

REPORT ON
Wind Speed Forecasting using CNN Model: Building, Deploying and Automation

Dhinakar.S.P

B. Tech CSE (Cyber Security) – Shiv Nadar University, Chennai

Submitted to: **National Institute of Wind Energy,
Chennai, Tamil Nadu 600100.**



Mentor

1. SHRI.YELCHURI SRINATH
Assistant Director (Technical)

Date of Submission:

Signature of Candidate

Signature of Mentor

ACKNOWLEDGEMENT

First, I would like to thank **SHRI.S. ARULSELVAN** (*Assistant Executive Engineer*) for giving me the opportunity to do an internship within the organization.

I am highly indebted to **Dr.P. KANAGAVEL** (*Director & Division Head*) for the facilities provided to accomplish this internship.

It is indeed with a great sense of pleasure and immense sense of gratitude that I acknowledge the help of my Mentor **SHRI.YELCHURI SRINATH** Assistant Director (**Technical**) for the successful completion of this internship.

I also would like to thank all the people who worked with me for their patience and openness, as they created an enjoyable working environment.

I am extremely grateful to my department staff members and friends who helped me with the successful completion of this internship.

Dhinakar.S.P

LEARNING OBJECTIVES / INTERNSHIP **OBJECTIVES**

- Internships are generally thought of to be reserved for college students looking to gain experience in a particular field.
- However, a wide array of individuals can benefit from Training Internships to receive real world experience and develop their skills.
- An objective for this position should emphasize the skills already known in the area and the interest in learning more.
- Internships are utilized in several different career fields, including architecture, engineering, healthcare, economics, advertising and many more.
- Some internships are used to allow individuals to perform scientific research while others are specifically designed to allow people to gain first-hand experience working.
- Utilizing internships is a great way to build a resume and develop skills that can be emphasized in the resume for future jobs.
- When applying for a Training Internship, highlighting special skills or talents, have an improved chance of landing the position.

DECLARATION

This thesis work entitled “**Wind Data Forecasting using CNN Model: Building, Deploying and Automation**” is my own work carried out under the guidance of **SHRI.YELCHURI SRINATH Assistant Director (Technical)**. This work in the same form or in any other form is not submitted by me or by anyone else for the award of any degree.

TABLE OF CONTENTS

<u>Serial. No</u>	<u>Name</u>	<u>Page Number</u>
1.	Nomenclatures	6
2.	Abstract	8
3.	Background	9
4.	Introduction	15
5.	Literature Survey	16
6.	Methodology	18
7.	System Implementation	34
8.	Software and Tools Used	39
9.	Result	43
10.	Conclusion	50
11.	References	51

I: NOMENCLATURES

h5 (HDF5) - A file format and set of tools for managing complex data, used for storing copious amounts of data efficiently. h5 files are used to store trained machine learning models.

na ma Interpolation (Moving Average) - A technique for filling missing values in time series data by first performing linear interpolation and then applying a moving average to smooth the interpolated values.

Kalman Interpolation - A method for estimating and filling missing values in a dataset by applying a Kalman Filter, which uses observed data trends to predict missing values.

TensorFlow -An open-source machine learning framework developed by Google, used for building, and deploying machine learning models, including neural networks.

Keras - An open-source neural network library written in Python, which runs on top of TensorFlow. It provides a user-friendly interface for building and training deep learning models.

TensorBoard - A visualization tool included with TensorFlow that allows users to monitor and visualize metrics such as loss and accuracy during the training of machine learning models.

CSV (Comma-Separated Values) - A file format used to store tabular data, where each line represents a row of data, and each value is separated by a comma. CSV files are commonly used for importing and exporting data in a simple, human-readable format.

NumPy - A fundamental library for scientific computing in Python, providing support for arrays, matrices, and a large collection of mathematical functions to operate on these data structures.

Pandas - An open-source data analysis and manipulation library for Python, offering data structures like DataFrame and Series to handle and analyse structured data efficiently.

Matplotlib - A comprehensive library for creating static, animated, and interactive visualizations in Python. It is widely used for generating plots, histograms, bar charts, and other graphical representations of data.

Docker - A platform for developing, shipping, and running applications in containers. Containers encapsulate an application and its dependencies, ensuring consistency across different environments.

Streamlit - An open-source app framework for creating and sharing data science and machine learning applications quickly. It allows for the creation of interactive web applications using Python scripts.

GitHub - A web-based platform for version control and collaboration, allowing multiple people to work on projects together. It uses Git for source code management and provides features like repositories, pull requests, and issues.

Workflow.yml - A configuration file used in GitHub Actions to define automated workflows. It specifies the triggers (such as pushes or pull requests), the jobs to run, and the steps within each job.

CNN (Convolutional Neural Network) - A class of deep neural networks commonly used for analysing visual data. CNNs consist of layers that automatically and adaptively learn spatial hierarchies of features from input.

II: ABSTRACT

This project develops a **wind speed forecasting system** using **meteorological data** from the **Jafrabad coast mast** (December 2018 - November 2019). It integrates **data processing, time series modelling, web application development, containerization, and automation** for comprehensive weather prediction.

Raw meteorological data is converted into **Excel files**, which are read into structured **Dataframes using pandas**. This enables efficient manipulation and extraction of relevant features such as **wind speed** and **direction**. **Data preprocessing** includes handling missing values with techniques like **moving average** and **Kalman filter interpolation**.

A **Convolutional Neural Network (CNN)** model is trained on the pre-processed data to forecast wind speed values, leveraging historical data to capture patterns and trends. Additional data quality checks, such as **flagging suspect data points** and **applying rate of change tests**, ensure robust input data.

To provide convenient access, a **Streamlit web application** is developed, allowing users to upload data and retrieve forecasted wind speeds. **Docker containerization** packages the web application and its dependencies into a portable container, ensuring consistency across different computing environments.

Automation via GitHub Actions facilitates continuous integration and deployment. This ensures updates to the codebase trigger workflows that process data, train the model, and deploy the web application, streamlining development and maintenance.

The project's results showcase the successful application of **advanced data analysis, machine learning modelling, and automation** to weather forecasting. Visualization and interpretation of forecasted values offer users insights into predicted trends and patterns, aiding informed decision-making in industries reliant on weather predictions.

Overall, this project advances predictive modelling in meteorology, offering a practical solution for wind speed forecasting and empowering stakeholders with accessible decision support tools.

III: BACKGROUND

3.1 Wind Energy: Harnessing the Power of Nature

In the evolving renewable energy landscape, wind stands tall as a sustainable power source. Harnessing the kinetic energy of the wind, wind turbines have become iconic symbols of clean energy initiatives worldwide.

3.1.1 Understanding Wind Energy

Wind energy, a renewable energy subset, relies on converting wind's kinetic energy into electricity. Turbines equipped with aerodynamic blades capture the energy, setting a generator in motion that transforms it into a readily usable power source. This process's efficiency and reliability make wind energy a cornerstone in the quest for greener and more sustainable power solutions.

3.1.2 The Importance of Data in Wind Energy

Optimizing wind energy systems requires accurate and comprehensive wind-related data. Data on wind speed, direction, and other meteorological variables form the foundation for informed decision-making. However, real-world scenarios often introduce missing values, creating challenges in data integrity.

3.1.3 Machine Learning for Data Integrity

Our project explores machine learning to predict and fill missing values within large wind datasets. This endeavour not only improves analysis accuracy but also enhances wind energy management systems' efficiency.

3.1.4 Toward a Sustainable Future

Combining technology, data science, and renewable energy underscores our commitment to a sustainable future. By addressing challenges in wind data, we enhance the reliability and effectiveness of wind energy solutions.

1.2 Exploring Wind Data Collection Methods

Wind energy demands precise and reliable data for effective harnessing. This section explores various methodologies employed for collecting wind data.

3.2.1 Anemometers: Measuring Wind Speed and Direction

Anemometers have been instrumental in capturing wind speed and direction for decades. Installed atop meteorological towers, these devices offer real-time insights into wind patterns, making them indispensable in the wind energy industry.

