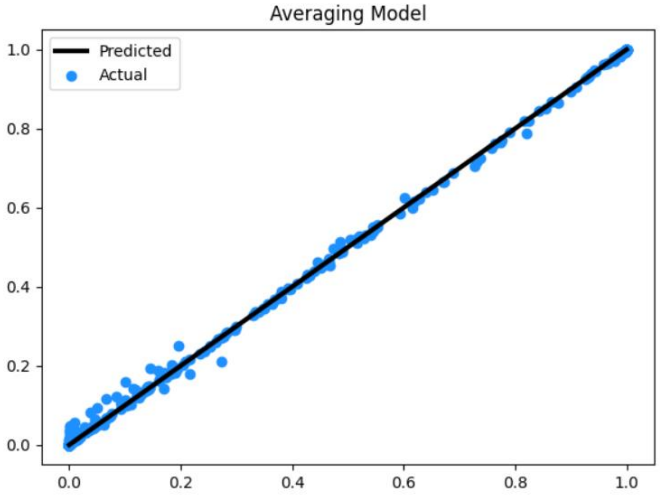
Bagging Regressor:



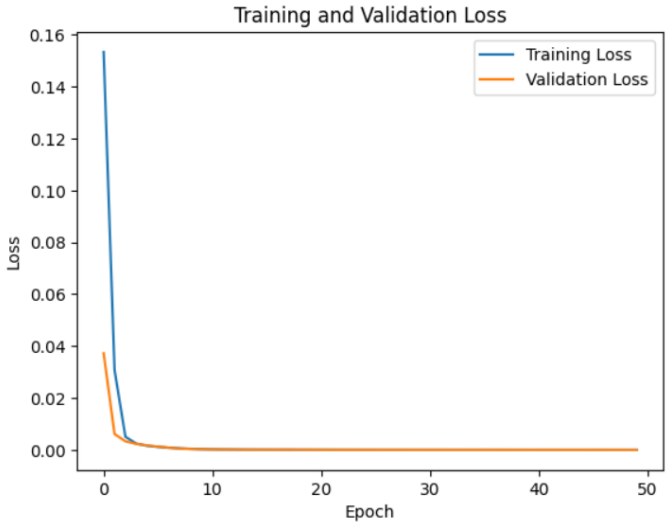
KNN Regressor:

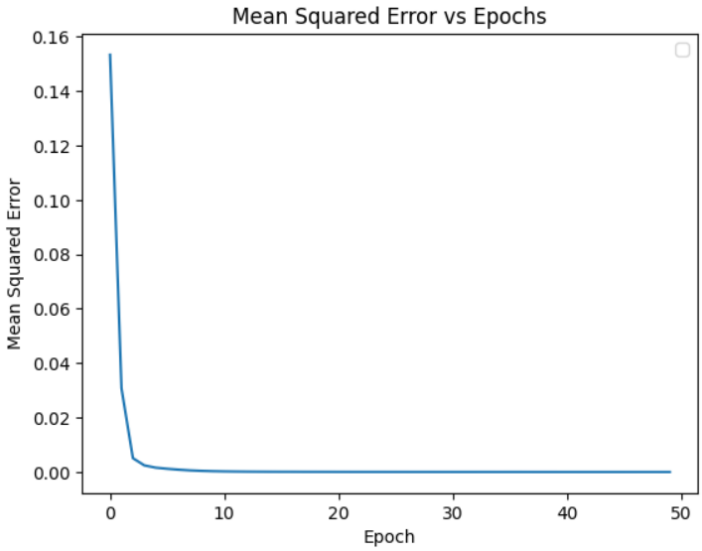


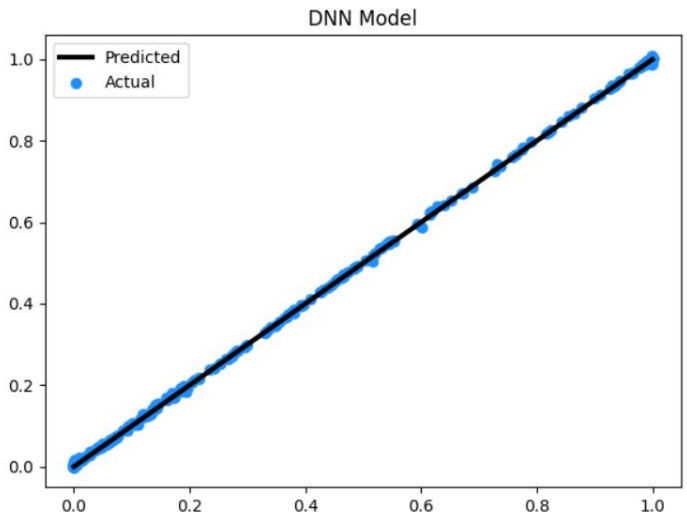
Averaging Model:



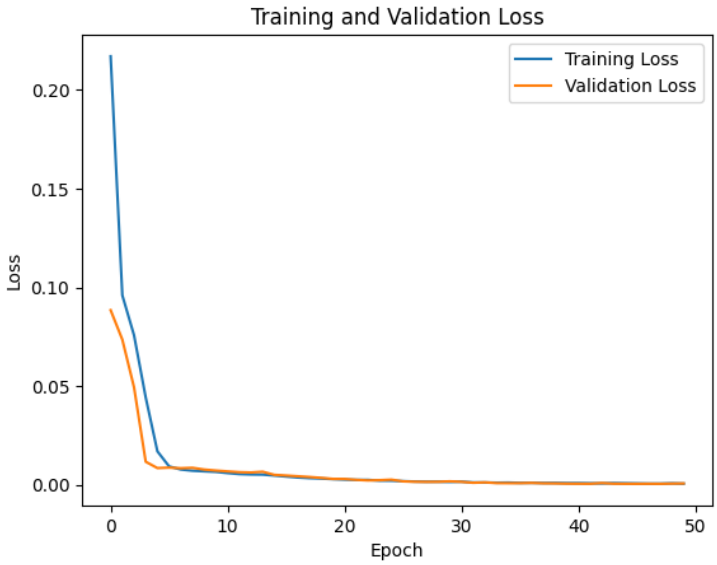
Deep Neural Network:

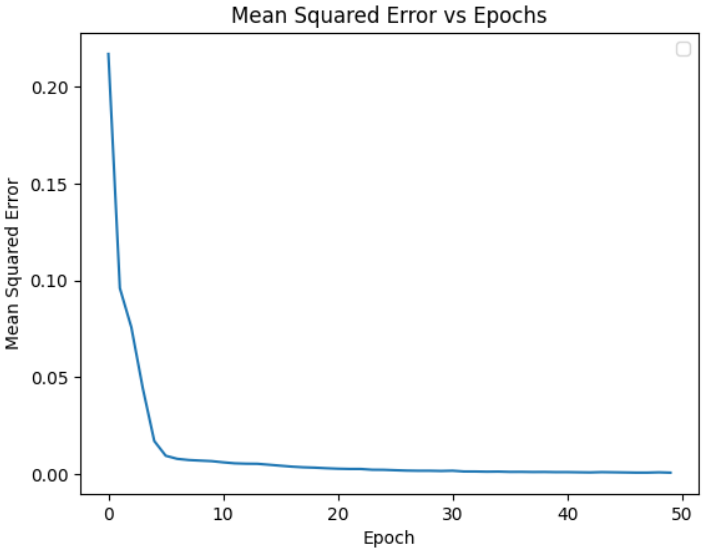


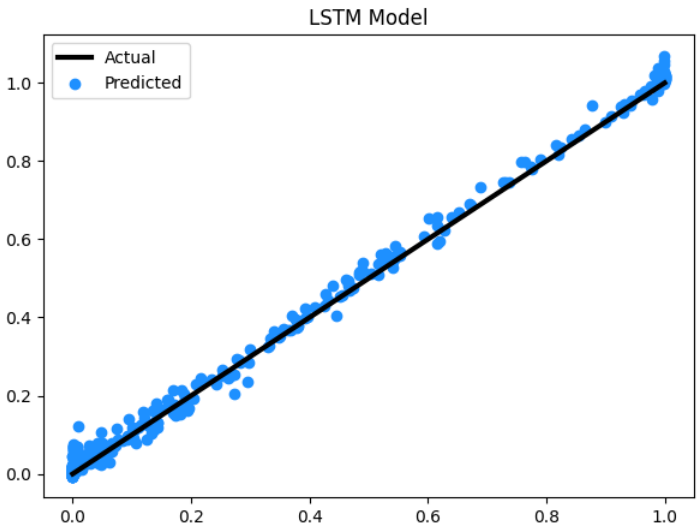




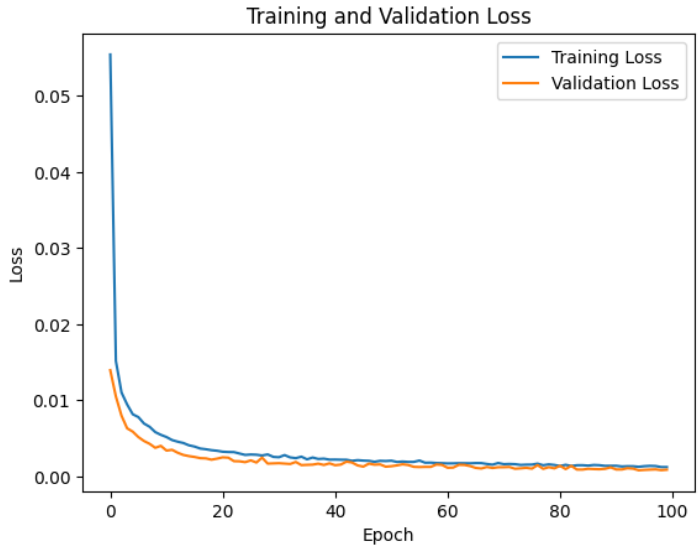
Long Short Term Memory LSTM:

