



Predicting the Energy Output of Wind Turbine Based on Weather Condition



IBM NALAIYA THIRAN REPORT

Submitted by

BRINDHA M

ARUNA N

DEEPIKA K

HARINI K S

Date	01 November 2022
Team ID	PNT2022TMID05673
Project Name	EMERGING METHODS FOR EARLY DETECTION OF FOREST FIRES
Maximum Marks	8 Marks

SRI RAMAKRISHNA ENGINEERING COLLEGE

COIMBATORE – 641 022

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	i
	LIST OF FIGURES	ii
	LIST OF TABLES	iii
	LIST OF ABBREVIATIONS	iv
1	INTRODUCTION	1
	1.1 Overview	1
	1.2 Deep learning	1
	1.3 Bi-LSTM	3
2	LITERATURE REVIEW	4
	2.1 Related Works	4
	2.2 Comparative Study	7
3	SOFTWARE DESCRIPTION	9
	3.1 Software Requirements	9
4	PROJECT DESCRIPTION	10
	4.1 System Architecture	11
	4.1.1 Data Collection	11
	4.1.2 Data Pre-Processing	14
	4.1.3 Training with deep learning models	15
	4.1.4 Graphical User Interface	15
	4.2 Module Description	16
	4.2.1 Prediction Module	16
	4.2.2 Forecasting Module	16
5	SYSTEM IMPLEMENTATION	17
6	ADVANTAGES OF THE PROPOSED SYSTEM	19
7	RESULTS	20
8	CONCLUSION AND FUTURE SCOPE	22
9	REFERENCES	24
	APPENDIX	26

ABSTRACT

Extracting electricity from renewable resources has been widely investigated in the past decades to decrease the worldwide crisis in the electrical energy and environmental pollution. For a wind farm which converts the wind power to electrical energy, a big challenge is to accurately predict the wind power in spite of the fluctuations. The energy output of a wind farm is highly dependent on the weather conditions present at its site. For the wind-farm operator, this poses difficult in the system scheduling and energy dispatching, as the schedule of the wind-power availability is not known in advance. A precise forecast needs to overcome problems of variable energy production caused by fluctuating weather conditions. If the output can be predicted more accurately, energy suppliers can coordinate the collaborative production of different energy sources more efficiently to avoid costly overproduction. The objective of our project to develop an end-to-end web application to predict & forecast the energy output of the wind turbine based on weather conditions. In this project, a prediction system is developed using a special kind of RNN (Recurrent Neural Network), Bidirectional Long Short Term Memory (Bi-LSTM) deep learning model which has a prominent performance in capturing the long-term dependencies along the time steps, and thus very applicable for wind power prediction.

LIST OF FIGURES

FIGURE NO	NAME OF THE FIGURE	PAGE NO
1.1	Recurrent Neural Network	2
1.2	Bi-LSTM	3
3.1	Technology Stack	9
4.1	System Architecture	11
4.2	Power versus Wind Speed	12
4.3	Power versus Air Temperature	12
4.4	Power versus Air Pressure	13
4.5	Power versus Wind Direction	14
4.6	Accuracy observed with different models	15
4.7	Prediction Module Architecture	16
4.8	Forecasting Module Architecture	17
5.1	Block Diagram	18
6.1	Preferences Tab	19
7.1	User Interface	20
7.2	Forecasting Results	21

LIST OF TABLES

TABLE NO	NAME OF THE TABLE	PAGE NUMBER
2.1	Comparative Study	8

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
ANN	Artificial Neural Network
Bi-LSTM	Bidirectional Long Short Term Memory
CNN	Convolutional Neural Network
DBN	Deep Belief Network
GA-LSTM	Genetic Algorithm-Optimized Long Short-Term Memory
GRU	Gated Recurrent Units
LSTM	Long Short Term Memory
NARX	Nonlinear Autoregressive Exogenous Model
NWP	Numerical Weather Prediction
RBM	Restricted Boltzmann Machine
RNN	Recurrent Neural Network
SVM	Support Vector Machine

CHAPTER 1

INTRODUCTION

1.1 Overview

In today's power grid, the penetration of **wind energy** has been booming. The growing integration of wind turbines into the power grid can only be balanced with **precise forecasts of upcoming energy productions**. Knowing the wind power beforehand helps us in many ways by minimizing the losses. The technique incorporated in our project is deep learning. Bi-LSTM (Bidirectional Long Short Term Memory), a special kind of Recurrent Neural Network (RNN) is a deep learning model which makes use of multiple hidden layers to train the model and provide accurate results. The trained models are rendered and deployed in a web server with a simple user interface.

1.2 Deep Learning:

In recent years, many deep learning methods, such as the CNN (Convolutional Neural Network), and the RNN (Recurrent Neural Network), are more and more extensively used in wind power prediction. In weather conditions based forecasting features like pressure, temperature, humidity, precipitation are used and they are not constant all the time and hence they form a time series. Time series problems are mostly solved using RNN. Fig.1.1 represents the structure of RNN. These models have memory, i.e., they can remember the information throughout the time.

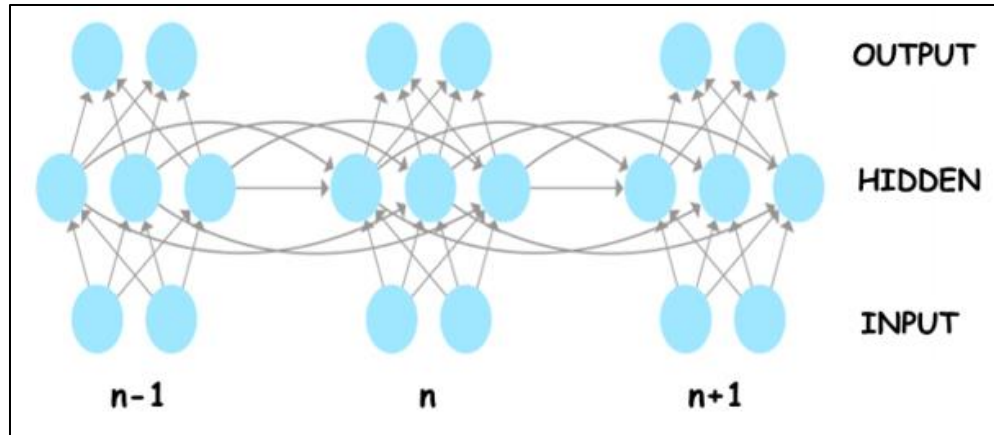


Figure 1.1: Recurrent Neural Network

Thus, they are mostly used in sequence prediction problems. RNN with the help of internal memory and what it has learnt from the previous inputs, tries to predict the next output. A simple RNN model will not be able to store the long-term dependencies. Due to gradient disappearance and gradient explosion its long-term information gets lost and also it will affect the prediction performance. Bidirectional Long short-term memory (Bi-LSTM) was proposed in order to overcome the gradient disappearance and the gradient explosion. Another shortcoming of conventional RNNs is that they are only able to make use of previous context. Bidirectional RNNs (BRNNs) do this by processing the data in both directions with two separate hidden layers, which are then fed forwards to the same output layer.

Combining BRNNs with LSTM gives Bidirectional LSTM, which can access long-range context in both input directions. So, our prediction system is developed using BiLSTM deep learning model. In Bidirectional LSTM the output layer gets feedback from past (forward) as well as future (backward) states simultaneously, thus output is more accurate and precise.

1.3 Bi-LSTM:

The idea of Bidirectional LSTMs (Bi-LSTM) is to aggregate input information in the past and future of a specific time step in LSTM models. In Bi-LSTM, at any point in time, you are able to preserve information from both past and future.