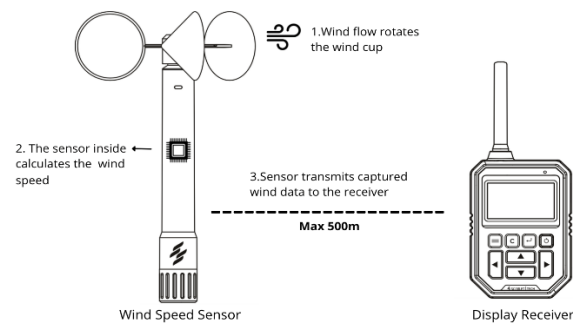


Fig.3.1.1

3.2.2 Lidar Technology: Precision Wind Measurement

Lidar (Light Detection and Ranging) technology measures wind speed at different altitudes with remarkable precision. Lidar's versatility makes it an ideal choice for offshore wind farms and complex terrains.

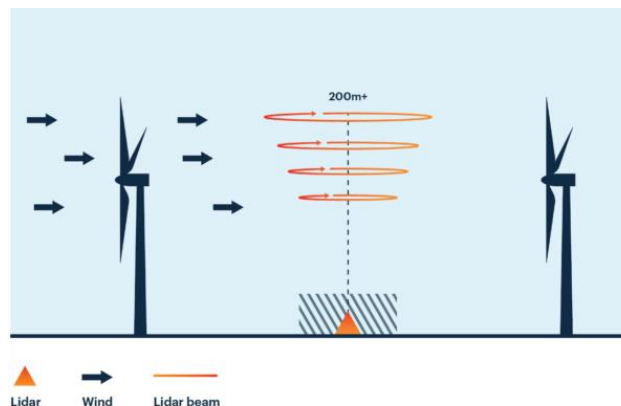


Fig.3.1.2

3.2.3 SODAR: Using Sound Waves to Measure Wind

SODAR (Sonic Detection and Ranging) devices use sound waves to measure wind profiles in the atmosphere. By analysing the Doppler shift in returned signals, SODAR provides valuable data on wind speed and direction at varying heights.



Fig. 3.1.3

3.2.4 Windmills: Dual Purpose Generators

Windmills, primarily designed for energy generation, also serve as data collectors. Monitoring the rotational speed of turbine blades offers insights into local wind conditions, contributing to both energy production and data acquisition.

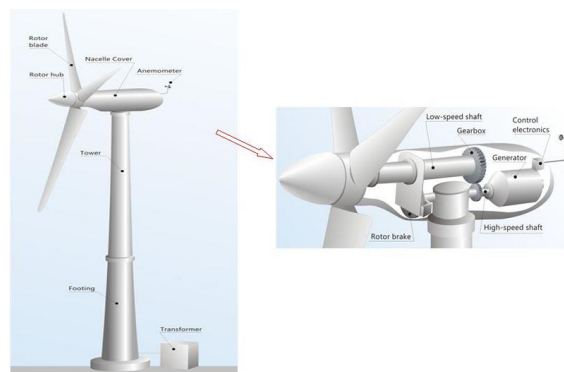


Fig.3.1.4

3.2.5 Remote Sensing: Satellite Wind Data

Satellite technology enables remote sensing of wind data over vast areas. This global perspective supports macro-level analyses, aiding large-scale planning and decision-making.

3.2.6 Combining Data Collection Methods

Integrating data from anemometers, Lidar devices, SODAR instruments, windmills, and remote sensing creates a comprehensive understanding of the wind environment, facilitating robust predictions and informed decisions.

3.2.7 Future Innovations in Wind Data Collection

As technology evolves, so does wind data collection. AI-powered sensors and drone-based data collection promise sophisticated methods for understanding wind patterns.

3.3 Time Series Data Analysis in Wind Forecasting

Time series data, a sequential collection of observations measured over time, offers a rich narrative in various domains. This section explores time series data's significance, challenges, and innovative approaches.

3.3.1 Defining Time Series Data

Time series data portrays how phenomena evolve over time. Each data point is part of a continuous story, offering unique challenges and opportunities.

3.3.2 Challenges in Analysing Time Series Data

Analysing time series data involves dealing with temporal dependencies, seasonality, and trends. Missing values can significantly impact analysis accuracy and predictions.

3.3.3 Addressing Data Gaps

Our project addresses the challenge of missing values in large time series datasets. We leverage machine learning techniques tailored for time series analysis to predict and fill in missing data

3.3.4 Machine Learning in Time Series

Machine learning models reveal patterns and insights in time series data that traditional methods might miss. We employ advanced algorithms to predict missing values and uncover latent patterns and trends.

3.3.5 Future Prospects in Time Series Analysis

Our venture into time series data analysis aims to unlock its latent potential. By unraveling the mysteries of time series data, we pave the way for informed decision-making and deeper process understanding.

Weather Forecasting and Its Relevance in This Project

Weather forecasting has been crucial throughout human history, aiding in agriculture, navigation, and planning outdoor events. Modern technology significantly enhances our ability to collect, analyse, and forecast weather data accurately.

Meteorologists now collect vast amounts of data from satellites, weather stations, and ocean buoys. NetCDF (Network Common Data Form) files store multidimensional scientific data, including meteorological data, efficiently and flexibly, facilitating analysis and modeling tasks.

Our project focuses on wind speed forecasting using data from the Jafrabad coast mast (December 2018 to November 2019). By leveraging historical meteorological data, we aim to develop a model providing accurate wind speed predictions. This targeted approach enables stakeholders in renewable energy, aviation, and agriculture to make informed decisions.

We chose a Convolutional Neural Network (CNN) model for its effectiveness in capturing short-term fluctuations and long-term trends in time series data. CNN models handle complex patterns, making them well-suited for analysing meteorological data with distinct seasonal patterns.

Deploying the forecasting model using Streamlit and Docker modernizes predictive model accessibility and scalability. Streamlit, a lightweight web application framework in Python, and Docker, with containerization capabilities, ensure consistency and portability across computing environments.

Automation using GitHub Actions facilitates continuous integration and deployment. This automation triggers workflows to process data, train the model, and deploy the web application, streamlining development and maintenance.

Overall, our project demonstrates advanced data analysis, machine learning techniques, and automation in solving real-world challenges. By harnessing meteorological data and modern computing technologies, we aim to improve weather forecasting accuracy and empower decision-makers across industries.

IV: INTRODUCTION

Wind energy, a pivotal player in the renewable energy landscape, harnesses the power of the wind to generate electricity. As a clean and sustainable source, wind energy significantly contributes to global climate change mitigation efforts. Accurate data on wind speed and direction is crucial for effective wind energy initiatives. Our project focuses on predicting missing values in a comprehensive wind dataset to enhance the reliability of wind energy analytics.

The project's goal is to develop a wind speed forecasting model for the Jafrabad coast from December 2018 to November 2019 using historical data. The dataset, consisting of raw meteorological data, is converted into Excel files and structured Dataframes using pandas for efficient processing. This includes extracting key features such as wind speed and direction and addressing missing values with techniques like moving average interpolation and Kalman filter interpolation.

Key steps in the methodology include data preprocessing, building a Convolutional Neural Network (CNN) model, developing a Streamlit web application, Docker containerization, and automation with GitHub Actions. Data preprocessing ensures the dataset is clean and suitable for modelling. The CNN model uses historical data to identify patterns and predict wind speed accurately. Data quality checks enhance the input data's robustness.

The Streamlit web application allows users to upload data and obtain forecasted wind speed values, providing an accessible interface. Docker containerization ensures the application's scalability and portability across different environments. Automation through GitHub Actions enables continuous integration and deployment, streamlining updates and maintenance.

By integrating these components, the project demonstrates the practical application of data science, machine learning, and automation in solving real-world problems. The forecasted wind speed data empowers stakeholders in various industries to make informed decisions and optimize operations based on accurate weather predictions. The project advances predictive modelling in meteorology, offering a practical solution for wind speed forecasting and empowering stakeholders with accessible decision-support tools.

V: LITERATURE REVIEW

The literature review for our project encompasses a diverse range of studies and techniques pertinent to wind speed forecasting, meteorological data analysis, time series forecasting, and machine learning deployment. Prior research has extensively explored various methodologies for predicting wind speed, including both statistical models and machine learning algorithms. Traditional statistical methods, such as ARIMA (Autoregressive Integrated Moving Average), have been widely used for weather forecasting. These methods rely on historical data and mathematical models to predict future outcomes. However, they often struggle with capturing the complex and non-linear patterns inherent in meteorological data, including seasonal variations and long-term trends.