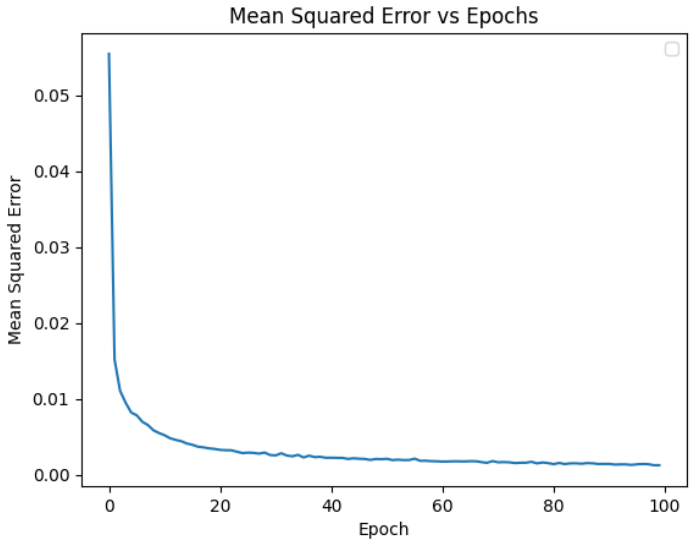


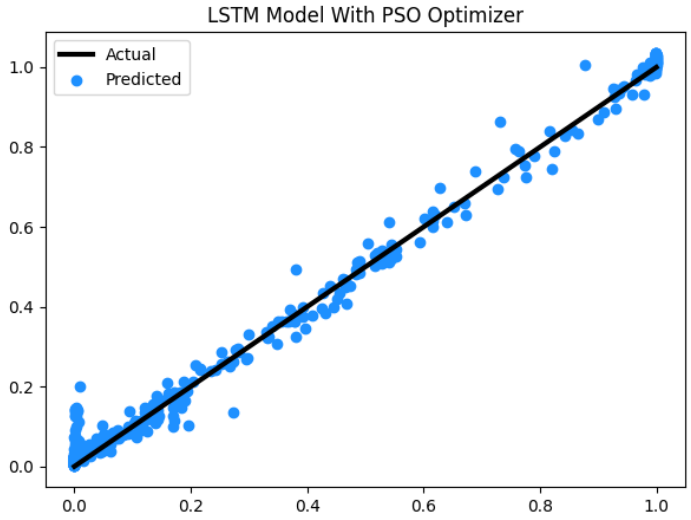




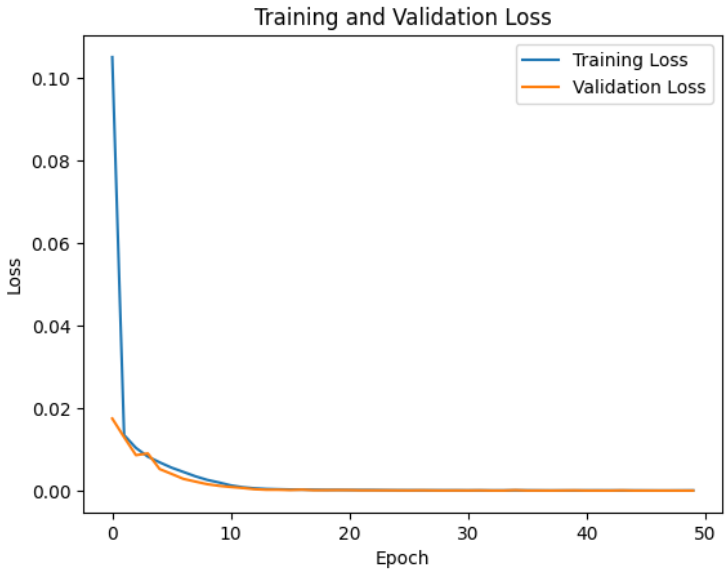
Long Short Term Memory (LSTM) with PSO:

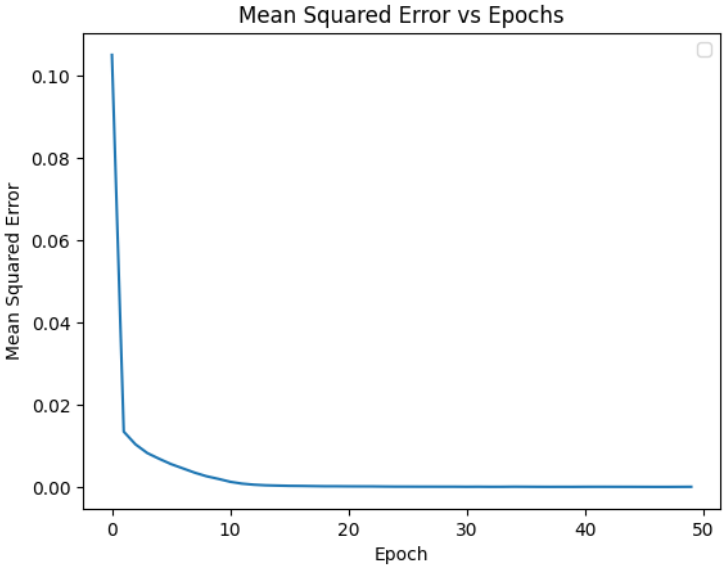


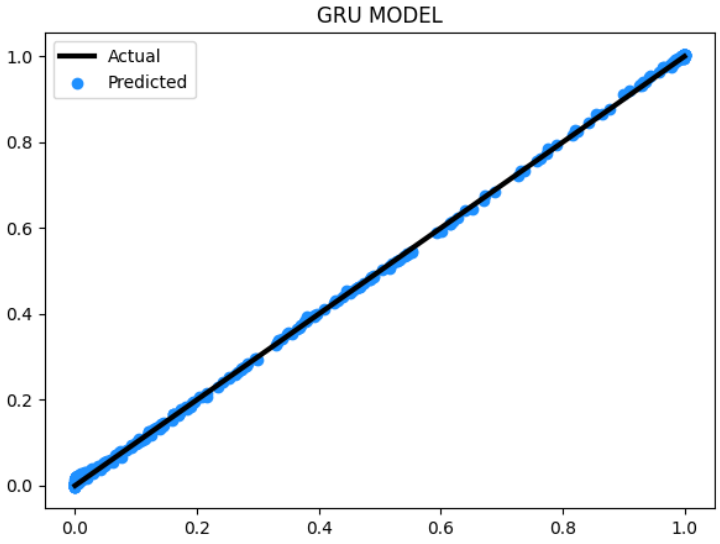




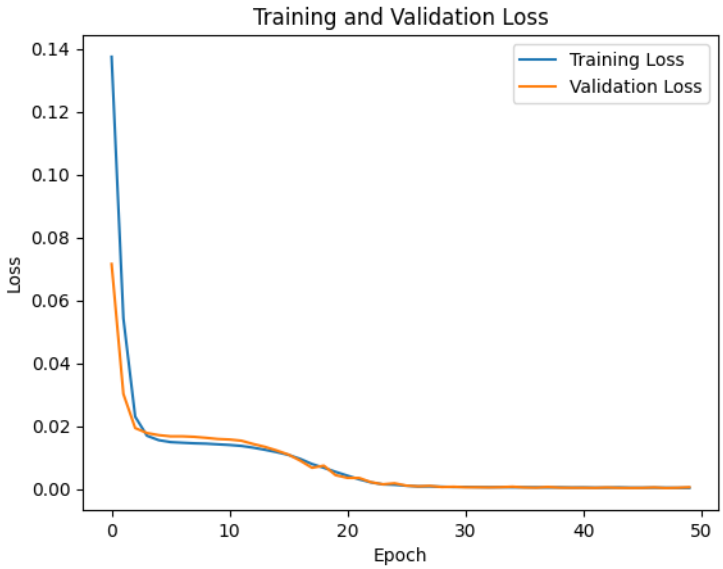
GRU Model:

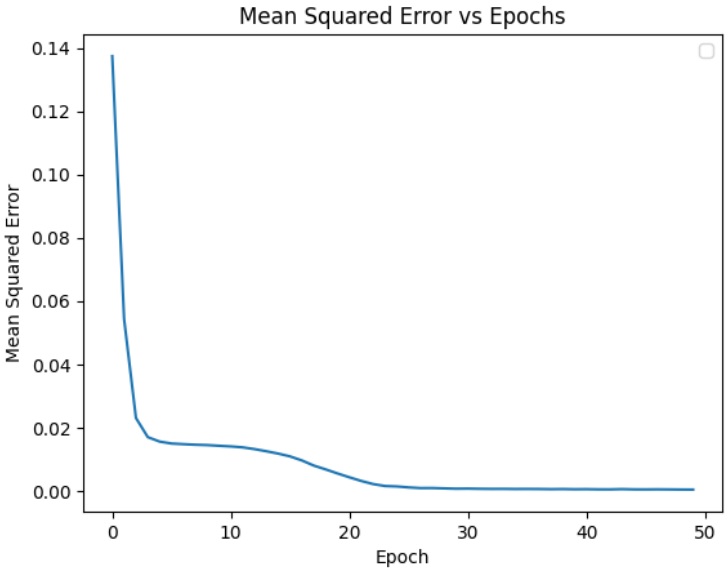

Training and Validation Loss
Training Loss
Validation Loss
Loss
Epoch
0.00
0.02
0.04
0.06
0.08
0.10
0
10
20
30
40
50

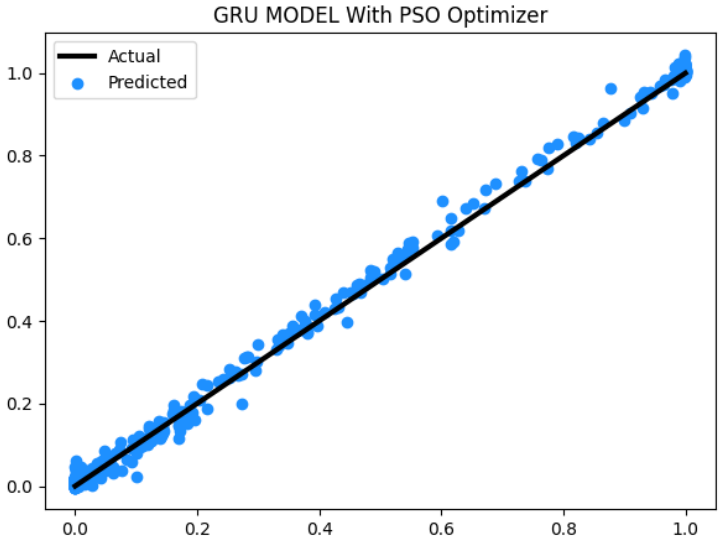

Mean Squared Error vs Epochs
Mean Squared Error
Epoch
0.00
0.02
0.04
0.06
0.08
0.10
0
10
20
30
40
50


GRU MODEL
Actual
Predicted
0.0
0.2
0.4
0.6
0.8
1.0

GRU with PSO Model:







CHAPTER 5

CONCLUSION AND FUTURE PLANS:

Final Results:

	MAE	NSE	MBE	R2	RMSE
DNN	0.004578	8.2e+02	0.001442	99.966386	0.006695
KNN	0.012914	3.7e-03	-0.002184	99.634346	0.022083
LSTM	0.010925	8.1e+02	0.001764	99.820728	0.015462
Averaging	0.005621	1.1e-03	0.002734	99.892023	0.012000
RF Regressor	0.005527	1.1e-03	0.002677	99.887388	0.012255
Bagging Regressor	0.006836	1.8e-03	0.002635	99.824385	0.015304
Gradient Boosting Regrssor	0.006915	1.5e-03	0.002889	99.854969	0.013908
PSO LSTM	0.023857	8.1e+02	0.012735	99.017435	0.036199
GRU	0.004112	8.2e+02	0.002147	99.974307	0.005854
PSO GRU	0.014285	8.2e+02	0.008244	99.709641	0.019678

The project's goal is to combine deep learning and machine learning techniques to estimate wind power more accurately. Experiments were carried out to evaluate the performance of the suggested strategy and compare its outcomes with those of six other machine learning models in order to demonstrate its viability. Four assessment criteria are utilized to determine the project's success. Plots of analysis created from the acquired data further demonstrated the stability and efficacy of this approach. Comparisons with various models make it evident that the Gated Neural Network (GRU) offers a higher accuracy.

Future research will concentrate on using Explainable AI techniques to improve the suggested method's explainability. Through the use of these methods, the inner workings of the Gated Neural Network (GRU) model will be made clear, revealing how it makes its predictions. Stakeholders will have a better knowledge of the variables affecting wind power estimation if the model's decision-making process is made clear and comprehensible. Furthermore, an evaluation of the model's resilience in a variety of prediction tasks and datasets will be conducted in order to guarantee its practicality.

REFERENCES

[1] Tarek, Z., Shams, M. Y., Elshewey, A. M., El-kenawy, E. S. M., Ibrahim, A., Abdelhamid, A. A., & El-dosuky, M. A. (2023). Wind Power Prediction Based on Machine Learning and Deep Learning Models. Computers, Materials & Continua, 75(1).

[2] Z. Lin, X. Liu and M. Collu, "Wind power prediction based on high-frequency SCADA data along with isolation forest and deep learning neural networks," International Journal of Electrical Power & Energy Systems, vol. 118, no. 105835, pp. 1–10, 2020.

[3] T. Ahmad and D. Zhang, "A data-driven deep sequence-to-sequence long-short memory method along with a gated recurrent neural network for wind power forecasting," Energy, vol. 239, no. 122109, pp. 1–20, 2022.

[4] Kisvari, A., Lin, Z., & Liu, X. (2021). Wind power forecasting–A data-driven method along with gated recurrent neural network. Renewable Energy, 163, 1895-1909.

[5] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

[6] Peng, Z., Peng, S., Fu, L., Lu, B., Tang, J., Wang, K., & Li, W. (2020). A novel deep learning ensemble model with data denoising for short-term wind speed forecasting. Energy Conversion and Management, 207, 112524.

[7] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

[8] A. Alkesaiberi, F. Harrou and Y. Sun, "Efficient wind power prediction using machine learning methods:A comparative study," Energies, vol. 15, no. 2327, pp. 1–24, 2022.

[9] B. Liu, S. Zhao,X.Yu, L. Zhang and Q.Wang, "Anovel deep learning approach for wind power forecastingbased on WD-LSTM model," Energies, vol. 13, no. 4964, pp. 1–17, 2020.

[10] J. Zhang, J. Yan, D. Infield, Y. Liu and F. Lien, "Short-term forecasting and uncertainty analysis of wind turbine power based on long short-term memory network and Gaussian mixture model," Applied Energy,vol. 241, pp. 229–244, 2019.

[11] M. Qolipour, A.Mostafaeipour,M. Saidi-Mehrabad and H. R. Arabnia, "Prediction of wind speed using a new Grey-extreme learning machine hybrid algorithm: A case study," Energy & Environment, vol. 30, no.1, pp. 44–62, 2019.

[12] D. Karamichailidou, V. Kaloutsa and A. Alexandridis, "Wind turbine power curve modeling using radial basis function neural networks and tabu search," Renewable Energy, vol. 163, pp. 2137–2152, 2021.

[13] H. Demolli, A. S. Dokuz, A. Ecemis andM. Gokcek, "Wind power forecasting based on daily wind speed data using machine learning algorithms," Energy Conversion and Management, vol. 198, no. 111823, pp.1–12, 2019.

[14] R. Yu, J. Gao, M. Yu,W. Lu, T. Xu et al., "LSTM-EFG for wind power forecasting based on sequential correlation features," Future Generation Computer Systems, vol. 93, pp. 33–42, 2019.

.