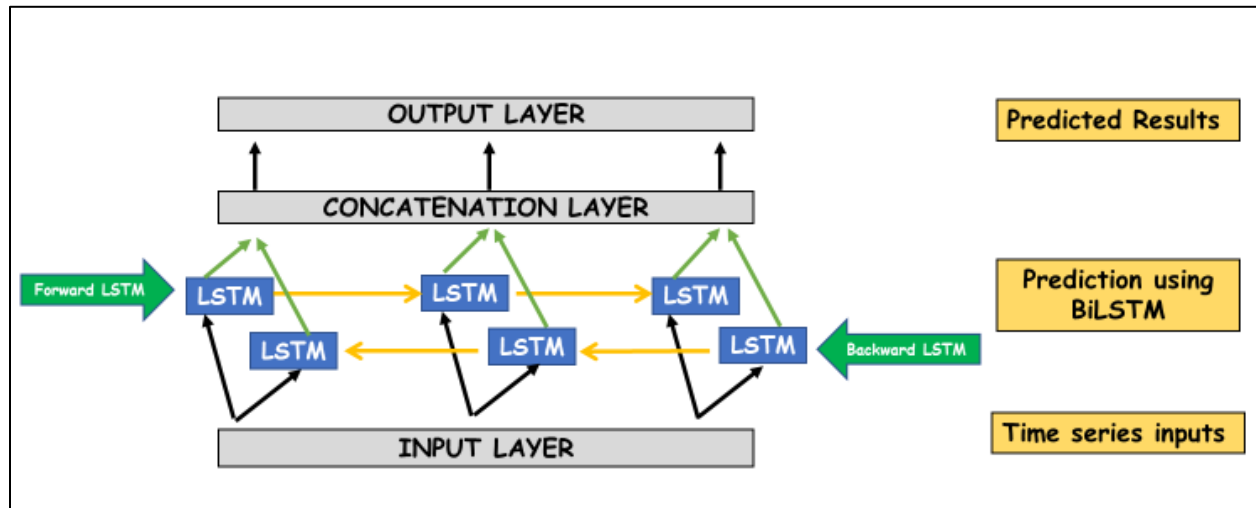


Figure 1.2: Architecture of Bi-LSTM

Fig.1.2 explains the architecture of Bi-LSTM. It involves duplicating the first recurrent layer in the network so that there are now **two layers side-by-side**, then providing the input sequence as input to the first layer and providing a reversed copy of the input sequence to the second. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell is responsible for remembering values over arbitrary time intervals; each of the three gates can be thought of as regulators of the flow of values that goes through the connections of the LSTM. There are connections between these gates and the cell. The expression long short-term refers to the fact that BiLSTM is a model for the short-term memory which can last for a long period of time. An BiLSTM is well suited to classify, process and predict time series given time lags of unknown size and duration between important events.

CHAPTER 2

LITERATURE REVIEW

2.1 Related Works

With rapid rise in technology, the demand of wind energy is humongous and is increasing tremendously. Wind energy has great potential for electric power conversion and will be able to ensure sizeable contribution to the electrical energy demand of the world. Power generation from wind is highly susceptible to climatic variables viz. geographical allocation, wind speed, pressure, temperature, wind direction etc. The pattern of wind power is highly erratic in nature. Thus direct statistical models cannot give accurate predictions. Most of the works in the literature employ hybrid models which combine physical and statistical models.

Yubotao et al [1]., built a model using deep belief networks DBN is used to extract the hidden rules of wind power based on the historical data from wind farm has higher accuracy. The multi-layer BP network is used with DBN as the primary hidden layer with node number as 100. The other hidden layers are selected as same as that of DBN with node number as 200. This method is effective in improving the prediction accuracy of wind power, which prediction error (MSE AND MAE) with prediction of the size and the growth rate of the step is far less than the other methods.

Justin Philip Heinerman [2], proposed ensemble models to predict the wind power to reduce the prediction error. Ensemble models combines the prediction numerous and preferably diverse models to predict the wind power. SVR ensembles and RF offer a good prediction for a data driven prediction of the power output to except. The use of appropriate features helps to improve the prediction. According to the author, this method offers efficient and comfortable balancing of a preferably low prediction error.

Tayeb Brahim [3], proposed Artificial Neural Networks (ANN) as a powerful to predict the wind speed. Based on the tests, ANN proved itself to be a flexible method in terms of accuracy and computer time usage. Correlation between actual and predicted data was low in the case of random tree, where the RMS was relatively high. ANN prediction could be improved by conducting additional tests on hidden layers and ANN parameters producing the best RMS and correlation.

Qu Xiaoyun et al. [4], proposed a deep learning model for short-term wind power prediction based on LSTM. In this proposed method, in order to reduce the dimension of the input variables and to reduce the complexity of the network, suitable input samples are selected using principal component analysis. Simulation results show that, compared with BP neural network and SVM, the LSTM prediction model has higher prediction accuracy and greater potential for the field of wind power prediction.

Sowmya SevoorVaitheeswaran et al. [5], proposed LSTM model to predict and validate wind power using time series measurements based on the online wind power measurements. For wind power prediction short term pattern prediction and mid-term are studied. A RMSE of 0.0957993 and 0.0929905 is obtained for the short and medium term time horizon models respectively. The GA model improves the reliability by finding the appropriate number of time lags in the model.



































Yiwei Fu et al. [6], proposed a prediction model based on RNN with LSTM or GRU to improve the accuracy. Firstly an overall forecasting framework for wind power with diverse forms of optional hybrid models is proposed. With the help of wind speed correction process an innovative LSTM/GRU based forecasting model is developed using NWP data. Based on the results it is seen that LSTM and GRU based models

provides better forecasting results when compared with ARIMA and SVM models.

Zinchao Shi et al. [7], proposed a new idea of Prediction Intervals (PIs) is employed to capture the uncertainty of wind power generation in power systems. To construct PIs with lower upper bound estimation (LUBE) a RNN model is proposed. To tune the parameters of RNN, the dragonfly algorithm (DA) with a linearly random weight update method is used as optimization tool. The results with wind power dataset show that the RNN model has higher prediction accuracy when compared to other benchmark forecasting models.

Jaseela V Rasheed [8], proposed a hybrid model to forecast wind power using LSTM. An ANN model is used to forecast the wind speed. A special type of RNN – LSTM is used for the model. The time series model used to predict the wind power which requires only small amount data. The errors in the model are quite less and this model can be used where there is difficulty in data acquisition. But the prediction errors are very high, especially in the turning points of the time series.

2.2 Comparative Study

Algorithm	Input parameters considered	Findings of the study
[5] Wind Power Pattern Prediction in time series measurement data for wind energy prediction modelling using LSTM-GA networks.		
Deep Learning Algorithm [1] Wind Power Prediction and Pattern Feature Based on Deep Learning Method	 Wind energy data set.	The Genetic Algorithm model improves the reliability by finding the appropriate number of time lags in the model.
Deep learning algorithm.  DBN  RBM training process  RBM hidden layer	 Variation patterns of wind.	This method is effective in improving the prediction accuracy of wind power and prediction error far less than other methods.
[6] Multi-step Ahead Wind Power Forecasting Based on Recurrent Neural Networks		
[2] Wind Power Prediction with Machine Learning Ensembles		
 LSTM	 Pitch angle	This methods have significantly better forecasting performances compared with the ARIMA and SVM methods.
 GRU Learning Algorithms  k-Nearest Neighbors  Decision Tree	 Radiation capacity  Fast parameters of wind turbine	ANN training yield superior results with preferably low prediction error.
[7] Direct Interval Forecast of Uncertain Wind Power Based on Recurrent Neural Networks		
 Wind Power forecasting:- NWP  Wind Power Predictions:- Historical time series	 Hourly wind power data.	The proposed RNN prediction model can construct better prediction compared with the benchmark models.
[3] Using Artificial Intelligence to Predict Wind Speed for Energy Application in Saudi Arabia		
Deep Learning Algorithm (RNN)  Elman Network  NARX Network	 Air temperature  Wind direction  Speed	ANN predictions could be improved by conducting additional tests on hidden layers and ANN parameters producing the best RMS and correlation.
[8] A Survey on Hybrid model to Forecast Wind Power using LSTM		
 Dragonfly Algorithm (for Randomforest)  Randomtree  Reptree  ANN  SVM	 Global-Horizontal  Humidity  Wind speed.	There are less errors and the model is suitable for situations with Short-Term Memory difficulty.
[4] Short-Term Prediction of Wind Power Based on Deep Long Short-Term Memory		
Deep Learning Algorithm.  LSTM	 Wind speed	The LSTM prediction model has higher prediction accuracy and greater potential compared with BP network and SVM.
 LSTM	 Temperature  Humidity  Pressure	