In recent years, machine learning techniques have emerged as promising alternatives for enhancing weather forecast accuracy. The Convolutional Neural Network (CNN) model is particularly notable for its ability to capture intricate patterns and trends in time series data. Unlike traditional models, CNNs can leverage deep learning to model the spatial and temporal dependencies in meteorological data, resulting in more accurate predictions.

Additionally, the implementation of machine learning models as web applications using frameworks like Streamlit has gained popularity. Streamlit offers a user-friendly and interactive platform for deploying predictive models, allowing users to upload data and retrieve forecasted values easily.

Docker containerization provides a scalable and portable solution for deploying web applications. By encapsulating the application and its dependencies into a Docker container, developers can ensure consistency and reliability across different computing environments. This approach simplifies deployment and maintenance, allowing applications to run seamlessly on various platforms with minimal configuration.

Automation through GitHub Actions facilitates continuous integration and deployment, streamlining the workflow from data processing to model deployment. This ensures that updates to the codebase trigger automated processes for data handling, model training, and application deployment, enhancing efficiency and reducing manual intervention.

Overall, the literature review underscores the importance of leveraging advanced data analysis and machine learning techniques in wind speed forecasting. By building on existing methodologies and integrating modern technologies, our project aims to improve the accuracy and reliability of wind speed predictions. This, in turn, benefits various industries reliant on accurate weather forecasts, contributing to informed decision-making and operational optimization.

VI: METHODOLOGY

6.1 Data Collection & Description

The wind speed forecasting project utilizes a comprehensive dataset collected from the Jafrabad offshore wind mast, spanning from December 2018 to November 2019. This dataset, comprising 52,561 rows and 13 features, is essential for developing and validating the forecasting model. The raw data was recorded and then converted to Excel and structured DataFrames for efficient manipulation and analysis.

Dataset Overview:

- **Location and Instrumentation:**
 - **Latitude:** N 20° 53' 29.810"
 - **Longitude:** E 71° 27' 35.680"
 - **Elevation:** 9 meters
 - **Anemometer Height:** 10 meters (commissioned in February 2019)
- **Flags and Anomalies:**
 - Data includes unflagged entries and excludes flags such as icing and invalid data.
 - Notable issues: Erroneous data from the 100m_S anemometer sensor starting from July 2019 and from the temperature sensor.
- **Calm Threshold:** Set at 0 m/s to distinguish between calm and active wind conditions.

Wind Speed Features:

1. **100m_N Avg [m/s]:** The average wind speed measured at 100 meters height on the north side in meters per second.
2. **100m_S Avg [m/s]:** The average wind speed measured at 100 meters height on the south side in meters per second.
3. **80m Avg [m/s]:** The average wind speed measured at 80 meters height in meters per second.
4. **50m Avg [m/s]:** The average wind speed measured at 50 meters height in meters per second.
5. **20m Avg [m/s]:** The average wind speed measured at 20 meters height in meters per second.
6. **10m Avg [m/s]:** The average wind speed measured at 10 meters height in meters per second.

Atmospheric Features:

1. **Pressure [mbar]:** The atmospheric pressure at the measurement location in millibars. Pressure changes can affect wind patterns.
2. **Temp 5m [°C]:** The temperature at 5 meters height in degrees Celsius. Temperature influences air density and wind speed.
3. **Hum 5m [%]:** The humidity at 5 meters height in percentage. Humidity can affect air density and thermal properties of the atmosphere.

Wind Direction Features:

1. **98m WV [°]:** The wind direction at 98 meters height in degrees. Wind direction is typically measured in degrees from true north (0°).
2. **78m WV [°]:** The wind direction at 78 meters height in degrees.
3. **48m WV [°]:** The wind direction at 48 meters height in degrees.

The dataset's detailed structure, with measurements at various heights and intervals, provides a robust foundation for the wind speed forecasting model. This offshore lidar raw data is critical for accurately capturing wind patterns and improving the reliability of wind energy analytics.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Date/Time	100m_NAvg[m/s]	100m_SAvg[m/s]	80m Avg[m/s]	50m Avg[m/s]	20m Avg[m/s]	10m Avg[m/s]	Pressure 5m [mbar]	98m WV [°]	78m WV [°]	48m WV [°]	Temp 5m [°C]	Hum 5m	
2	01-12-2018 00:00	10.654	10.809	5.182	7.096	4.852		1012.2	52	45	29.5	22.5	46.5	
3	01-12-2018 00:10	10.531	10.688	5.334	7.415	4.953		1012.1	53.4	47.8	32.3	22.2	47.5	
4	01-12-2018 00:20	9.976	10.11	5.319	7.398	4.796		1012.2	53.4	46.4	29.5	22.1	48.3	
5	01-12-2018 00:30	9.617	9.735	5.319	7.43	4.749		1012.1	52	46.4	30.9	22.2	48.5	
6	01-12-2018 00:40	9.001	9.112	5.363	7.344	4.833		1012	53.4	47.8	32.3	22.2	49	
7	01-12-2018 00:50	8.841	8.967	5.354	7.496	4.813		1012	52	46.4	33.8	22.2	49.3	
8	01-12-2018 01:00	8.228	8.345	5.216	7.172	4.702		1011.9	49.2	43.6	28.1	22.2	49.3	
9	01-12-2018 01:10	8.628	8.756	5.454	7.126	4.857		1011.8	49.2	40.8	25.3	22.4	49.2	
10	01-12-2018 01:20	8.679	8.808	5.554	6.48	3.817		1011.8	47.8	39.4	22.5	22.4	49.4	
11	01-12-2018 01:30	8.964	9.087	5.895	5.564	3.077		1011.7	47.8	35.2	23.9	21.6	51.6	
12	01-12-2018 01:40	9.35	9.472	6.047	6.058	3.529		1011.6	49.2	39.4	25.3	21.5	51.8	
13	01-12-2018 01:50	9.568	9.705	6.128	5.999	3.19		1011.6	47.8	39.4	26.7	21.4	52	
14	01-12-2018 02:00	9.671	9.798	6.282	5.542	2.622		1011.5	47.8	40.8	25.3	20.7	53.7	
15	01-12-2018 02:10	9.254	9.409	6.29	5.306	2.541		1011.4	45	38	23.9	20.2	55.1	
16	01-12-2018 02:20	9.377	9.524	6.245	5.446	2.58		1011.3	46.4	39.4	23.9	19.8	56.4	
17	01-12-2018 02:30	9.401	9.541	6.236	6.256	2.915		1011.2	46.4	39.4	25.3	19.7	57	
18	01-12-2018 02:40	9.802	9.955	6.248	6.185	3.003		1011	45	36.6	22.5	19.7	56.8	
19	01-12-2018 02:50	9.973	10.132	6.28	5.932	3.023		1010.9	45	35.2	21.1	19.6	56.6	
20	01-12-2018 03:00	10.349	10.531	6.334	6.482	3.633		1010.8	47.8	39.4	23.9	19.8	55.4	
21	01-12-2018 03:10	10.106	10.257	6.057	6.193	2.971		1010.7	46.4	38	22.5	20.2	54.1	
22	01-12-2018 03:20	9.91	10.053	6.228	5.601	2.656		1010.7	46.4	38	21.1	19.7	55.8	
23	01-12-2018 03:30	9.831	9.982	6.152	5.697	2.499		1010.7	45	38	21.1	19.6	56.2	

Fig 6.1.1 Sample Dataset Snapshot

6.2 Data Preprocessing

In preparing the wind speed dataset for forecasting, several data processing steps were undertaken to ensure the accuracy and reliability of the model.

Renaming and Dropping Columns:

To streamline the dataset and avoid redundancy, the column 100m_N Avg [m/s] was renamed to 100m Avg [m/s]. Since the dataset contained wind speed measurements from two sensors at the same 100-meter height, the 100m_S Avg [m/s] column was dropped. This decision was made to eliminate unnecessary duplication and simplify the analysis process.