APPENDIX-BASE PAPER

Computers, Materials & Continua
DOI: 10.32604/cmc.2023.032533
Article

Tech Science Press

Wind Power Prediction Based on Machine Learning and Deep Learning Models

Zahraa Tarek[1], Mahmoud Y. Shams[2,*], Ahmed M. Elshewey[3], El-Sayed M. El-kenawy[4,5], Abdelhameed Ibrahim[6], Abdelaziz A. Abdelhamid[7,8] and Mohamed A. El-dosuky[1,9]

[1]Faculty of Computers and Information, Computer Science Department, Mansoura University, Mansoura, 35561, Egypt
[2]Faculty of Artificial Intelligence, Kafrelsheikh University, Kafrelsheikh, 33511, Egypt
[3]Faculty of Computers and Information, Computer Science Department, Suez University, Suez, Egypt
[4]Department of Communications and Electronics, Delta Higher Institute of Engineering and Technology, Mansoura, 35111, Egypt
[5]Faculty of Artificial Intelligence, Delta University for Science and Technology, Mansoura, 35712, Egypt
[6]Computer Engineering and Control Systems Department, Faculty of Engineering, Mansoura University, Mansoura, 35516, Egypt
[7]Department of Computer Science, Faculty of Computer and Information Sciences, Ain Shams University, Cairo, 11566, Egypt
[8]Department of Computer Science, College of Computing and Information Technology, Shaqra University, 11961, Saudi Arabia
[9]Department of Computer Science, Arab East Colleges, Riyadh, 13544, Saudi Arabia
*Corresponding Author: Mahmoud Y. Shams. Email: mahmoud.yasin@ai.kfs.edu.eg
Received: 21 May 2022; Accepted: 23 June 2022

Abstract: Wind power is one of the sustainable ways to generate renewable energy. In recent years, some countries have set renewables to meet future energy needs, with the primary goal of reducing emissions and promoting sustainable growth, primarily the use of wind and solar power. To achieve the prediction of wind power generation, several deep and machine learning models are constructed in this article as base models. These regression models are Deep neural network (DNN), k-nearest neighbor (KNN) regressor, long short-term memory (LSTM), averaging model, random forest (RF) regressor, bagging regressor, and gradient boosting (GB) regressor. In addition, data cleaning and data preprocessing were performed to the data. The dataset used in this study includes 4 features and 50530 instances. To accurately predict the wind power values, we propose in this paper a new optimization technique based on stochastic fractal search and particle swarm optimization (SFS-PSO) to optimize the parameters of LSTM network. Five evaluation criteria were utilized to estimate the efficiency of the regression models, namely, mean absolute error (MAE), Nash Sutcliffe Efficiency (NSE), mean square error (MSE), coefficient of determination (R^2), root mean squared error (RMSE). The experimental results illustrated that the proposed optimization of LSTM using SFS-PSO model achieved the best results with R^2 equals 99.99% in predicting the wind power values.

Keywords: Prediction of wind power; data preprocessing; performance evaluation

1 Introduction

Wind energy is crucial to reducing global carbon emissions. Wind turbine producers' power curves are an efficacious way of showing wind turbine overall effectiveness. Nevertheless, wind power prediction is difficult because of the complexities of offshore wind turbine dynamics and the surrounding cruel environment but prediction is necessary to permit condition monitoring [1]. Wind energy production has a significant influence not just on power markets, but also on retail market and wholesale layouts. Jointly, technological challenges occur as an output of the requirement to assure the power grid's proper operation. High-quality, long-term wind data values are necessary to create technique findings which resulting in good policy suggestions. Middle and long-term prediction requirements (e.g., reliability, thoroughness and rapidity) are more stringent, making it challenging to generate trustworthy findings [2].

Wind forecasting technologies have been the focus of global research, serving as the foundation for power system planning and operation, power dispatch reference and optimum energy flow distribution. With the advancement of information, artificial intelligence technologies and edge computing devices, forecasting approach is evolving toward surveillance refinement. The real-time forecasting of wind power can help wind turbines improve their overall productivity by enabling sophisticated wind turbine adjustment planning and pre-setting of pitch, yaw surveillance devices [3]. Wind power prediction is important for power traders' unit obligation, maintenance scheduling and profit optimization. The present progress of effective and precise wind power prediction methodologies provides an improved cost-effective operating and maintenance strategies for futuristic wind turbines [4].

AI techniques have showed great accuracy, enhanced generalization performance, and better learning capability, making them excellent for dealing with unreliable, inflexible, and discontinuous wind power. Because of its high thoroughness, versatility, and enhanced efficiency, AI-based hybrid techniques for wind power prediction such as deep learning, classification and regression, neural network and rule-based techniques have recently gained popularity [5]. Accurate wind power and wind speed (WP/WS) prediction has steadily become more important in reducing wind power variability in network deployment management. Because of their greater capacity to cope with complicated nonlinear issues, data uncertainties, and missing features, deep learning techniques are widely being evaluated for WP/WS prediction as intelligent methodologies, particularly deep learning [6].

Wind power forecasting has evolved through the years into a method of addressing the high fluctuation issues produced by large-scale integration. Improved forecast accuracy is essential for improving power grid reliability and economics. Machine learning techniques have been used to improve wind power forecasting. Bagging Neural Networks, Adaptive Boosting, Gradient Boosting and Random Forest. Machines are some of the methods utilized for prediction. Ensembles increase predicting accuracy by diversifying the model and are appropriate for middle and long-term prediction. Optimization algorithms, signal decomposition techniques and several of mentioned algorithms are integrated into hybrid models, which encompass the integration of both unsupervised and supervised algorithms employing clustering models such as Spectral and Bayesian, and clustering analysis is performed using K-means [7]. Aside from traditional machine learning techniques, the effective employment of deep learning in object classification have attracted an increasing number of academics and experts in the field of prediction. Given deep learning's distinct advantages in

time-dependent depictions and feature extraction, it is appropriate to integrate it with auxiliary mode decomposition in forecasting of wind power [8]. Furthermore, when a substantial quantity of hidden layers are employed in the building of a prediction models, deep learning techniques improve computational and statistical approaches, although they are attacked for their comparatively poor learning rates [9].

Wind power prediction relies heavily on assessment. Evaluating suggested prediction models enables for ongoing comparison of alternative models and, as a result, their continuous evolution. Both probabilistic and deterministic forecasts require evaluation. Over the years, basic comparison measures like as Mean Squared Error (MSE), Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) have been employed in deterministic prediction to assess the effectiveness of prediction methods. Furthermore, assessing probabilistic projections is more difficult than assessing point predictions. While the variance between anticipated and measured power values is used to evaluate point predictions, this is not viable in probabilistic prediction since such a comparison is not achievable explicitly [10].

Our contributions can be stated as follows. (1) Data preprocessing and data cleaning were performed to the chosen dataset. (2) Deep neural network (DNN) model, k-nearest neighbor (KNN) regressor model, long short-term memory (LSTM) model, averaging model, random forest (RF) regressor model, bagging regressor model, and gradient boosting (GB) regressor model are used as regression models for predicting wind power. (3) Five evaluation criteria namely, mean absolute error (MAE), median absolute error (MedAE), mean square error (MSE), coefficient of determination (R^2), root mean squared error (RMSE) were utilized to estimate the efficiency of the regression models. Fig. 1 illustrates the framework of the recommended methodology for wind power prediction.

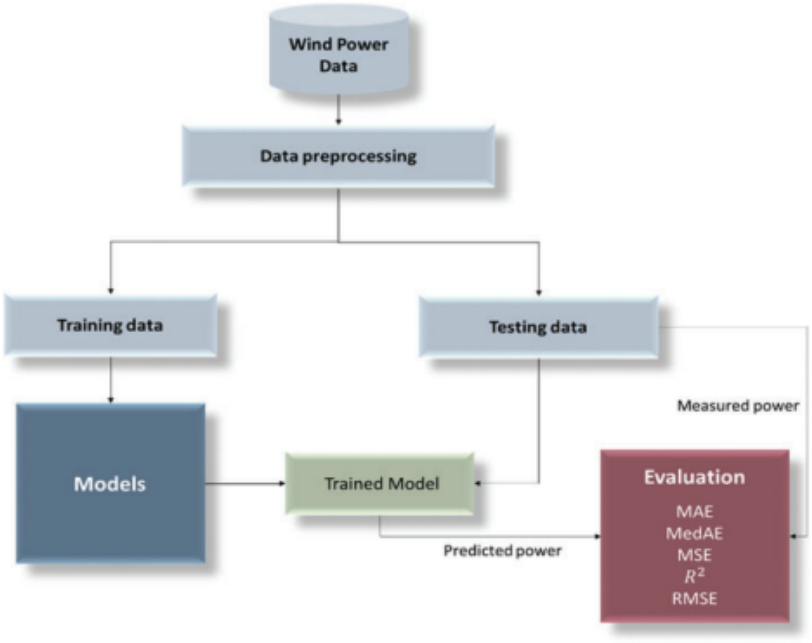


Figure 1: Framework for the proposed wind power prediction

The rest of the paper is organized as follows. The categorization of existing wind power forecasting systems was presented in Section 2. Section 3 discussed the different ML/ DL strategies that may be utilized to increase wind power forecasting performance. Section 4 presented the proposed model for wind power prediction. Performance evaluation and results discussion were provided in Section 5. At last, conclusion and future work are mentioned in Section 6.

2 Related Works

Machine learning approaches are currently utilized in many applications to classify and predict the enrolled features based on supervised learning [11]. Furthermore, its ability to cluster the data and reduce the dimensionality size of the features based on unsupervised learning. Deng et al. [12] used a bidirectional gated recurrent unit approach with deep learning system to anticipate wind power. The findings demonstrate the model's capacity to spontaneously simulate the link between wind direction, speed and power. Yildiz et al. [13] presented a Convolutional Long Short-Term Memory network (Conv-LSTM) for short-term wind power prediction. To remove any irregularities in the raw data, Variational Mode Decomposition (VMD) was applied. Following that, Conv-LSTM is used to generate early forecasting findings, along with extract spatiotemporal data from the forecasted samples. In two separate studies, the suggested model outperformed the other models in terms of RMSE, MSE, MAE, and MRE error metrics. Authors in [14] suggested a two-step revolutionary deep learning technique for wind power forecasting: feature extraction based on VMD, followed by an enhanced CNN to predict wind power. Due to its competitive effectiveness, the suggested technique surpassed the other systems and provided impressive outcomes for extremely short-term prediction.