CHAPTER 3

SOFTWARE DESCRIPTION

3.1 Software Requirements

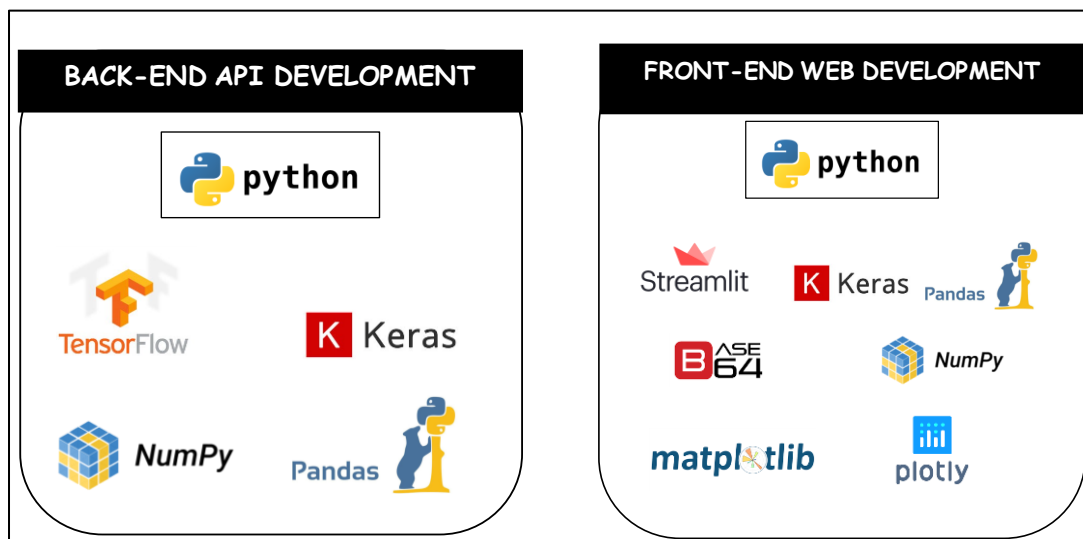


Figure 3.1: Technology stack

The software used in our project are:

- **Python 3.9:** Python is an interpreted, high level, general programming language. It provides a vast library for data mining and predictions.
- **Spyder/Pycharm:** It is an open source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder

integrates with a number of prominent packages as well as another open source software.

- **Streamlit:** Streamlit is an open-source Python library that makes it easy to create and share beautiful, custom web apps for machine learning and data science.
- **Base64:** To encode and decode a binary image (for background image).
- **Tensorflow:** It is a foundation library that can be used to create Deep learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.
- **Keras:** Keras is a powerful and easy-to-use free open source Python library for developing and evaluating deep learning models.
- **Numpy:** Numpy supports large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- **Pandas:** Pandas was used for the data pre-processing and statistical analysis of data.
- **Matplotlib, Plotly:** Matplotlib & Plotly was used for the graphical representation of our forecasting.

CHAPTER 4

PROJECT DESCRIPTION

4.1 System Architecture

The models operating on the production server would work with the real-life data and provide predictions to the users. The below mentioned framework represents the most basic way data scientists handle deep learning. Fig.4.1 represents our System Architecture.

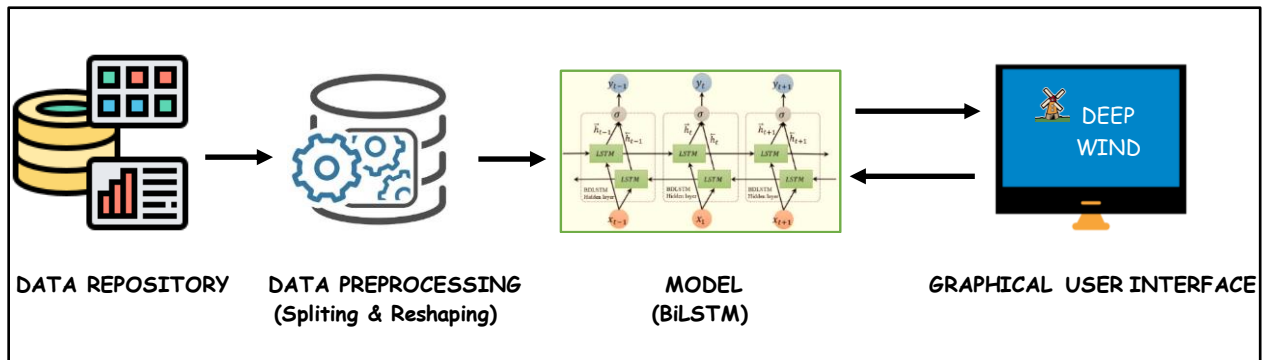


Figure 4.1: System Architecture

4.1.1. Data Collection:

Collecting the required data is the beginning of the whole process. Data repository has the repository of all the data related to weather conditions and power generated. The data which is used to train the model is obtained from National Renewable Energy Laboratory (NREL) to do this analysis. The dataset contains the details about

timestamp, air temperature (°C), pressure (atm), wind direction (deg), wind speed (m/s) and Power generated by the system (kW). We have hourly data for about 6 years (i.e) almost 52000 entries.

Input Parameters:

- **Wind Speed (m/s):**

Higher wind speeds generate more power because stronger winds allow the blades to rotate faster. Faster rotation translates to more mechanical power and more electrical power from the generator. Fig.4.2 shows the variation of power output with respect to wind speed.

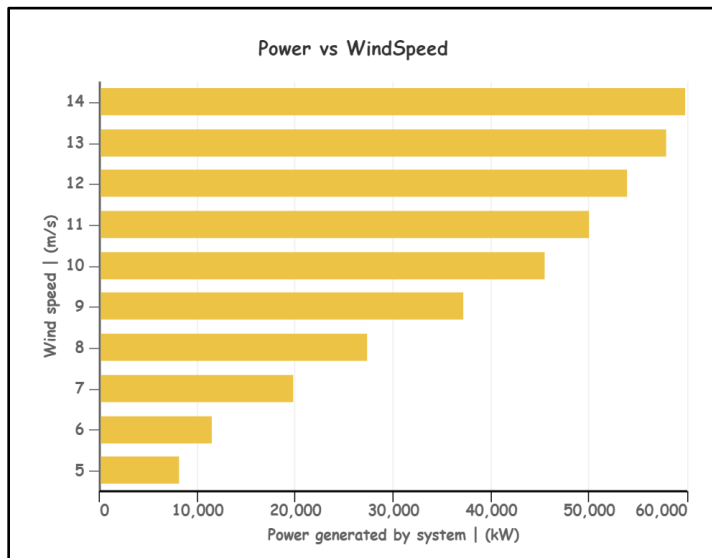


Figure 4.2: Graph of Power versus Wind Speed

- **Air Temperature (°C):**

Wind speeds increase with a greater temperature difference. If the temperature is too high, the air density will be low, which will lessen the energy output. If

the temperature is too low, the blades and other parts might be frozen, and the wind turbine will stop working. Fig.4.3 shows the variation of power output with respect to air temperature.