Code Snippet:

```
df.rename(columns={'100m_N Avg [m/s]': '100m Avg[m/s]'}, inplace=True)

# Drop the '100m_S Avg [m/s]' column
df.drop(columns=['100m_S Avg [m/s]'], inplace=True)

print("\nDataFrame after renaming '100m_N Avg [m/s]' to '100m Avg[m/s]' and dropping '100m_S Avg [m/s]' column:")
print(df)
```

Fig.6.2.1

Data Imputation:

1. Kalman Interpolation for Wind Speed

To address missing values in wind speed measurements (100m Avg[m/s], 80m Avg [m/s], 50m Avg [m/s], 20m Avg [m/s], 10m Avg [m/s]), we employed the Kalman filter. This method is particularly suitable for wind speed data due to its ability to handle dynamic patterns and noise, providing smoothed estimates based on historical observations.

Code Snippet:

```
from pykalman import KalmanFilter
def kalman_interpolation(df, feature):
    # Extract the series with missing values
    values = df[feature].values

    # Create a Kalman Filter with default parameters
    kf = KalmanFilter(initial_state_mean=0, n_dim_obs=1)

    # Fit the Kalman Filter to the observed values
    values = np.ma.masked_invalid(values)
    kf = kf.em(values, n_iter=10)

    # Apply the Kalman Filter to fill missing values
    smoothed_values, _ = kf.smooth(values)

    # Return the filled series, reshaped to 1D
    return pd.Series(smoothed_values.flatten(), index=df.index)

# Apply Kalman interpolation for each feature with missing values
features_with_missing_values = ['100m Avg [m/s]', '80m Avg [m/s]', '50m Avg [m/s]', '20m Avg [m/s]', '10m Avg [m/s]']
for feature in features_with_missing_values:
```

Fig.6.2.2

2. Moving Average Interpolation for Atmospheric Variables

For atmospheric variables such as pressure, temperature, and humidity (Pressure 5m [mbar], Temp 5m [°C], Hum 5m), we applied a moving average (MA) interpolation technique. After initial linear interpolation, the MA approach helped to smooth out fluctuations in the data, ensuring that the imputed values captured both short-term variations and long-term trends.

Code Snippet:

```
def na_ma_interpolation(df, feature, window=5):
    # Interpolate missing values using linear interpolation first
    df[feature] = df[feature].interpolate(method='linear', limit_direction='both')

    # Apply moving average to smooth the interpolated values
    df[feature] = df[feature].rolling(window=window, min_periods=1).mean()

    return df[feature]

# Features to interpolate with MA interpolation
ma_features = ['Pressure 5m [mbar]', 'Temp 5m [°C]', 'Hum 5m']
```

Fig.6.2.3

3. Mean Values for Wind Direction

Ensure consistency and completeness in wind direction measurements (98m WV [°], 78m WV [°], 48m WV [°]), we computed the mean values.

Data Flagging:

Primary tests: range test, climatology test, rate of change test, and flat line test. Each test was designed to identify and flag potentially erroneous data points such that the error data can be removed later if necessary.

1. Range Test

The range test checks if the wind speed, wind direction, and temperature values fall within specified sensor and user-defined limits. Flags are assigned based on the results:

- **Fail (4):** Values outside sensor limits.
- **Suspect (3):** Values within sensor limits but outside user-defined limits.
- **Pass (1):** Values within user-defined limits

Data Type	Sensor Min	Sensor Max	User Min	User Max	Flagging Condition
Wind Speed (Avg [m/s])	0	30	0	40	Flag if outside range
Wind Direction (WV [°])	0	360	0	360	Flag if outside range
Temperature (Temp [°C])	0	50	0	50	Flag if outside range

Table 6.2.1

Code Snippet:

```
def flag_wind_speed(wind_speed_cols, sensor_min, sensor_max, user_min, user_max, df_round):
    flags = []
    for col in wind_speed_cols:
        wind_speed = df_round[col]
        if (wind_speed < sensor_min).any() or (wind_speed > sensor_max).any():
            flags.append(4) # Fail
        elif (wind_speed < user_min).any() or (wind_speed > user_max).any():
            flags.append(3) # Suspect
        else:
            flags.append(1) # Pass
    return flags

def flag_wind_direction(wind_direction_cols, sensor_min, sensor_max, user_min, user_max, df_round):
    flags = []
    for col in wind_direction_cols:
        wind_direction = df_round[col]
        if (wind_direction < sensor_min).any() or (wind_direction > sensor_max).any():
            flags.append(4) # Fail
        elif (wind_direction < user_min).any() or (wind_direction > user_max).any():
            flags.append(3) # Suspect
        else:
            flags.append(1) # Pass
    return flags

sensor_min_wind_speed = 0
sensor_max_wind_speed = 30
user_min_wind_speed = 0
user_max_wind_speed = 40
```

Fig.6.2.4

2. Climatology Test

The climatology test ensures that temperature values are within realistic historical limits for the given region.

Data Type	Min Temp	Max Temp	Flagging Condition
Temperature (Temp[°C])	0	50	Flagging if outside historical range

Table 6.2.2

Code Snippet:

```
def check_temperature(temperature_cols, min_temp, max_temp, df_round):
    flags = []
    for col in temperature_cols:
        T = df_round[col]
        if (T < min_temp).any() or (T > max_temp).any():
            flags.append(3) # Fail
        else:
            flags.append(1) # Pass
    return flags

temperature_cols = ["Temp 5m [°C]"]
min_temp = 0 # Lowest in Gujarat = 1.7 C
max_temp = 50 # Highest in Gujarat = 47 C

temperature_flags = check_temperature(temperature_cols, min_temp, max_temp, df_round)
print("Flags for the temperature:", temperature_flags)
```

Fig.6.2.5

3. Rate of Change Test

The rate of change test identifies sudden and unrealistic changes in wind speed. Flags are assigned based on deviations exceeding a specified number of standard deviations over a given period.

Data Type	N_DEV	TIM_DEV	Flagging Condition
Wind Speed (Avg [m/s])	3	5	Flag if change > N_DEV * rolling standard deviation

Table.6.2.3

Code Snippet:

```
def rate_of_change_test(wind_data, features, N_DEV, TIM_DEV):
    for feature in features:
        # Ensure the feature exists in the DataFrame
        if feature not in wind_data.columns:
            print(f"Feature '{feature}' not found in the DataFrame.")
            continue

        # Calculate the first differences of wind speed for the feature
        wind_data[f'{feature}_diff'] = wind_data[feature].diff()

        # Calculate the rolling standard deviation of the first differences
        wind_data[f'{feature}_std_diff'] = wind_data[f'{feature}_diff'].rolling(window=TIM_DEV).std()

        # Initialize all flags to 1 (Pass)
        wind_data[f'{feature}_flag'] = 1

        # Determine the suspect condition based on the threshold
        suspect_condition = wind_data[f'{feature}_diff'].abs() > (N_DEV * wind_data[f'{feature}_std_diff'])
        wind_data.loc[suspect_condition, f'{feature}_flag'] = 3 # Set flag to 3 (Suspect) where condition is met

        # Drop intermediate calculation columns if desired
        wind_data.drop(columns=[f'{feature}_diff', f'{feature}_std_diff'], inplace=True)

    return wind_data

# Parameters
N_DEV = 3 # Number of standard deviations
TIM_DEV = 5 # Period over which the standard deviation is calculated

# Features to apply the test
features = ['100m Avg [m/s]', '80m Avg [m/s]', '50m Avg [m/s]', '20m Avg [m/s]', '10m Avg [m/s]']
```

Fig .6.2.6

4. Flat Line Test

The flat line test detects prolonged periods where the data does not change, indicating possible sensor malfunction or data recording issues.