Authors in [15] suggested SVM-based approach with Improved Dragonfly Algorithm (IDA) for a hybrid wind power forecasting approach. When compared against existing models, such as the Gaussian Process Regression and Back Propagation Neural Network (BPNN), the suggested model outperformed them (GPR). The suggested IDA-SVM model outperformed the other techniques for winter and fall datasets using the R^2, NMAE, MAPE and NRMSE error metrics. Authors in [16] suggested a new deep transfer learning strategy based on a one-of-a-kind serio-parallel CL feature extractor for multi-step forward wind power forecasting of targeted wind ranches in the absence of wealthy historical information. The findings validated the supremacy of the proposed model over the independent LSTM and CNN techniques. Furthermore, the suggested CL-TL-CSO and CL-TL approaches outperformed the other non-transfer techniques irrespective of whether they were deep learning algorithms (i.e., LSTM, CNN, etc.) or shallow-layer network techniques (i.e., ELM and Elman).

Authors in [17] suggested four ML techniques, namely, SVR, ANN, RF and regression trees for wind power forecasting. The findings indicated that using a single metric that takes into account both training time and performance, the SVR might be the optimum choice. Authors in [18] utilized a unique STCM based on CNN-LSTM for ultra-short-term wind power prediction. The efficacy and supremacy of the suggested CNN-LSTM technique were demonstrated by comparing four assessment criteria with LSTM and CNN employed separately. The experiment findings demonstrate that the overall model's mean MAE, MAPE, RMSE, and NRMSE reduce by 33.77 percent, 30.69 percent, 25.3 percent, and 23.3 percent (compared to CNN), and 12.0 percent, 10.6 percent, 14 percent, and 12.7 percent (compared to LSTM). The suggested STCM for multi-step prediction based on CNN-LSTM completely addressed the spatio-temporal correlation of meteorological parameters across the wind farm and can predict wind farm power more correctly.

Authors in [19] suggested a hybrid wind power projection technique based on ELM and KMPE. The results indicated that the suggested ELM-KMPE strategy outperformed the traditional FFBPNN approach. Nonetheless, because to the fixed-point phase, ELM-KMPE demonstrated a slower estimating speed. As a result, more investigation is necessary to address the computation complexity difficulties. Authors in [20] introduced a parameter optimization mechanism into the framework of back-propagation neural networks (BPNN), where layers are merely piled in depth, to improve training effectiveness. Authors in [21] demonstrated a hybrid technique for predicting wind power production for twenty-four hours in advance. This unique technique is built on CNN with a Radial Basis Function Neural Network (RBFNN). The simulation outcomes show that the suggested technique was more effective than standard methods for estimating wind power 24 h in advance. Authors in [22] introduced a novel effective hybrid prediction model integrating variational mode decomposition (VMD) and mixed Kernel ELM (MKELM) for accurate wind energy forecasting. According to the results, the suggested model produced the best effective predictive results with the lowest SMAPE, MAE and RMSE values. A recent study presented by Alkesaiberi et al. [23] presented a comparative study of wind power prediction using machine learning methods. The models presented in [23–30] describes the algorithms and approaches to predict the wind power based on different evaluation matrices as presented in Tab. 1.

Table 1: Some of ML/DL for wind power prediction

Paper	Technique	Criterion	Value	Conclusion
[1]	Deep Learning Neural Networks (DNN)-Isolate Forest (IF) and	MSE	0.003	IF (for outlier identification) was a more successful approach, when the input characteristics could not be expected to be Gaussian.
[23]	Ensemble learning (ES)- Gaussian process regression (GPR) models	R^2	0.95	The findings demonstrate the value of considering the input factors and lag data for forecasting wind power. The improved GPR and ensemble algorithms also surpassed the other machine learning techniques, according to the results.
[24]	LSTM- Wavelet Decomposition (WD)	MAPE	5.831	When compared to a single prediction model and machine learning, the approach can more correctly anticipate wind power generation all over the state.
[25]	Gaussian Mixture Model (GMM), Long Short-Term Memory Network (LSTM)	RMSE	6.37	The LSTM approach was demonstrated to be more accurate and have a quicker convergence rate than the other techniques. Furthermore, the GMM technique outperformed and was evaluated superior to the other approaches.

(Continued)

Table 1: Continued

Paper	Technique	Criterion	Value	Conclusion
[26]	Grey model, Extreme Learning Machine (ELM)	MSE R^2	0.000376 0.99376	The findings reveal that the suggested approach outperformed the standard extreme learning machine technique in predicting wind speed in the research region.
[27]	Tabu Search- Non Symmetric Fuzzy (TS-NSFM)	MAE	(Feb.) 18.995 (July) 18.932	In terms of modelling accuracy and efficiency, the suggested algorithm's answer exceeds the results generated by its competitors.
[28]	xGBoost, SVR, and RF algorithms	R^2 MAE	0.995 7.048	RF is the most efficient technique for predicting long-term total daily wind power.
[29]	LSTM- Enhanced Forget-Gate (EFG)	MSE	5.5311	LSTM-EFG can better estimate wind power at a given moment, relieving the burden of surge and, hesitation control in the power system and making maximum use of wind power.

According to the previous table, numerous strategies are routinely employed to forecast wind power, although with limited model enhancement and data preprocessing capabilities. This shortage of experience is a substantial problem for authoritative and consistent wind energy forecasts. As a result, in comparison to the prior research discussed in this article, we suggest an approach to attain the best performance of the wind power prediction system.

3 Proposed Methodology

A Deep Neural Network (DNN) is essentially an ANN with several hidden layers. The approach is believed to be especially useful when working with huge data samples. It is built on a feedforward multilayer network. A backpropagation method is used in the learning phase to change neuron weights in order to decrease training error. The knowledge between layers is transferred through a nonlinear activation function modification. Multiple layers of nonlinearity result in improved feature extraction and info gain [30]. Deep learning techniques are very deep structures based on successive layers of impersonation and abstraction [31]. Fig. 2 presents the deep network architecture of DNN with multiple layers network.

In this work, we utilized DNN architecture with three basic layers which are the input layers that represents the features selected from the applied wind dataset. The hidden layers which include five stacked hidden layers each with five neurons in each layers with an ReLU activation function and Adam optimizer. We utilized 50 epochs as a number of iterations with a patch size 64 and the learning rate 0.0001. The proposed scheme is a hybrid model of type sequential hybrid means the first paradigm is the LSTM that passes its output to the second paradigm SFS-PSO to optimize the results and to boost the evaluation criteria results.

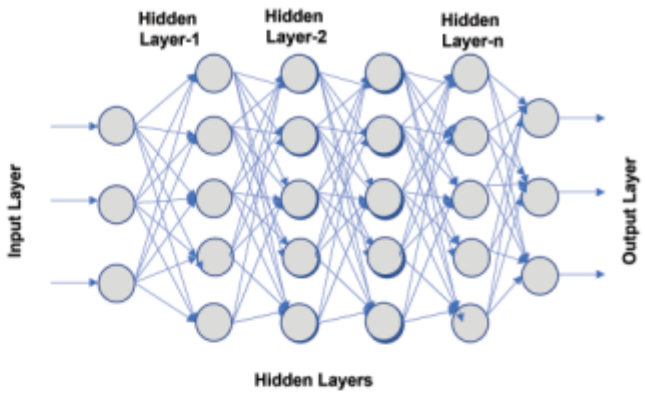


Figure 2: Deep network architecture with multiple layers

3.1 Long Short-Term Memory

Long Short-Term Memory (LSTM) is a type of recursive neural network (RNN) that may learn to depend on input for an extended period of time. Furthermore, it is appropriate for analyzing and anticipating necessary events in time series with relatively lengthy gaps and delays. LSTM has had significant success and is extensively utilized on a variety of conditions [29]. The LSTM presents a memory unit based on RNN that is regulated by input, forgetting gates and output. It may improve the screening, storage, and information flow monitoring under the time feedback procedure, efficiency avoid risk of data loss, and address the gradient absence and explosion issue. The transformation equation is defined as in Eqs. (1) to (5) [29]:

$$f_t = \sigma(W_f[h_{t-1}; x_t] + b_f) \tag{1}$$

$$i_t = \sigma(W_i[h_{t-1}; x_t] + b_i) \tag{2}$$

$$o_t = \sigma(W_o[h_{t-1}; x_t] + b_o) \tag{3}$$

$$s_t = f_t \odot s_{t-1} + i_t \odot \tanh(W_s[h_{t-1}; x_t] + b_s) \tag{4}$$

$$h_t = o_t \odot \tanh(s_t) \tag{5}$$

where notation $[h_{t-1}; x_t]$ is the union of the previously concealed state with the current vector of input. $\odot$ represents the elementwise multiplication and σ denotes the logistic function defined elementwise by Eq. (6):

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{6}$$

The cell state s_t is the core innovation of LSTM. In comparison to the rapid shift in the concealed state h_t, The cell state can recall a more extensive past. The forget gate f_t controls its memory. Through an output gate, the hidden state h_t is acquired from s_t.