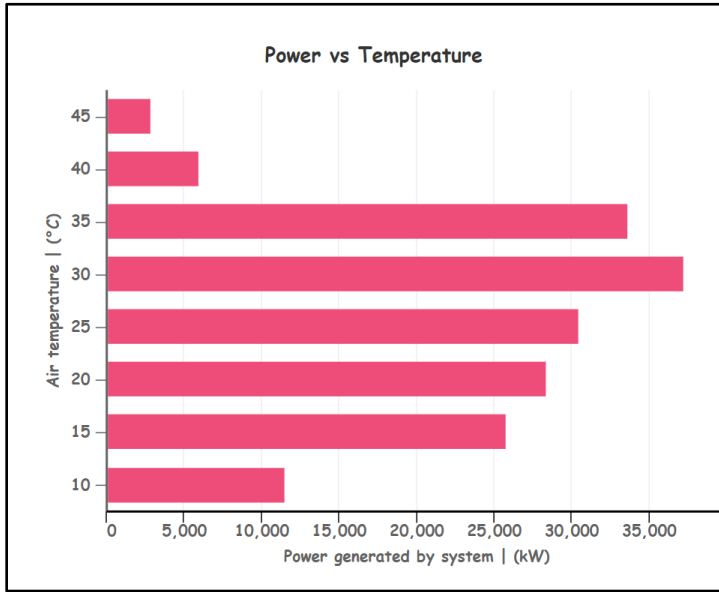


Figure 4.3: Graph of Power versus Air Temperature

- **Air Pressure (atm):**

When air slows down, its pressure increases. This means that higher wind speeds will show lower air pressure readings. Fig.4.4 shows the variation of power output with respect to air pressure. We can conclude that wind power increases with decrease in air pressure.

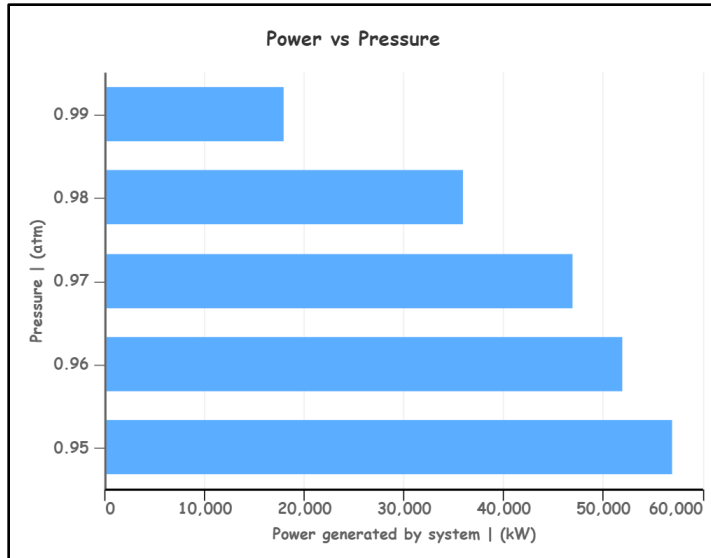


Figure 4.4: Graph of Power versus Air Pressure

- **Wind Direction (deg):**

Wind flow direction affects the turbines, reducing the wind speed and increasing turbulence for the wind turbines. A weather vane is a instrument which shows the direction the wind is blowing. Getting more nearer to 360° (north), wind speed increases, so severe winds blow from north generating more power. Fig.4.5 shows the variation of power output with respect to wind direction.

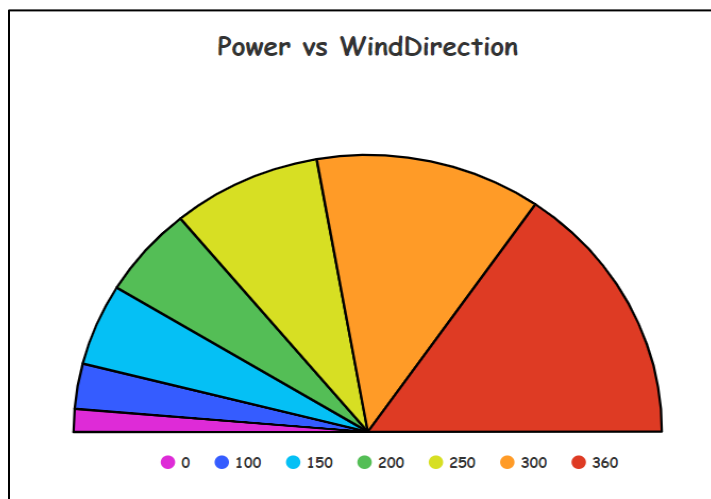


Figure 4.5: Graph of Power versus Wind Direction

From all the above graphs we can able to learn the pattern in the data very well. Wind speed available at the wind farm is a crucial parameter. Other parameters that influence the energy output are for example temperature, pressure, wind direction. Our goal is to make use of the correlation of these parameters with respect to the energy output.

$$\text{Power (kW)} = \frac{(\text{wind speed})^3 * \text{wind direction} * 10}{\text{pressure} * \text{temperature (in } ^\circ\text{F)}}$$

4.1.2. Data Pre-processing:

The model is completely dependent on the data, so it needs to be consistent and precise. Any missing value is handled, redundant values and columns are dropped out in this step. EDA (Exploratory Data Analysis) helps in visualizing the data. This process helps to decide which attributes are relevant for the model. With the help of pre-processing we get the data that is suitable to train the data.

4.1.3. Training with Deep Learning Models:

Train the deep learning model with different sets of processed data. Finally select the model that gives the maximum accuracy of all. Fig.4.6 represents the accuracy observed with different models.

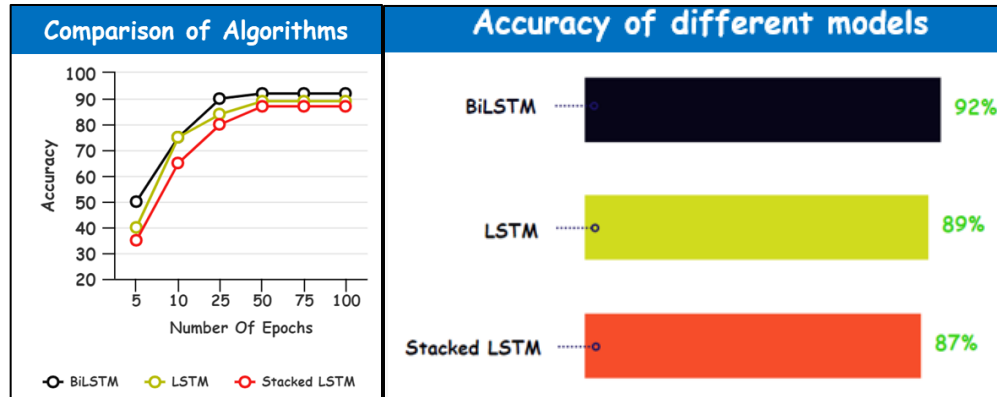





Figure 4.6: Accuracy observed with different models

4.1.4. Graphical User Interface (GUI):

A simple, easy and understandable user interface is designed so that any layman can use it to know the future wind power status. For Predicting the power output for user defined values, user just needs to input the weather parameters for which the power is to be predicted. For Forecasting the power output of wind turbine from several hours up to 24 hours ahead, user can upload their own real time dataset (csv or xlsx format) for forecasting. A simple visualization tool is also presented in our user interface, where the data can be visualized using the graphs.

4.2Module Description

4.2.1. Prediction Module

-  Predicting the power output for user defined values.
-  User just needs to input the weather parameters for which the power is to be predicted. Hence, the GUI is user friendly, easy, simple in nature.
-  Fig.4.7 represents the architecture of prediction module. First, we train our model with the historic data. Later, we can feed a new weather data to the

model and predict the power output.