Data Type	Tolerance	Fail Repetitions	Suspect Repetitions	Flagging Condition
Wind Speed (Avg [m/s])	0.001	10	3	Flag if no change within tolerance
Wind Direction (WV [°])	0.001	10	3	Flag if no change within tolerance

Table.6.2.4

Code Snippet:

```
# Define the feature names
feature_names = ['100m Avg[m/s]', '80m Avg [m/s]', '50m Avg [m/s]', '20m Avg [m/s]', '10m Avg [m/s]', '98m WV [°]', '78m WV [°]', '48m WV [°]']

# Define the tolerance and repetition counts
tolerance = 0.001
fail_rep_count = 10
suspect_rep_count = 3

# Define the evaluate_features function
def evaluate_features(observations, tolerance, fail_rep_count, suspect_rep_count):
    flags = {}

    for feature, obs in observations.items():
        if len(obs) < max(fail_rep_count, suspect_rep_count):
            flags[feature] = 1 # Not enough data for comparison, pass by default
            continue

        recent_obs = obs[-max(fail_rep_count, suspect_rep_count):]

        if all(abs(obs[-1] - prev_obs) < tolerance for prev_obs in recent_obs[-fail_rep_count:]):
            flags[feature] = 4 # Fail
        elif all(abs(obs[-1] - prev_obs) < tolerance for prev_obs in recent_obs[-suspect_rep_count:]):
            flags[feature] = 3 # Suspect
        else:
            flags[feature] = 1 # Pass

    return flags

# Create a dictionary to store the observations for each feature
observations = {feature: [] for feature in feature_names}
```

Fig.6.2.7

6.3 Exploratory Data Analysis:

1. Data Distribution of 100m

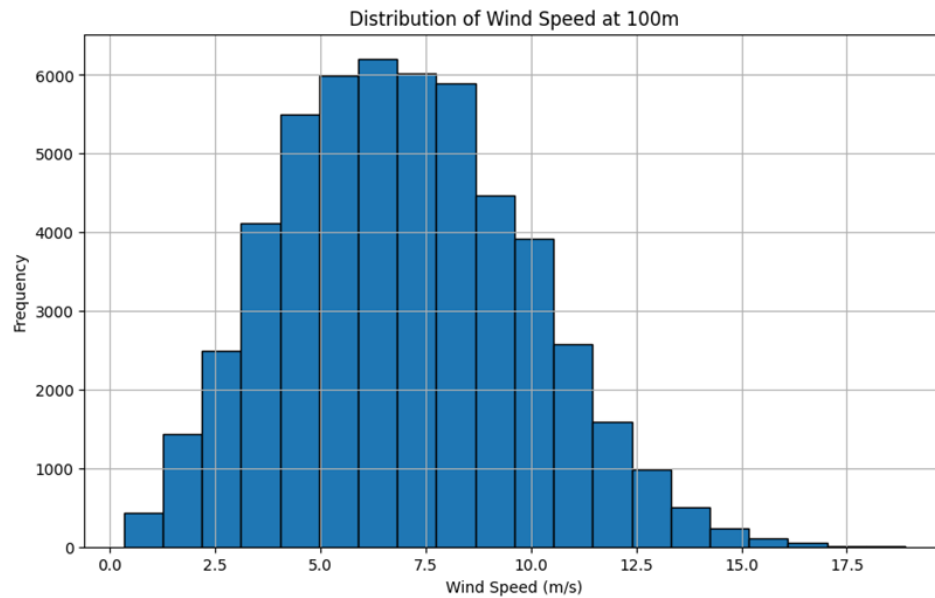


Fig.6.3.1

Code Snippet:

```
# Example code for distribution and summary statistics
import matplotlib.pyplot as plt

# Visualize distribution of wind speed at 100m
plt.figure(figsize=(10, 6))
plt.hist(df['100m Avg[m/s]'], bins=20, edgecolor='black')
plt.xlabel('Wind Speed (m/s)')
plt.ylabel('Frequency')
plt.title('Distribution of Wind Speed at 100m')
plt.grid(True)
plt.show()

# Calculate summary statistics for wind speed at 100m
print("Summary Statistics for Wind Speed at 100m:")
print(df['100m Avg[m/s]'].describe())
```

Fig.6.3.2

2. Correlation Analysis

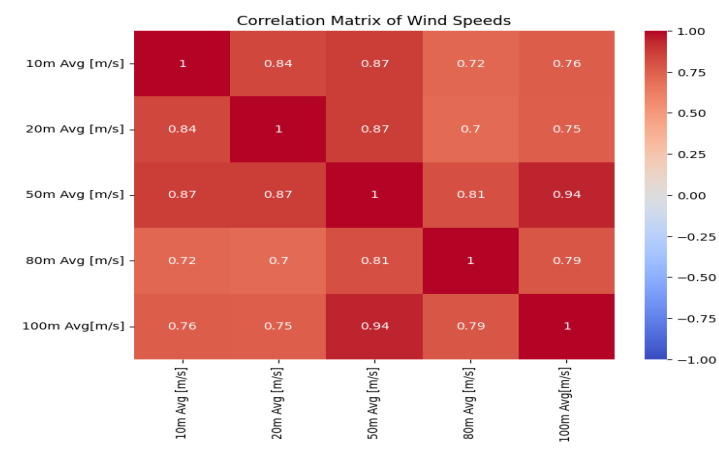


Fig.6.3.3

Code Snippet:

```
# Example code for correlation analysis
correlation_matrix = df[['10m Avg [m/s]', '20m Avg [m/s]', '50m Avg [m/s]', '80m Avg [m/s]', '100m Avg [m/s]']].corr()

# Plot heatmap of correlations
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Matrix of Wind Speeds')
plt.show()
```

Fig.6.3.4

6.4 Model Development

Model Selection

The selection of Convolutional Neural Network (CNN) architecture for wind speed forecasting was driven by the dataset's characteristics, which include multivariate time-series data collected from multiple heights and meteorological parameters. CNNs are particularly suited for this task due to their ability to capture spatial and temporal dependencies simultaneously.

Code Snippet:

```
# Define a function to create sequences
def create_dataset(data_x, data_y, look_back=10):
    data_X, data_Y = [], []
    for i in range(len(data_x) - look_back):
        a = data_x[i:(i + look_back)]
        data_X.append(a)
        data_Y.append(data_y[i + look_back - 1])
    return np.array(data_X), np.array(data_Y)

# Create the dataset with a look_back period
look_back = 10
train_size = int(len(data_x) * 0.8)
test_size = len(data_x) - train_size

train_x, test_x = data_x[:train_size], data_x[train_size:]
train_y, test_y = data_y[:train_size], data_y[train_size:]

train_X, train_Y = create_dataset(train_x, train_y, look_back)
test_X, test_Y = create_dataset(test_x, test_y, look_back)

# Reshape the data into 3D arrays for CNN
train_X = np.reshape(train_X, (train_X.shape[0], look_back, len(input_cols)))
test_X = np.reshape(test_X, (test_X.shape[0], look_back, len(input_cols)))

# Define a function to create the CNN model
def create_cnn_model(look_back, input_cols):
    model = Sequential()
    model.add(Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(look_back, len(input_cols))))
    model.add(Flatten())
    model.add(Dense(32, activation='relu'))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model
```