3.2 Particle Swarm Optimization

When it comes to the particle swarm optimization (PSO) method, it's inspired by the way flocks of birds move together [32]. For example, the social behavior of birds may be simulated using the PSO

algorithm Changes in velocity are used to guide the movement of swarms in their quest for food. These are the parameters of each particle in the PSO.

- The fitness function is used to evaluate the particles' current positions.
- Last best positions, which store better positions' values of the particles.
- Velocity or rate of position change.
- Position, which indicated a point in the search space.

All particles' locations and speeds change during the algorithm's rounds. The following commands are used to change the particle's location as in Eq. (7).

$$x_{t+1} = x_t + v_{t+1} \tag{7}$$

where x_{t+1} is the new particle position, and the updated velocity of each particle v_{t+1} can be calculated as in Eq. (8):

$$v_{t+1} = \omega v_t + C_1 r_1 (p_t - x_t) + C_2 r_2 (G - x_t) \tag{8}$$

where ω is the inertia weight, C_1 and C_2 represent cognition learning factor and the social learning factor. Parameter G is the global best position and r_1 and r_2 are random numbers in [0; 1].

3.3 Stochastic Fractal Search

An algorithm that uses random fractals as an inspiration for metaheuristic algorithms can be as efficient and accurate as the original fractal technique. To find a solution to a given problem, the Fractal Search (FS) technique follows three basic rules.

1. The best particles are kept while the others are removed with each new generation.
2. There are several ways to make and disperse new particles. In the new particles, the original particle energy is shared.
3. A particle can have electrical potential energy.

Finding fractals in any given object is possible with the use of the Stochastic Fractal Search (SFS), as explained in detail in [33]. The fractal-shaped objects are built using the Diffusion Limited Aggregation (DLA) process. When compared to the original FS approach, the SFS method makes use of diffusion and two distinct types of update processes. This solution BP might be surrounded by the solutions BP1-BP5 in order to arrive at the best potential outcome.

3.4 Diffusion Process

The Gaussian distribution approach is used in the DLA growth process to produce new particles based on the diffusion procedure of SFS. In the diffusion process, a list of walks created by the best solution $\overrightarrow{P}$ may be determined as Eq. (9).

$$\overrightarrow{P'} = Gaussian\left(\mu_{\overrightarrow{P}}, \sigma\right) + \left(\beta \times \overrightarrow{P} - \beta' \times \overrightarrow{V}\right) \tag{9}$$

where the updated best solution is denoted by $\overrightarrow{P'}$. The values of β and β' are selected randomly from the range [0; 1]. The point in the group is denoted by $\overrightarrow{V}$, whereas the position of the best point is referred to as $\overrightarrow{P}$. In addition, $\mu_{\overrightarrow{P}}$ is equal to $|\overrightarrow{P}|$ and σ is equal to $|\overrightarrow{V} - \overrightarrow{P}|$. A better solution can be found by using the application of SFS into the process of PSO, which is an exploration method based on the algorithm's diffusion process. The proposed algorithm is based on employing the SFS

to improve the exploration of the PSO optimization algorithm. The details of the proposed algorithm are presented in Algorithm 1.

3.5 The Proposed SFS-PSO Algorithm

To improve performance of LSTM network, we proposed a new optimization approach based on PSO and SFS to achieved better balance between the exploration and exploitation of the optimization process, the proposed algorithm employs the SFS technique to boost the performance of the exploitation step. The steps of the proposed algorithm are listed in Algorithm 1, and the balance between the exploration and exploitation is depicted in Fig. 3 depicts. In addition, the flowchart shown in Fig. 4 clarifies the steps of the proposed methodology.

Algorithm 1: The proposed SFS-PSO algorithm

Initialize the population particles $\overrightarrow{X_i}(i = 1, 2, 3, \ldots, n)$ with size n,
Fitness function Fn, and max iterations $iter\_max$.
Initialize the particles with random positions and velocities.
Initialize parameters ω, C_1, C_2, r_1, r_2, β, β'
Evaluate fitness function Fn for each $\overrightarrow{X_i}$
Find best individual $\overrightarrow{X_i^*}$
While t < $iter\_max$ **do**
  for ($i = 1$; $i \leq n$) **do**
    Update particle positions using:
    $x_{t+1} = x_t + v_{t+1}$
    Update particle velocities using:
    $v_{t+1} = \omega v_t + C_1 r_1 (p_t - x_t) + C_2 r_2 (G - x_t)$
  end for
  for ($i = 1$; $i \leq n$) **do**
    Apply Diffusion Process:
    $\overrightarrow{P'} = Gaussian(\mu_{\overrightarrow{P}}, \sigma) + (\beta \times \overrightarrow{P} - \beta' \times \overrightarrow{V})$
  end for
  Update parameters ω, C_1, C_2, r_1, r_2, β, β'
  Evaluate fitness function Fn for each $\overrightarrow{X_i}$
  Find best individual $\overrightarrow{X_i^*}$
  Set t = t + 1
end while
return $\overrightarrow{X_i^*}$

In this paper, we present Exploitation that described as a greedy strategy in which the proposed model use estimated value rather than real value to try to acquire greater rewards. As a result, we use this strategy to make the best decision possible based on current knowledge. Unlike exploitation strategies are more concerned with enhancing their understanding of each action than with receiving more rewards in order to reap long-term gains. Accordingly, this work depends on multi-feature learning model with greater local attention. At the same time, this model might gain more representative global and local properties. To extract more complete global characteristics, global features combine both the middle and final layers [34,35].

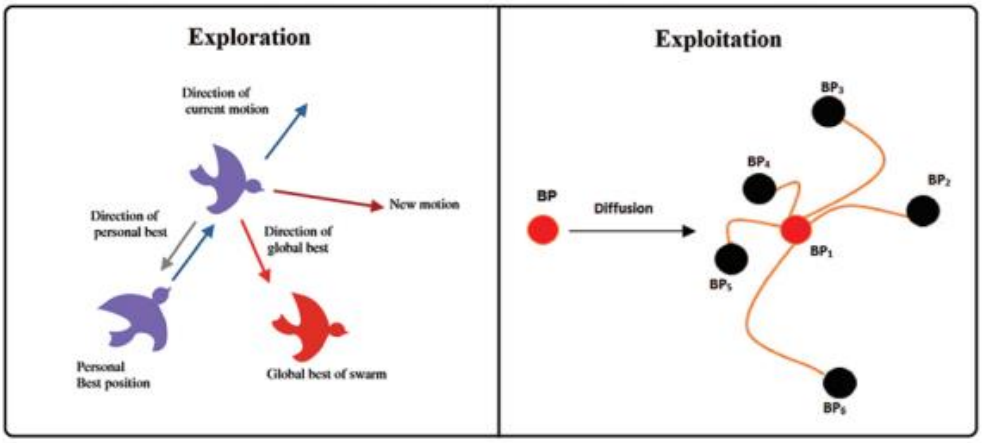


Figure 3: The balance between exploration and exploration tasks of the proposed method

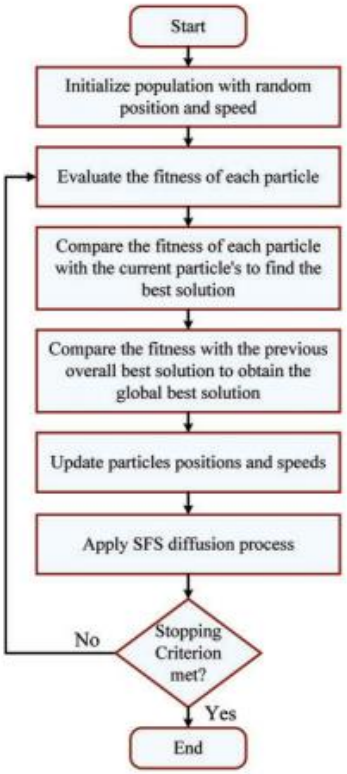


Figure 4: Flowchart of the proposed SFS-PSO optimization algorithm steps

4 Results

The results of the conducted experiments on the wind forecasting using the proposed approach along with the traditional baseline models are presented and discussed in this section. The section starts with presenting the employed dataset, followed by presenting the achieved results.

4.1 Dataset

As a case example, the studies use a wind power forecasting dataset to estimate hourly power output at seven wind farms for up to 48 h in advance. On Kaggle, the dataset is called Wind Turbine Scada Dataset linked in https://www.kaggle.com/datasets/berkerisen/wind-turbine-scada-dataset. Fig. 5 shows the correlation matrix of the features of this dataset. In addition, a statistical analysis of the dataset features is presented in Tab. 2, and the histogram of the wind power values is shown in Fig. 6.