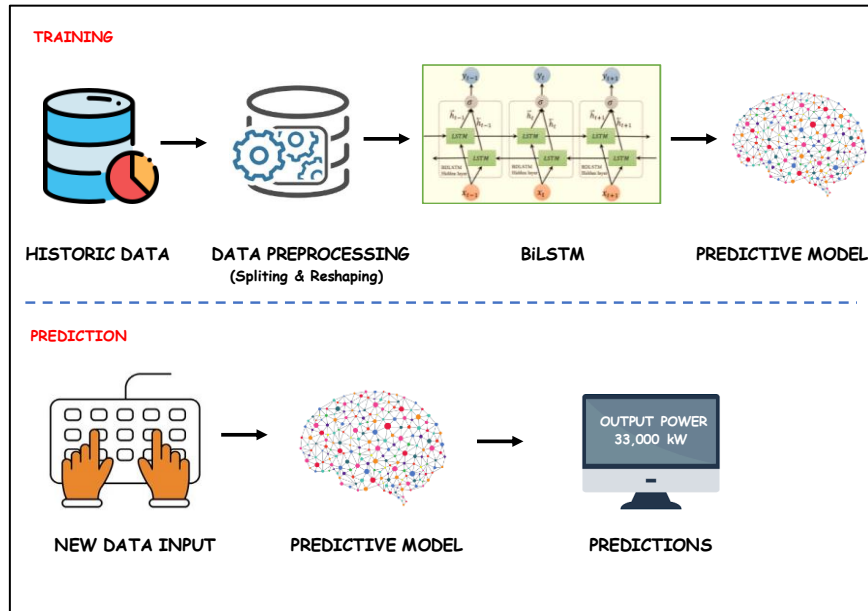






Figure 4.7: Prediction Module Architecture

4.2.2. Forecasting Module

-  Forecasting the power output of wind turbine from several hours up to 24 hours ahead.
-  Our Deep Wind is individually different from other wind power forecasting websites where the user can upload their own real time dataset (csv or xlsx format) for forecasting.
-  Fig.4.8 represents the architecture of forecasting module. The dataset uploaded by the user is divided into train and test sets and given to our model for forecasting the power output.
-  Accurate Results with minimum load time.

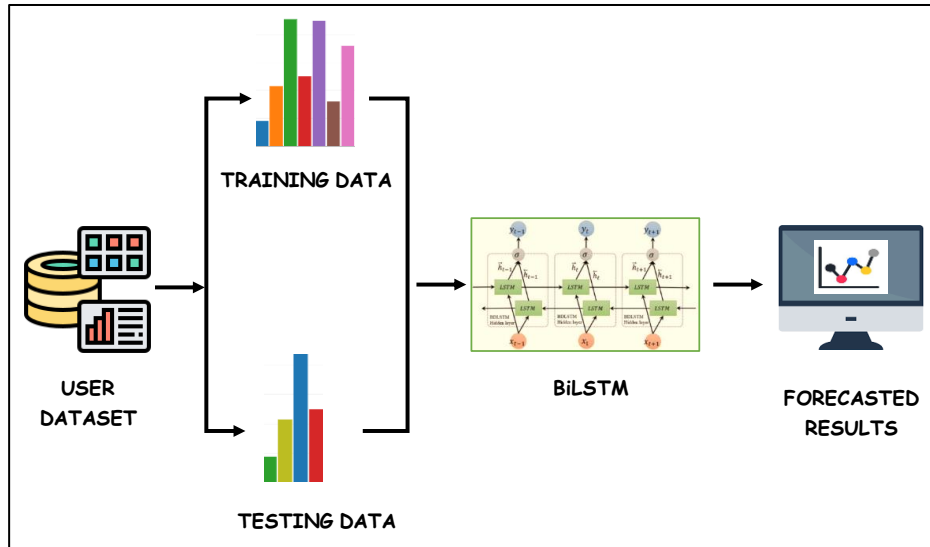


Figure 4.8: Forecasting Module Architecture

CHAPTER 5

SYSTEM IMPLEMENTATION

Our project is deep learning based. This is implemented by making use of python language. The models are developed using many libraries available in python language. The deep learning based model is implemented with the help of numpy, pandas, tensorflow and keras libraries. The trained models are deployed on a web server which is implemented using streamlit library.

Fig.5.1 represents the block diagram. The flow of the application is as follows:

- ✚ The starting part of the application is the User Interface (UI). As the web application is visited by the user, a simple get request is sent from the streamlit

frontend to the backend. This request, signals the backend server to run the model file.

- ✚ Our Bi-LSTM requires some input dataset for the model to predict the future values. This data is then converted into the required format and sent as the input for the model to get the required predictions for 24 hours ahead.
- ✚ Once this data is provided to the model it starts the calculation and provides the necessary output.
- ✚ When the models provide the output, the server sends these outputs as the response to the request and these values are displayed in the User Interface.

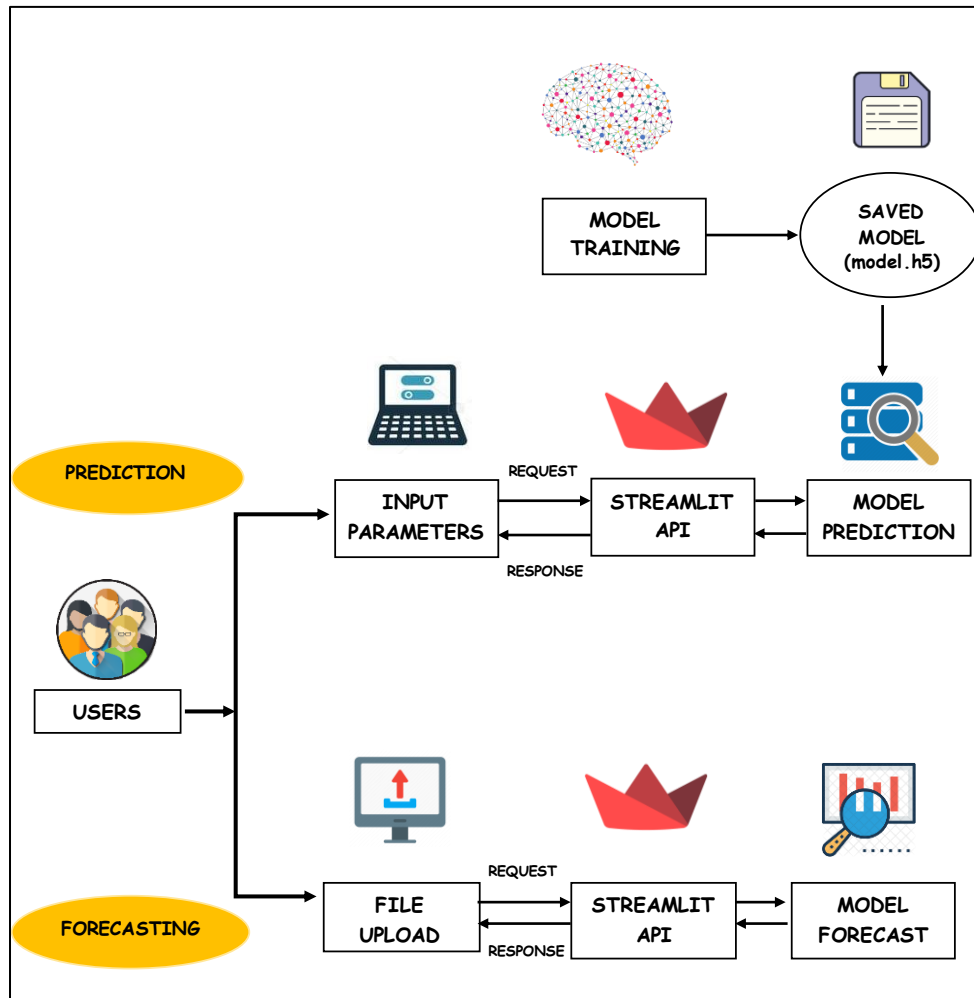


Figure 5.1: Block Diagram (flow of the application)

CHAPTER 6

ADVANTAGES OF THE PROPOSED SYSTEM

- ✚ The RNN can process any amount of data which is very useful in power forecasting because it is based on previous data.
- ✚ The neural network models presumes inputs as independent of each other and for every input the neural network layers and the output are also calculated independently, but in case of RNN as we use Bi-LSTM (Bidirectional Long Short-Term Memory) the computed output of the previous layer becomes the input of the next layer thereby forming a sequence of tasks which is useful in prediction and forecasting.
- ✚ The other main advantage of RNN is the model size does not depend on size of input.
- ✚ User can also set their preferable cut in (minimum) and cut out(maximum) values for windspeed and temperature with the help of preferences tab.

The screenshot shows a 'Preferences' window with two main sections: 'Temperature' and 'Wind Speed'. Each section has a minimum and maximum value input field with minus and plus buttons for adjustment.

Category	Parameter	Value
Temperature (°C)	Minimum Temperature	-15
	Maximum Temperature	50
Wind Speed (m/s)	Minimum Wind Speed	1
	Maximum Wind Speed	27

Figure 6.1: Preferences Tab

- Our Deep Wind is individually different from other wind power forecasting websites where the user can upload their own real time dataset (csv or xlsx format with a minimum of 30 entries) for forecasting.
- Accurate Results with minimum load time.