Fig.6.4.1

CNN Architecture

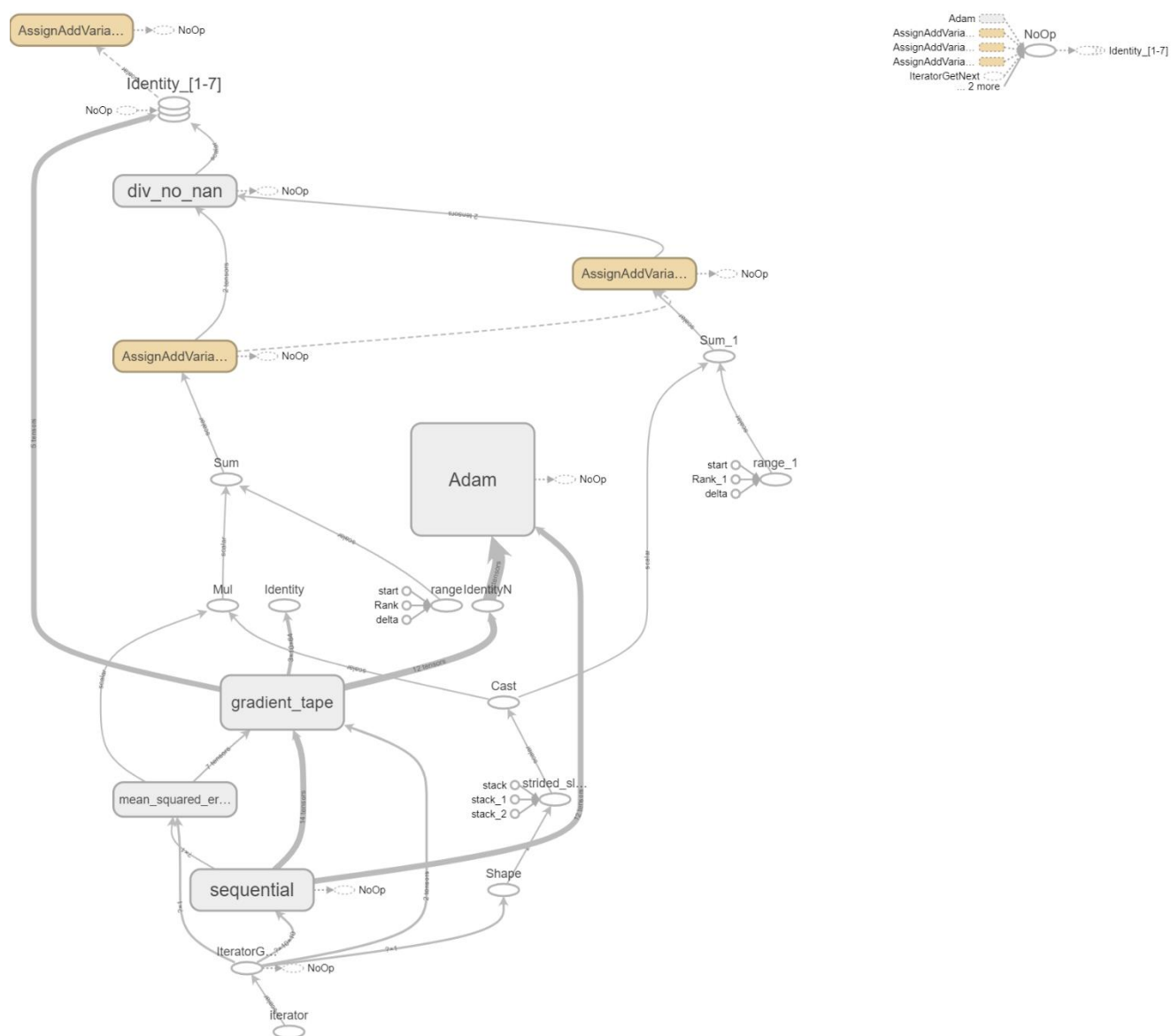


Fig.6.4.2

- **Convolutional Layer:** The model starts with a 1-dimensional convolutional layer (Conv1D) with 64 filters and a kernel size of 3. This layer is effective in capturing spatial patterns across the input time-series data.
- **Flattening Layer:** Following the convolutional layer, a Flatten layer reshapes the output into a one-dimensional vector, preparing it for fully connected layers.
- **Dense Layers:** Two fully connected (Dense) layers follow, with 32 neurons each and ReLU activation functions. These layers help in learning complex relationships in the data.
- **Output Layer:** The final Dense layer with a single neuron predicts the wind speed at the 100m height, trained to minimize mean squared error using the Adam optimizer.

Model Training:

The training phase ensures the model's effectiveness in predicting wind speeds accurately.

```
# Define a function to train the model
def train_model(model, train_X, train_Y, epochs, batch_size, validation_data, callbacks=None):
    model.fit(train_X, train_Y, epochs=epochs, batch_size=batch_size, validation_data=validation_data, callbacks=callbacks)
    return model

# Train the CNN model with TensorBoard and custom callback
cnn_model = train_model(cnn_model, train_X, train_Y, epochs=30, batch_size=15, validation_data=(test_X, test_Y), callbacks=[tensorboard_callback, CustomCallbac
```

Fig.6.4.3

The CNN model is trained for 30 epochs using a batch size of 15. During training, it learns to minimize the mean squared error loss function on the training data while monitoring performance on the validation set.

Tensorboard images of Training phase:

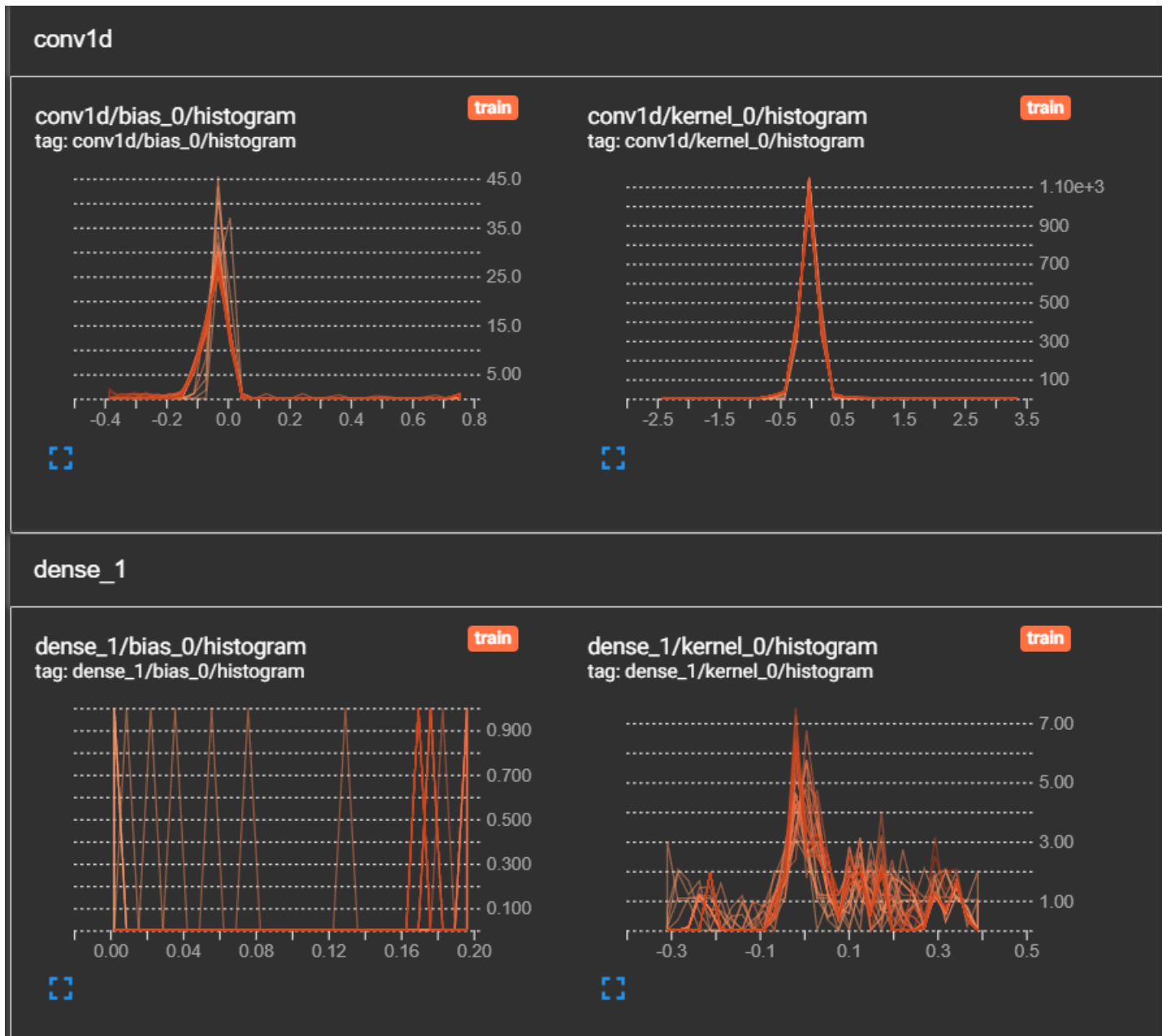


Fig.6.4.4

Validation and Evaluation:

After training, the model's performance is evaluated on the test set to assess its ability to generalize to unseen data. Metrics such as Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) are computed to quantify prediction accuracy

```
1314/1314 [=====] - 4s 3ms/step
329/329 [=====] - 1s 3ms/step
2803/2803 [=====] - 20s 7ms/step - loss: 31.5841 - val_loss: 6.0761
Epoch 2/30
1314/1314 [=====] - 3s 2ms/step
329/329 [=====] - 1s 3ms/step
2803/2803 [=====] - 17s 6ms/step - loss: 5.9998 - val_loss: 3.3886
Epoch 3/30
1314/1314 [=====] - 3s 2ms/step
329/329 [=====] - 1s 2ms/step
2803/2803 [=====] - 16s 6ms/step - loss: 3.2246 - val_loss: 3.9031
Epoch 4/30
1314/1314 [=====] - 4s 3ms/step
329/329 [=====] - 1s 3ms/step
2803/2803 [=====] - 17s 6ms/step - loss: 1.8258 - val_loss: 6.2236
Epoch 5/30
...
Test Score: 1.3846331696511402 RMSE
Test Score: 1.055717502333825 MAE
1314/1314 [=====] - 3s 2ms/step
329/329 [=====] - 1s 2ms/step
```

Fig.6.4.5

Code Snippet:

```
# Define a function to evaluate the model
def evaluate_model(model, test_X, test_Y):
    test_predict = model.predict(test_X)
    test_score = np.sqrt(np.mean((test_predict - test_Y) ** 2))
    test_mae = np.mean(np.abs(test_predict - test_Y))
    return test_score, test_mae

# Evaluate the CNN model
test_score, test_mae = evaluate_model(cnn_model, test_X, test_Y)
print(f'Test Score: {test_score} RMSE')
print(f'Test Score: {test_mae} MAE')
```

```
# Make predictions using the CNN model
train_predict = cnn_model.predict(train_X)
test_predict = cnn_model.predict(test_X)

# Plot the results
predict = np.concatenate((train_predict, test_predict))
plot_results(data_y, predict)
```

Fig.6.4.6

Epoch Loss over Validation (Tensorboard):

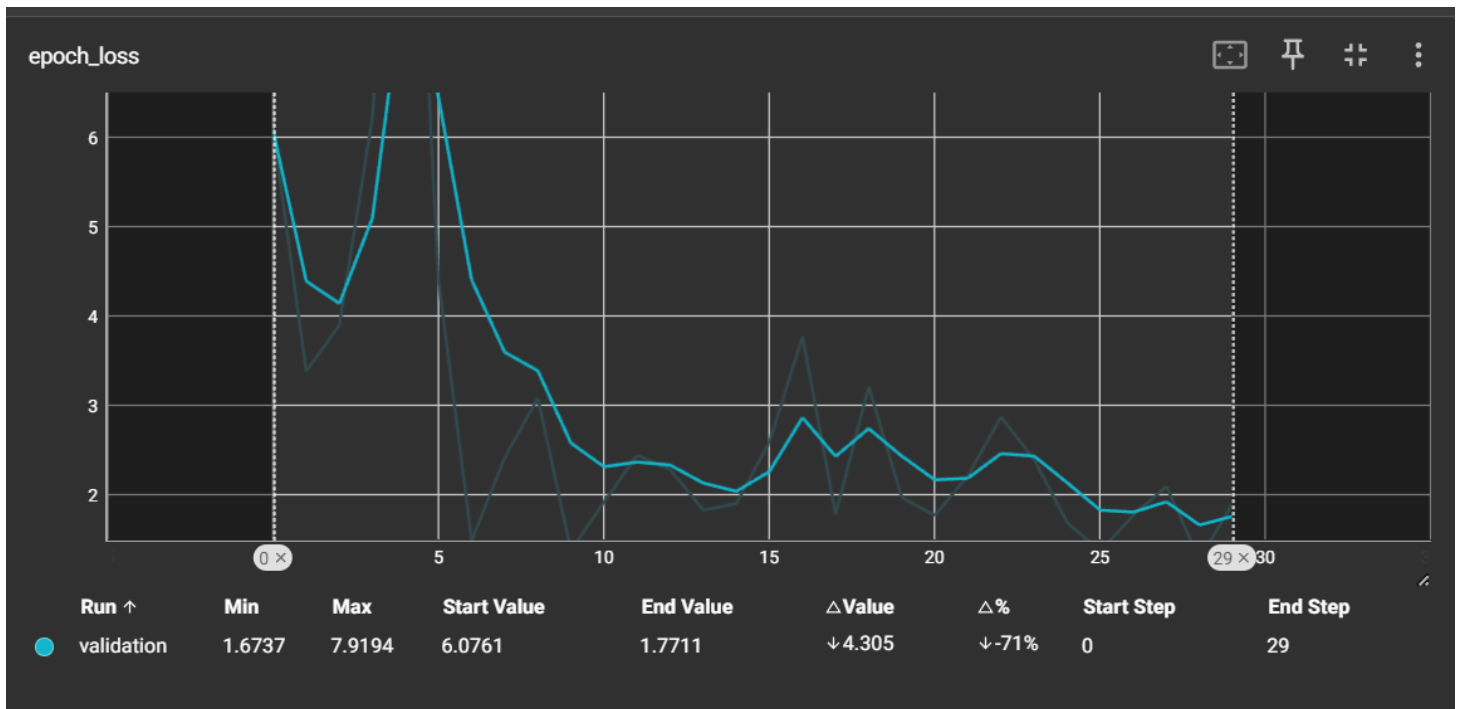


Fig.6.4.7

VII: SYSTEM IMPLEMENTATION

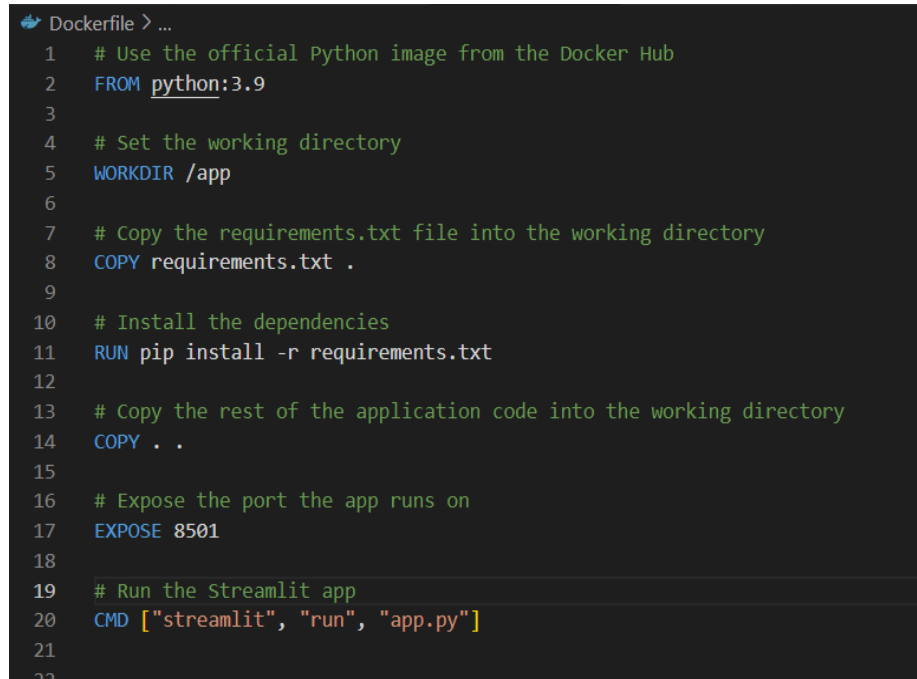
7.1 Docker Deployment

By integrating Docker and Streamlit, achieved a robust deployment pipeline for wind speed forecasting CNN model. Docker ensured seamless environment standardization and dependency management, encapsulating the entire model, preprocessing scripts, and necessary libraries into portable containers. This approach guaranteed consistency across development, testing, and deployment phases, regardless of the underlying infrastructure. Concurrently, Streamlit provided an intuitive web-based interface for showcasing the model's predictions, enabling interactive exploration of forecasted wind speeds.