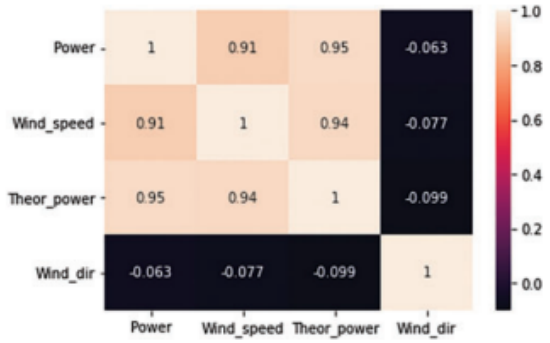


Figure 5: Correlation matrix of the wind dataset features

Table 2: Statistical analysis of the wind dataset features

	Count	Mean	Std	Min	25%	50%	75%	Max
Power	50530	1307.68	1312.45	2.4714	50.677	825.83	2482.50	3618.73
Wind_speed	50530	7.55795	4.22716	0	4.2013	7.1045	10.3000	25.2060
Theor_power	50530	1492.17	1368.01	0	161.32	1063.77	2964.97	3600
Wind_dir	50530	123.687	93.4437	0	49.315	73.7129	201.696	359.997

4.2 Evaluation Criteria

The evaluation of the proposed approach is performed in terms of the metrics presented in Tab. 3. These metrics are Nash Sutcliffe Efficiency (NSE), coefficient of determination (R^2), mean bias error (MBE), root mean error (RMSE) and mean absolute error (MAE) where N is the number of observations in the dataset; $\widehat{V}_n$ and V_n are the n^{th} predicted and actual wind power values, and $\overline{\widehat{V}_n}$ and $\overline{V_n}$ are the arithmetic means of the estimated and observed values Eqs. (10)–(14).

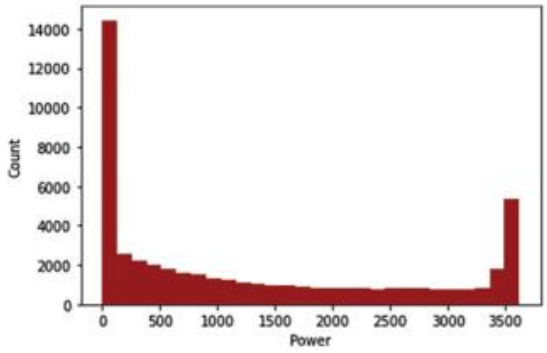


Figure 6: Histogram of the wind power based on the records of the wind dataset

Table 3: Performance evaluation metrics

Metric	Value	
RMSE	$\sqrt{\frac{1}{N}\sum_{n=1}^{N}\left[\widehat{V}_n - V_n\right]^2}$	(10)
MAE	$\frac{1}{N}\sum_{n=1}^{N}\left\|\widehat{V}_n - V_n\right\|$	(11)
MBE	$\frac{1}{N}\sum_{n=1}^{N}\left(\widehat{V}_n - V_n\right)$	(12)
R^2	$1 - \frac{\sum_{n=1}^{N}\left(V_n - \widehat{V}_n\right)^2}{\sum_{n=1}^{N}\left(\left(\sum_{n=1}^{N} V_n\right) - V_n\right)^2}$	(13)
NSE	$1 - \frac{\sum_{n=1}^{N}\left(V_n - \widehat{V}_n\right)^2}{\sum_{n=1}^{N}\left(V_n - \overline{\widehat{V}_n}\right)^2}$	(14)

4.3 Evaluation Results

In this section, the prediction results of the wind power is performed in terms of two experiments. Firstly, the prediction using the proposed optimized LSTM. Secondly, the prediction using a set of baseline models. These models include, deep neural network, K-nearest neighbor, average ensemble model, random forest, bagging regression model, and gradient boosting regression model. The discussion of the achieved results is presented in the next section.

Table 5: Parameters setting of the deep neural network

Batch size	Learning rate	Epochs	Optimizer	Activation function used in output	Activation function used in hidden
64	0.0001	50	Adam	Linear	ReLU

Table 6: Parameters setting of KNN regressor

Model	Parameters
KNN regressor	n_neighbors = 2, weights = distance

Table 7: Parameters setting of bagging regressor

Model	Parameters
Bagging regressor	n_estimators = 10, max_samples = 1

Table 8: Parameters setting of gradient boost regressor

Model	Parameters
GB regressor	n_estimators = 200, learning_rate = 0.1

On the other hand, the alignments of the predicted and actual values of the wind power using the five baseline models are shown in Fig. 8. Only 100 points of test samples are employed in this experiment. Although the alignment of the test points in this figure are properly fit a line, this alignment does not perform the same way when the number of points increased. This makes the proposed approach superior as it performs better in case of employing larger set of test points.

4.3.3 Results Comparison

A comparison between the results achieved by the proposed approach and the baseline models is presented in Tab. 9. As presented in the table, the proposed approach could achieve the best results among the models included in the conducted experiments. The MAE of the proposed approach is (0.000002), NSE is (1.2×10^{-7}), MBE is (0.00001), R^2 is (99.99%), and RMSE is (0.00002).

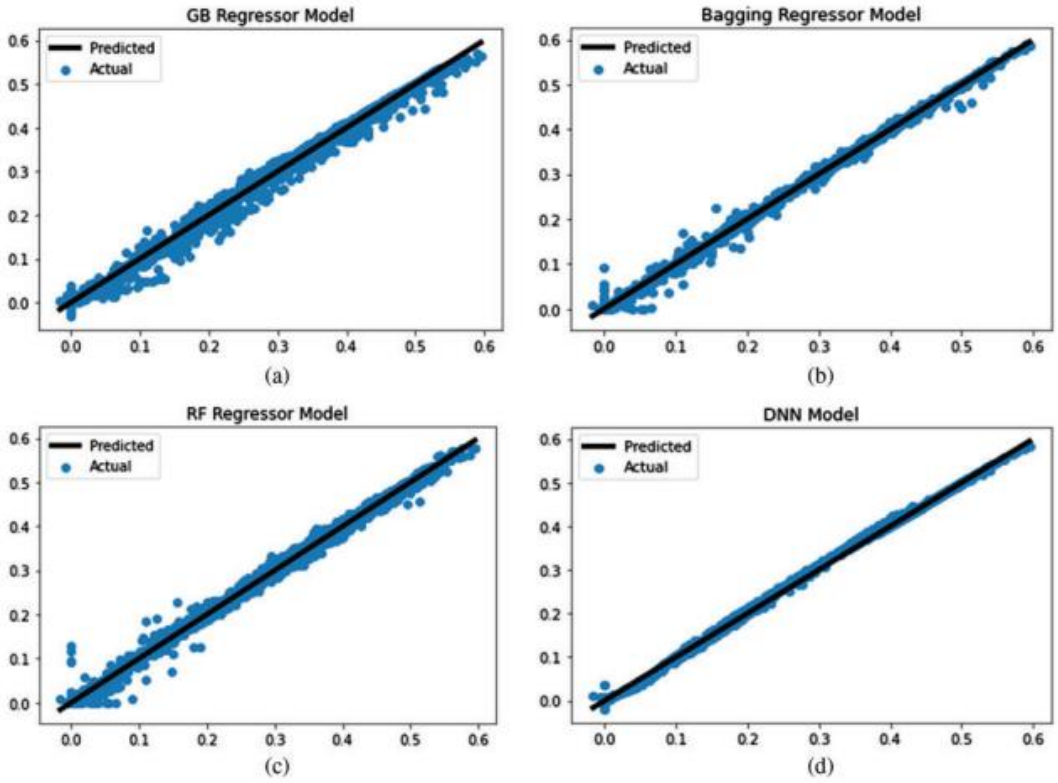


Figure 8: The predicted *vs.* actual values of wind power values with line fitting

Table 9: Evaluation criteria results using the proposed and other machine learning models

Model	MAE	NSE	MBE	R^2	RMSE
DNN	0.0021	1.42×10^{-5}	0.0008	99.96%	0.0037
KNN	0.0007	5.43×10^{-6}	0.0001	99.98%	0.0023
LSTM	0.0002	1.05×10^{-6}	0.0001	99.99%	0.0010
Averaging	0.002	4.38×10^{-5}	0.0008	99.90%	0.0070
RF regressor	0.003	3.65×10^{-5}	0.0016	99.90%	0.0060
Bagging regressor	0.001	1.3×10^{-5}	0.0002	99.96%	0.0040
Gradient Boost regressor	0.004	5.3×10^{-5}	0.0014	99.86%	0.0070
Proposed SFS-PSO LSTM	**0.000002**	$\mathbf{1.2 \times 10^{-7}}$	**0.00001**	**99.99%**	**0.00002**

5 Conclusions

Wind forecasting data is used in this study to test the proposed optimized LSTM model's efficiency. A new optimization technique is used to optimize the parameters of the LSTM network.