CHAPTER 7

RESULTS

The goal of an effective user interface is to make the experience of a user very simple and interactive, requiring minimal effort from the user to achieve the maximum desired results. Fig.7.1 represents our user interface consisting of three tabs home page, user defined prediction and forecasting tab.

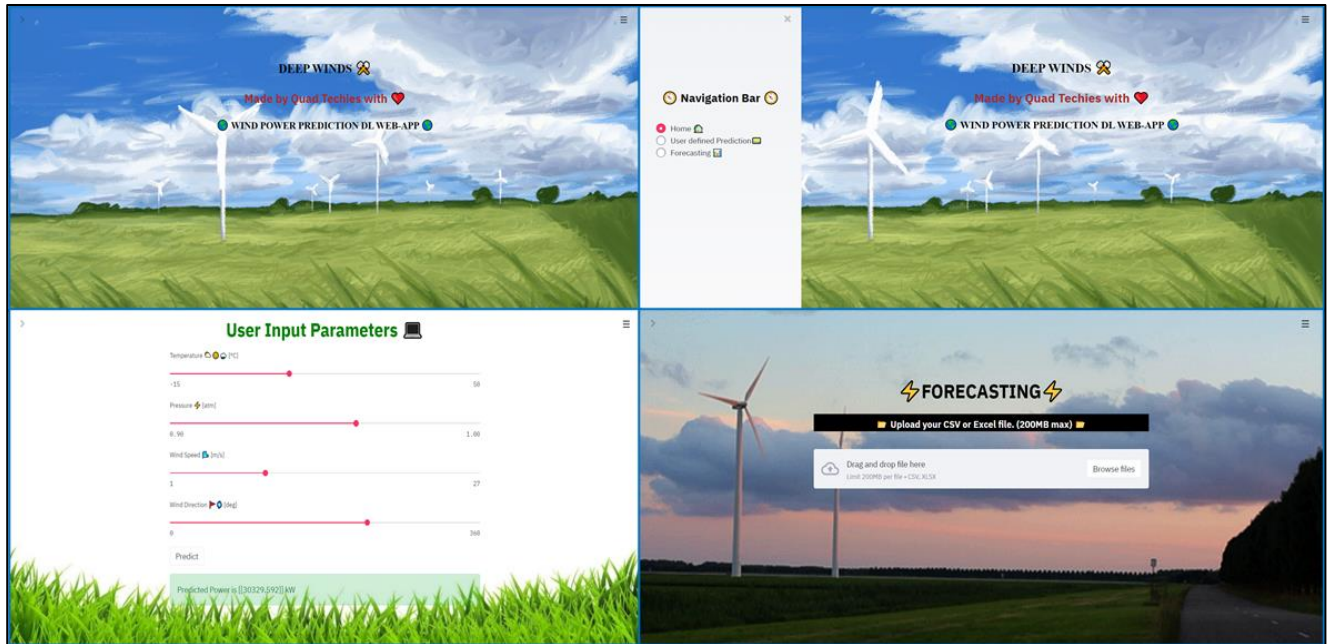


Figure 7.1: User Interface

Our Deep wind provides an interactive interface with a simple visualization tool which brings the accurate results with minimal load time. Fig.7.2 shows the result obtained from the forecasting.

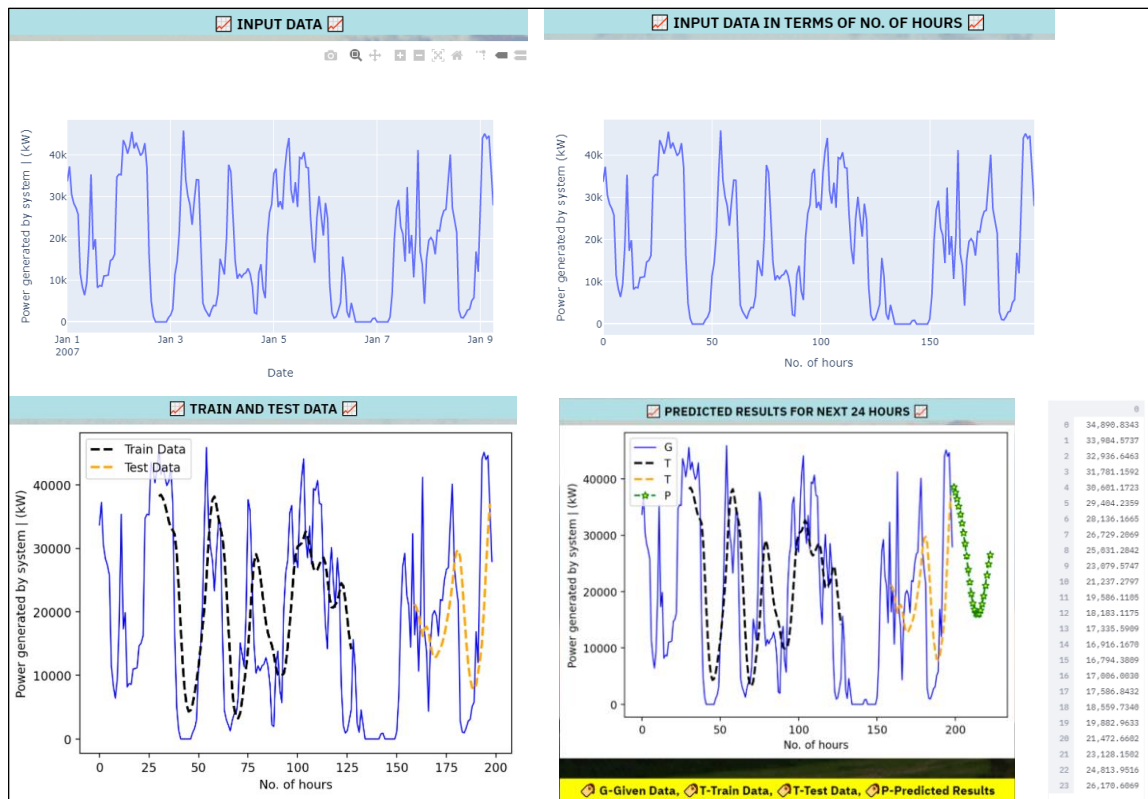


Figure 7.2: Forecasting Results

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

Thus accurate wind power forecasting plays a key role in dealing with the challenges of power system operation under uncertainties in an economical and technical way. This unique approach would surely open up new avenues and make wind farm data more reliable and precise. In our application only weather parameters are considered. Other features can also be added by training the model and integrating it with the current application. Making our website to send alerting message if the forecasted power drops to minimum value. Hopefully, the power of Deep Learning would boost the mass adoption of wind power and turn it into a popular alternative to traditional sources of electricity over the years.

CHAPTER 9

REFERENCES

- [1]Yubo Tan, Hongkunchen Chuang Qiu, “Wind Power Prediction and Pattern Feature Based on Deep Learning Method”, <http://pipad.org/tmp/WindPower.pdf> , 2014.
- [2]Justin Philipp Heinermann, “Wind Power Prediction with Machine Learning Ensembles”,<https://d-nb.info/1122481861/34>,2016.
- [3]T Brahim, “Using Artificial Intelligence to Predict Wind Speed for Energy Application in Saudi Arabia”, <https://www.mdpi.com/1996-1073/12/24/4669/pdf>,2019.
- [4]Qu Xiaoyun, Kang Xiaoning, Zhang Chao, Jiang Shuai, Ma Xiuda, “Short-Term Prediction of Wind Power Based on Deep Long Short-Term Memory”, <https://sci-hub.do/10.1109/APPEEC.2016.7779672>, 2016.
- [5] Sowmya SevoorVaitheeswaran , Varun Raj Ventrapragada , “Wind Power Pattern Prediction in time series measurement data for wind energy prediction modelling using LSTM-GA networks” , <https://sci-hub.do/https://doi.org/10.1109/ICCCNT45670.2019.8944827>, 2019
- [6] YiweiF, Wei HU, “Multi-step Ahead Wind Power Forecasting Based on Recurrent Neural Networks”, <https://scihub.do/https://doi.org/10.1109/APPEEC.2018.8566471>, 2018.
- [7]Zhichao Shi, Hao Liang, Venkata Dinavah, “Direct Interval Forecast of Uncertain Wind Power Based on Recurrent Neural Networks”, <https://sci-hub.do/https://doi.org/10.1109/TSTE.2017.2774195>, 2018.
- [8] Jaseela V Rasheed, “A Survey on Hybrid model to Forecast Wind Power using LSTM”,http://www.ijirset.com/upload/2017/july/138_A_Survey.pdf,2017.
- [9]Zhe Ren, Chengshuai Huang, Meng Li, “Research on Wind Power Prediction”, ,2019. <https://sci-hub.ee/10.1109/EI247390.2019.9061851>
- [10]Anwen Zhu, Xiaohui Li, Zhiyong Mo, HuarenWu, “Wind Power Prediction Based on a Convolutional Neural Network”,<https://sci-hub.ee/10.1109/ICCD.2017.8120465> , 2017.

- [11] Amila T. Peiris, Jeevani Jayasinghe, UpakaRathnayake, “Forecasting Wind Power Generation Using Artificial Neural Network: “Pawan Danawi”—A Case Study from Sri Lanka”, <https://doi.org/10.1155/2021/5577547> ,2021.
- [12] SumantaPasari, Aditya Shah, Utkarsh Sirpurkar, “Wind Energy Prediction Using Artificial Neural Networks”, https://link.springer.com/chapter/10.1007/978-3-030-44248-4_10 ,2020.
- [13] Z. Liu, W. Gao,Y-H. Wan, E. Muljadi, “Wind Power Plant Prediction by Using Neural Networks” , <https://www.nrel.gov/docs/fy12osti/55871.pdf> , 2012.
- [14] Zhongda Tian, “A state-of-the-art review on wind power deterministic prediction”, <https://sci-hub.ee/10.1177/0309524X20941203>, 2020.
- [15]Rowel Atienza, “ LSTM by Example using Tensorflow ”, <https://towardsdatascience.com/lstm-by-example-using-tensorflow-feb0c1968537> , 2017.

APPENDIX

➤ app.py

```
import numpy as np
import streamlit as st
import pandas as pd
import datetime
import plotly.graph_objects as go
import base64
import time
import tensorflow

st.set_page_config(
    page_title=" DEEP WIND ",
    page_icon=" 🚩 "
)
old_models =tensorflow.keras.models.load_model('model.h5')

# set background, use base64 to read local file
def get_base64_of_bin_file(bin_file):
    with open(bin_file, 'rb') as f:
        data = f.read()
    return base64.b64encode(data).decode()

def set_png_as_page_bg(png_file):
    bin_str = get_base64_of_bin_file(png_file)
    page_bg_img = ""
    <style>
        body {
            background-image: url("data:image/png;base64,%s");
            background-size: cover;
        }
    </style>
    "" % bin_str

st.markdown(page_bg_img, unsafe_allow_html=True)
return
set_png_as_page_bg('gr.gif')

def home():
    return "welcome"

def predict(temperature,pressure,wind_speed,wind_direction):
    values=np.array([[temperature,pressure,wind_speed,wind_direction]])
```



```

prediction=old_models.predict(values.reshape(-1,1,4), batch_size=1)
print(prediction)
return prediction

```

```
def main():
```

```

st.sidebar.markdown("<h1 style='text-align: center; color: black;'>🌟 Navigation Bar 🌟 </h1>",
unsafe_allow_html=True)

```

```

    nav = st.sidebar.radio("",["Home 🏠","User defined Prediction 📊","Forecasting 📈"])

```

```

    if nav == "Home 🏠":

```

```

st.markdown("<h1 style='color:black; text_align:center;font-family:times new roman;font-
size:20pt; font-weight: bold;'>DEEP WINDS ⚡ </h1>", unsafe_allow_html=True)

```

```

st.markdown("<h1 style=' color:brown; text_align:center;font-weight: bold;font-size:19pt;'>Made
by Quad Techies with ❤️ </h1>", unsafe_allow_html=True)

```

```

st.markdown("<h1 style='color:black; text_align:center;font-family:times new roman;font-weight:
bold;font-size:16pt;'>🌍 WIND POWER PREDICTION DL WEB-APP 🌍 </h1>",
unsafe_allow_html=True)

```

```

    if nav == "User defined Prediction 📊":

```

```

set_png_as_page_bg('gra (1).jpg')

```

```

st.markdown("<h1 style='text-align: center; color: green;'>User Input Parameters 🖥️ </h1>",
unsafe_allow_html=True)

```

```

with st.beta_expander("Preferences"):

```

```

st.markdown("<h1 style='text-align: left; font-weight:bold;color:black;background-
color:white;font-size:11pt;'> Temperature 🌤️ 🌞 ☁️ (°C) </h1>",unsafe_allow_html=True)

```

```

    col1,col2 = st.beta_columns(2)

```

```

    with col1:

```

```

min_temp=st.number_input('🔻 Minimum Temperature (°C)',min_value=-
89,max_value=55,value=-15,step=1)

```

```

    with col2:

```

```

max_temp=st.number_input('🔻 Maximum Temperature (°C)',min_value=-
88,max_value=56,value=50,step=1)

```

```

st.markdown("<h1 style='text-align: left; font-weight:bold;color:black;background-
color:white;font-size:11pt;'> Wind Speed 🌬️ (m/s) </h1>",unsafe_allow_html=True)

```

```

    col1,col2 = st.beta_columns(2)

```

```

    with col1:

```

```

min_speed=st.number_input('🚀 Minimum Wind Speed
(m/s)',min_value=0,max_value=99,value=1,step=1)

```

```

    with col2:

```

```

max_speed=st.number_input('🚀 Maximum Wind Speed
(m/s)',min_value=2,max_value=100,value=27,step=1)

```

```

st.write("")

```

```

    temperature = st.slider("Temperature 🌤️ 🌞 ☁️ [°C]", min_value=min_temp, step=1,
max_value=max_temp,value=max_temp)

```

```

    pressure = st.slider('Pressure ⚡ [atm]', 0.9, 1.0, 1.0)

```

```

wind_speed = st.slider('Wind Speed 🌬️ [m/s]', min_value=min_speed, step=1,
max_value=max_speed,value=max_speed)
wind_direction = st.slider('Wind Direction 🚩 🌀 [deg]', 0, 1, 360)
result = ""
if st.button("Predict"):
    result = predict(temperature,pressure,wind_speed,wind_direction)
st.balloons()
st.success('Predicted Power is {} kW'.format(result))

if nav == "Forecasting 📊 ":
    set_png_as_page_bg('04.gif')
    st.markdown("<h1 style='text-align: center; color:black ;> ⚡ FORECASTING ⚡ </h1>",
    unsafe_allow_html=True)
    # Setup file upload
    st.markdown("<h1 style='text-align:center; color:white;background-color:black;font-size:14pt> 📁
    Upload your CSV or Excel file. (200MB max) 📁 </h1>", unsafe_allow_html=True)
    uploaded_file = st.file_uploader(label="",type=['csv', 'xlsx'])
    global df
    if uploaded_file is not None:
        print(uploaded_file)
    st.markdown("<h1 style='text-align:center; color:black;background-color:lightgreen;font-
    size:14pt> 📁 File upload successful 📁 </h1>", unsafe_allow_html=True)
    try:
        df = pd.read_csv(uploaded_file)
    st.write(df)

    except Exception as e:
        df = pd.read_excel(uploaded_file)
    st.write(df)

    st.markdown("<h1 style='text-align: center; color:black ;background-color:powderblue;font-
    size:14pt> 📈 INPUT DATA 📈 </h1>", unsafe_allow_html=True)

    trace = go.Scatter(x = df['DateTime'], y = df['Power generated by system | (kW)'],mode =
    'lines',name = 'Data')
    layout = go.Layout(title = "",xaxis = {'title' : "Date"},yaxis = {'title' : "Power generated by
    system | (kW)"})
    fig = go.Figure(data=[trace], layout=layout)
    st.write(fig)

    df1=df.reset_index()['Power generated by system | (kW)']
    import matplotlib.pyplot as plt
    st.write("\n")