To improve the exploration and exploitation capabilities of the PSO optimizer, this optimization strategy utilizes both the PSO optimizer and SFS to generate stochastic groups of agents. Experiments were done to evaluate the suggested approach's performance and compare it to findings from six other machine learning models to demonstrate its viability. To assess the success of the project, five assessment criteria are used. This approach's stability and effectiveness were further illustrated with analysis plots derived from the obtained data. It's clear from the comparisons with other models that the strategy we've provided is superior. The future of this study entails testing the suggested technique on different datasets and assessing its applicability to other prediction tasks.

Data Availability: A data availability found in https://www.kaggle.com/datasets/berkerisen/wind-turbine-scada-dataset.

Funding Statement: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Z. Lin, X. Liu and M. Collu, "Wind power prediction based on high-frequency SCADA data along with isolation forest and deep learning neural networks," *International Journal of Electrical Power & Energy Systems*, vol. 118, no. 105835, pp. 1–10, 2020.

[2] T. Ahmad and D. Zhang, "A data-driven deep sequence-to-sequence long-short memory method along with a gated recurrent neural network for wind power forecasting," *Energy*, vol. 239, no. 122109, pp. 1–20, 2022.

[3] F. Zhang, P. -C. Li, L. Gao, Y. -Q. Liu and X. -Y. Ren, "Application of autoregressive dynamic adaptive (ARDA) model in real-time wind power forecasting," *Renewable Energy*, vol. 169, pp. 129–143, 2021.

[4] S. Hanifi, X. Liu, Z. Lin and S. Lotfian, "A critical review of wind power forecasting methods—past, present and future," *Energies*, vol. 13, no. 15, pp. 1–24, 2020.

[5] M. H. Lipu, M. S. Miah, M. A. Hannan, A. Hussain, M. R. Sarker *et al.*, "Artificial intelligence based hybrid forecasting approaches for wind power generation: Progress, challenges and prospects," *IEEE Access*, vol. 9, pp. 102460–102489, 2021.

[6] Y. Wang, R. Zou, F. Liu, L. Zhang and Q. Liu, "A review of wind speed and wind power forecasting with deep neural networks," *Applied Energy*, vol. 304, no. 117766, pp. 1–24, 2021.

[7] M. Yang, C. Shi and H. Liu, "Day-ahead wind power forecasting based on the clustering of equivalent power curves," *Energy*, vol. 218, no. 119515, pp. 1–10, 2021.

[8] H. Yin, Z. Ou, S. Huang and A. Meng, "A cascaded deep learning wind power prediction approach based on a two-layer of mode decomposition," *Energy*, vol. 189, no. 116316, pp. 1–11, 2019.

[9] H. -F. Yang and Y. -P. P. Chen, "Representation learning with extreme learning machines and empirical mode decomposition for wind speed forecasting methods," *Artificial Intelligence*, vol. 277, no. 103176, pp. 1–10, 2019.

[10] I. K. Bazionis and P. S. Georgilakis, "Review of deterministic and probabilistic wind power forecasting: Models, methods, and future research," *Electricity*, vol. 2, no. 1, pp. 13–47, 2021.

[11] H. Sun and R. Grishman, "Lexicalized dependency paths based supervised learning for relation extraction," *Computer Systems Science and Engineering*, vol. 43, no. 3, pp. 861–870, 2022.

[12] Y. Deng, H. Jia, P. Li, X. Tong, X. Qiu *et al.*, "A deep learning methodology based on bidirectional gated recurrent unit for wind power prediction," in *2019 14th IEEE Conf. on Industrial Electronics and Applications (ICIEA)*, Xi'an, China, pp. 591–595, 2019.

[13] C. Yildiz, H. Acikgoz, D. Korkmaz and U. Budak, "An improved residual-based convolutional neural network for very short-term wind power forecasting," *Energy Conversion and Management*, vol. 228, no. 113731, pp. 1–15, 2021.
[14] Z. Sun and M. Zhao, "Short-term wind power forecasting based on VMD decomposition, ConvLSTM networks and error analysis," *IEEE Access*, vol. 8, pp. 134422–134434, 2020.
[15] L. -L. Li, X. Zhao, M. -L. Tseng and R. R. Tan, "Short-term wind power forecasting based on support vector machine with improved dragonfly algorithm," *Journal of Cleaner Production*, vol. 242, no. 118447, pp. 1–12, 2020.
[16] H. Yin, Z. Ou, J. Fu, Y. Cai, S. Chen *et al.,* "A novel transfer learning approach for wind power prediction based on a serio-parallel deep learning architecture," *Energy*, vol. 234, no. 121271, pp. 1–16, 2021.
[17] A. -N. Buturache and S. Stancu, "Wind energy prediction using machine learning," *Low Carbon Economy*, vol. 12, no. 1, pp. 1–21, 2021.
[18] Q. Wu, F. Guan, C. Lv and Y. Huang, "Ultra-short-term multi-step wind power forecasting based on CNN-LSTM," *IET Renewable Power Generation*, vol. 15, no. 5, pp. 1019–1029, 2021.
[19] N. Li, F. He and W. Ma, "Wind power prediction based on extreme learning machine with kernel mean P-power error loss," *Energies*, vol. 12, no. 673, pp. 1–19, 2019.
[20] G. Chen, L. Li, Z. Zhang and S. Li, "Short-term wind speed forecasting with principle-subordinate predictor based on Conv-LSTM and improved BPNN," *IEEE Access*, vol. 8, pp. 67955–67973, 2020.
[21] Y. -Y. Hong and C. L. P. P. Rioflorido, "A hybrid deep learning-based neural network for 24-h ahead wind power forecasting," *Applied Energy*, vol. 250, pp. 530–539, 2019.
[22] V. K. Rayi, S. P. Mishra, J. Naik and P. K. Dash, "Adaptive VMD based optimized deep learning mixed kernel ELM autoencoder for single and multistep wind power forecasting," *Energy*, vol. 244, no. 122585, pp. 1–29, 2022.
[23] A. Alkesaiberi, F. Harrou and Y. Sun, "Efficient wind power prediction using machine learning methods: A comparative study," *Energies*, vol. 15, no. 2327, pp. 1–24, 2022.
[24] B. Liu, S. Zhao, X. Yu, L. Zhang and Q. Wang, "A novel deep learning approach for wind power forecasting based on WD-LSTM model," *Energies*, vol. 13, no. 4964, pp. 1–17, 2020.
[25] J. Zhang, J. Yan, D. Infield, Y. Liu and F. Lien, "Short-term forecasting and uncertainty analysis of wind turbine power based on long short-term memory network and Gaussian mixture model," *Applied Energy*, vol. 241, pp. 229–244, 2019.
[26] M. Qolipour, A. Mostafaeipour, M. Saidi-Mehrabad and H. R. Arabnia, "Prediction of wind speed using a new Grey-extreme learning machine hybrid algorithm: A case study," *Energy & Environment*, vol. 30, no. 1, pp. 44–62, 2019.
[27] D. Karamichailidou, V. Kaloutsa and A. Alexandridis, "Wind turbine power curve modeling using radial basis function neural networks and tabu search," *Renewable Energy*, vol. 163, pp. 2137–2152, 2021.
[28] H. Demolli, A. S. Dokuz, A. Ecemis and M. Gokcek, "Wind power forecasting based on daily wind speed data using machine learning algorithms," *Energy Conversion and Management*, vol. 198, no. 111823, pp. 1–12, 2019.
[29] R. Yu, J. Gao, M. Yu, W. Lu, T. Xu *et al.,* "LSTM-EFG for wind power forecasting based on sequential correlation features," *Future Generation Computer Systems*, vol. 93, pp. 33–42, 2019.
[30] M. Y. Shams, A. Tolba and S. Sarhan, "A vision system for multi-view face recognition," *International Journal of Circuits, Systems, and Signal Processing*, vol. 10, no. 1, pp. 455–461, 2017.
[31] M. Y. Shams, S. Sarhan and A. Tolba, "Adaptive deep learning vector quantisation for multimodal authentication," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 8, no. 3, pp. 702–722, 2017.
[32] R. Bello, Y. Gomez, A. Nowe and M. M. Garcia, "Two-step particle swarm optimization to solve the feature selection problem," in *Proc. of the Int. Conf. of Intelligent Systems and Design Applications*, Rio de Janeiro, Brazil, pp. 691–696, 2007.

[33] E. -S. M. El-Kenawy, M. M. Eid, M. Saber and A. Ibrahim, "MbGWO-SFS: Modified binary grey wolf optimizer based on stochastic fractal search for feature selection," *IEEE Access*, vol. 8, no. 1, pp. 107635–107649, 2020.

[34] W. Sun, X. Chen, X. R. Zhang, G. Z. Dai, P. S. Chang *et al.*, "A multi-feature learning model with enhanced local attention for vehicle re-identification," *Computers, Materials & Continua*, vol. 69, no. 3, pp. 3549–3560, 2021.

[35] W. Sun, G. C. Zhang, X. R. Zhang, X. Zhang and N. N. Ge, "Fine-grained vehicle type classification using lightweight convolutional neural network with feature optimization and joint learning strategy," *Multimedia Tools and Applications*, vol. 80, no. 20, pp. 30803–30816, 2021.