```

```

st.markdown("<h1 style='text-align: center; color:black ;background-color:powderblue;font-size:14pt'> INPUT DATA IN TERMS OF NO. OF HOURS </h1>",
unsafe_allow_html=True)
    trace = go.Scatter(x = df1.index,y = df['Power generated by system | (kW)'],mode = 'lines',
name = 'Data' )
    layout = go.Layout(title = "",xaxis = {'title' : "No. of hours"},yaxis = {'title' : "Power
generated by system (kW)"})

    fig = go.Figure(data=[trace], layout=layout)
    #fig.show()
st.write(fig)
    from sklearn.preprocessing import MinMaxScaler
    scaler=MinMaxScaler(feature_range=(0,1))
    df1=scaler.fit_transform(np.array(df1).reshape(-1,1))
    ##splitting dataset into train and test split
training_size=int(len(df1)*0.65)
test_size=len(df1)-training_size
    train_data,test_data=df1[0:training_size:],df1[training_size:len(df1),:1]

    import numpy
    # convert an array of values into a dataset matrix
    # convert an array of values into a dataset matrix
    def create_dataset(dataset, time_step=1):
        dataX, dataY = [], []
        for i in range(len(dataset)-time_step-1):
            a = dataset[i:(i+time_step), 0] ###i=0,0,1,2,3-----99 100
            dataX.append(a)
            dataY.append(dataset[i + time_step, 0])
        return numpy.array(dataX), numpy.array(dataY)
    # reshape into X=t,t+1,t+2,t+3 and Y=t+4
time_step = 30
X_train, y_train = create_dataset(train_data, time_step)
X_test, ytest = create_dataset(test_data, time_step)
    # reshape input to be [samples, time steps, features] which is required for LSTM
X_train=X_train.reshape(X_train.shape[0],X_train.shape[1] , 1)
X_test = X_test.reshape(X_test.shape[0],X_test.shape[1] , 1)
    # Create the BILSTM model
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Dense
    from tensorflow.keras.layers import LSTM
    from tensorflow.keras.layers import Bidirectional
    model = Sequential()
model.add(Bidirectional(LSTM(250, input_shape=(1, 30))))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam')

```

```

model.fit(X_train,y_train,validation_data=(X_test,ytest),epochs=10,batch_size=64,verbose=1)
import tensorflow as tf
train_predict=model.predict(X_train)
test_predict=model.predict(X_test)
#Transformback to original form
train_predict=scaler.inverse_transform(train_predict)
test_predict=scaler.inverse_transform(test_predict)
#### Calculate RMSE performance metrics
import math
from sklearn.metrics import mean_squared_error
math.sqrt(mean_squared_error(y_train,train_predict))
#### Test Data RMSEmath.sqrt(mean_squared_error(ytest,test_predict))
#### Plotting
# shift train predictions for plotting
look_back=30
trainPredictPlot = numpy.empty_like(df1)
trainPredictPlot[:, :] = np.nan
trainPredictPlot[look_back:len(train_predict)+look_back, :] = train_predict
# shift test predictions for plotting
testPredictPlot = numpy.empty_like(df1)
testPredictPlot[:, :] = numpy.nan
testPredictPlot[len(train_predict)+(look_back*2)+1:len(df1)-1, :] = test_predict
# plot baseline and predictions
st.markdown("<h1 style='text-align: center; color:black ;background-color:powderblue;font-size:14pt'> ✖ TRAIN AND TEST DATA ✖ </h1>", unsafe_allow_html=True)

#plt.plot(scaler.inverse_transform(df1))
plt.plot(scaler.inverse_transform(df1), color="blue", linewidth=1, linestyle="-")
plt.xlabel('No. of hours')
# Set the y axis label of the current axis.
plt.ylabel('Power generated by system | (kW)')
plt.plot(trainPredictPlot,label='Train Data',color="black",linewidth=2, linestyle="--")
plt.plot(testPredictPlot,label='Test Data',color="orange",linewidth=2, linestyle="--")
plt.legend(loc="upper left")
#plt.show()
st.pyplot(plt)

x_input=test_data[len(test_data)-30:].reshape(1,-1)
temp_input=list(x_input)
temp_input=temp_input[0].tolist()
# demonstrate prediction for next 24 hours
from numpy import array
lst_output=[]
n_steps=30
i=0

```

```

        while(i<24):
            if(len(temp_input)>30):
                #print(temp_input)
            x_input=np.array(temp_input[1:])
            x_input=x_input.reshape(1,-1)
            x_input = x_input.reshape((1, n_steps, 1))
            yhat = model.predict(x_input, verbose=0)
            temp_input.extend(yhat[0].tolist())
            temp_input=temp_input[1:]
            lst_output.extend(yhat.tolist())
            i=i+1
        else:
            x_input = x_input.reshape((1, n_steps,1))
            yhat = model.predict(x_input, verbose=0)
            print(yhat[0])
            temp_input.extend(yhat[0].tolist())
            print(len(temp_input))
            lst_output.extend(yhat.tolist())
            i=i+1

    print(lst_output)
    day_new=np.arange(1,31)
    day_pred=np.arange(len(df1),len(df1)+24)
    import matplotlib.pyplot as plt
    print(len(df1))
    progress=st.progress(0)
    for i in range(100):
        time.sleep(0.1)
        progress.progress(i+1)
        st.balloons()
        st.markdown("<h1 style='text-align: center; color:black ;background-color:powderblue;font-size:14pt'> ❌ PREDICTED RESULTS FOR NEXT 24 HOURS ❌ </h1>",
        unsafe_allow_html=True)
        plt.plot(day_pred,scaler.inverse_transform(lst_output),color="green",linewidth=1.5, linestyle="--",marker='*',markerfacecolor='yellow', markersize=7)
        plt.legend('GTTP',loc="upper left")

    plt.xlabel('No. of hours')
    # Set the y axis label of the current axis.
    plt.ylabel('Power generated by system | (kW)')

    st.pyplot(plt)
    st.markdown("<h1 style='text-align: center; color:black ;background-color:yellow;font-size:14pt'> 🟡 G-Given Data, \n 🟡 T-Train Data, \n 🟡 T-Test Data, \n 🟡 P-Predicted Results</h1>", unsafe_allow_html=True)

```

```
st.write(scaler.inverse_transform(lst_output))
```

```
if __name__ == "__main__":  
    main()
```

➤ **model.py**

```
import pandas as pd  
import datetime  
import numpy as np  
from keras.models import Sequential  
from keras.layers import Dense  
from keras.layers import LSTM  
from keras.layers import Bidirectional  
import pandas as pd  
import keras  
''' Loading data '''  
df = pd.read_excel('Dataset.csv')  
df=df.drop(columns=['DateTime'])  
''' Cleaning Data '''  
#dataframe.drop['Date'].values  
df['Power generated by system | (kW)'].replace(0, np.nan, inplace=True)  
df['Power generated by system | (kW)'].fillna(method='ffill', inplace=True)  
  
X = df.drop(columns=['Power generated by system | (kW)'])  
Y = df[['Power generated by system | (kW)']]  
X=np.array(X).reshape(-1,1,4)  
Y=np.array(Y).reshape(-1,1,1)  
  
model = Sequential()  
model.add(Bidirectional(LSTM(100, activation='relu',input_shape=(-1,1,4))))  
model.add(Dense(1))  
model.compile(loss='mae', optimizer='adam',metrics=['accuracy'])  
model.fit(X, Y,epochs=100,callbacks=[keras.callbacks.EarlyStopping(patience=3)])  
  
test_data = np.array([[ -4.858,0.989741,6.651,273]])  
o=model.predict(test_data.reshape(-1,1,4), batch_size=1)  
print(o)  
  
# Saving model to disk  
models=model.save('model.h5')
```