H5 model:

Saved the CNN model in h5 format

Docker file:



```
Dockerfile > ...
1  # Use the official Python image from the Docker Hub
2  FROM python:3.9
3
4  # Set the working directory
5  WORKDIR /app
6
7  # Copy the requirements.txt file into the working directory
8  COPY requirements.txt .
9
10 # Install the dependencies
11 RUN pip install -r requirements.txt
12
13 # Copy the rest of the application code into the working directory
14 COPY . .
15
16 # Expose the port the app runs on
17 EXPOSE 8501
18
19 # Run the Streamlit app
20 CMD ["streamlit", "run", "app.py"]
21
22
```

Fig.7.1.1

Streamlit Web Framework:

Code Snippet:

```
# Function to create sequences
def predict_missing_values(df, look_back):
    features = [
        '80m Avg [m/s]', '50m Avg [m/s]', '20m Avg [m/s]', '10m Avg [m/s]',
        'Pressure 5m [mbar]', '98m WV [°]', '78m WV [°]', '48m WV [°]',
        'Temp 5m [°C]', 'Hum 5m'
    ]

    # Find rows where the target value is missing
    missing_indices = df.index[df['100m Avg[m/s]'].isna()].tolist()

    for index in missing_indices:
        # Extract the relevant rows to create a sequence
        seq_look_back = min(look_back, index + 1)
        sequence = df[features].iloc[index - seq_look_back + 1: index + 1].values.astype(float)

        # Reshape the sequence
        sequence = sequence.reshape(seq_look_back, len(features))

        if len(sequence) < look_back:
            # Pad the sequence if its length is less than look_back
            pad_width = ((look_back - len(sequence), 0), (0, 0))
            sequence = np.pad(sequence, pad_width, mode='constant', constant_values=0)

        sequence = sequence.reshape(1, look_back, len(features))
        prediction = model.predict(sequence)
        df.at[index, '100m Avg[m/s]'] = prediction[0][0]

    return df

# Streamlit app
st.title('Wind Speed Forecasting')

# Read the input CSV file
input_csv = 'input.csv'
df = pd.read_csv(input_csv)
```

Fig.7.1.2

Requirements file:

To load all the necessary modules into the container.

```
≡ requirements.txt
1  streamlit
2  pandas
3  numpy
4  matplotlib
5  keras
6  tensorflow
7  
```

Fig.7.1.3

Docker Container

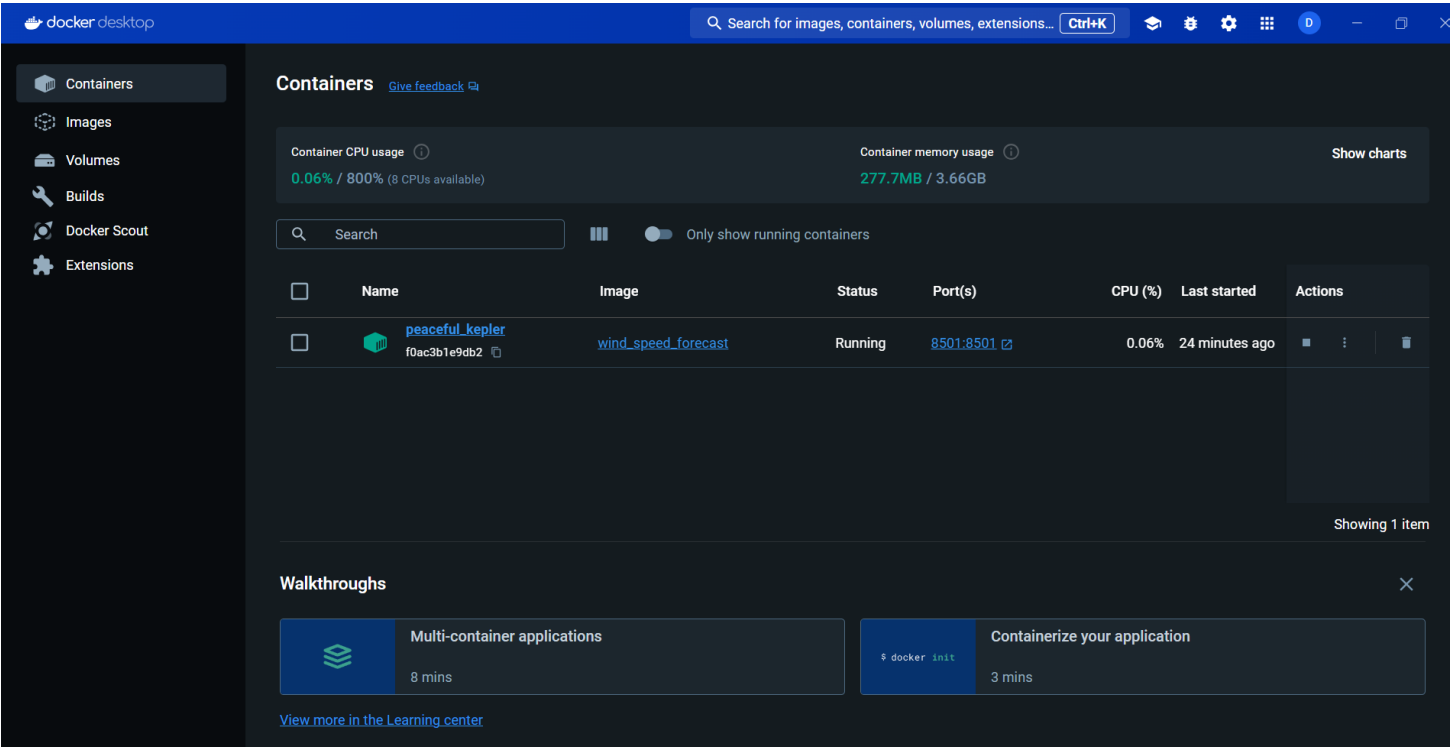


Fig.7.1.4

Docker Log for Errors

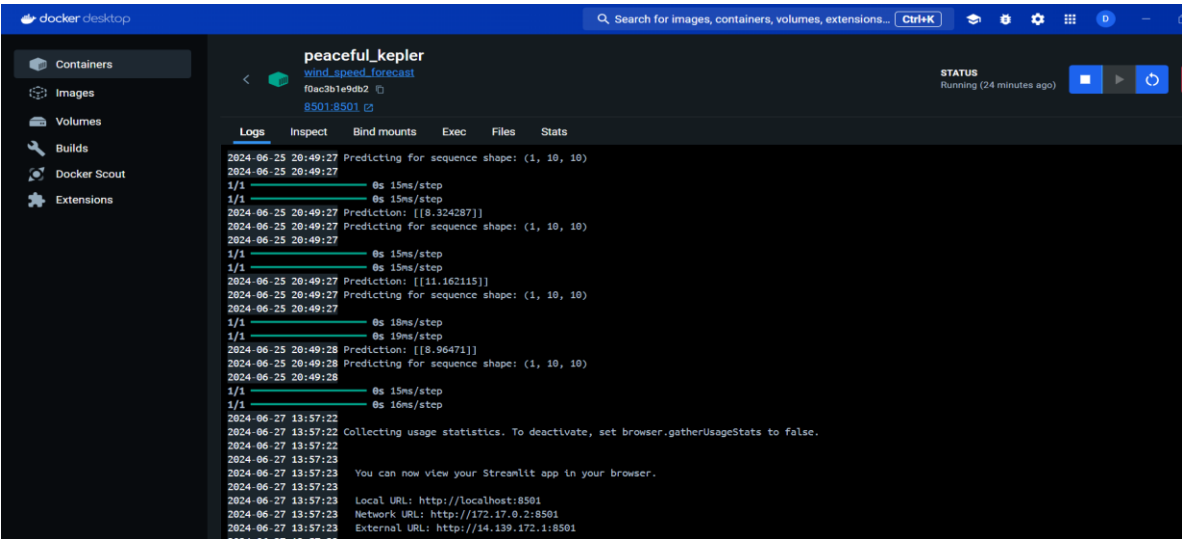


Fig.7.1.5

7.2 Automation

The process begins with a GitHub Actions workflow triggered on pushes or pull requests to the main branch. Then, it sets up Python and installs necessary dependencies like pandas, numpy, and TensorFlow.

The workflow executes a Python script which loads a pre-trained CNN model. This script processes CSV files located in the data/input directory, predicting missing values for the column using the model. Predictions are then saved to the data/output directory.

After processing, the workflow lists the contents of the output directory to verify the generated files. If changes are detected in the output files compared to the repository's state, it commits these changes with a message indicating automated data processing and pushes them back to the repository.

This automated pipeline ensures that whenever new CSV data is uploaded to the data/input directory on GitHub, the CNN model automatically fills in predictions for missing '100m Avg[m/s]' values, and updates are reflected in the repository.

GitHub Repository Overview

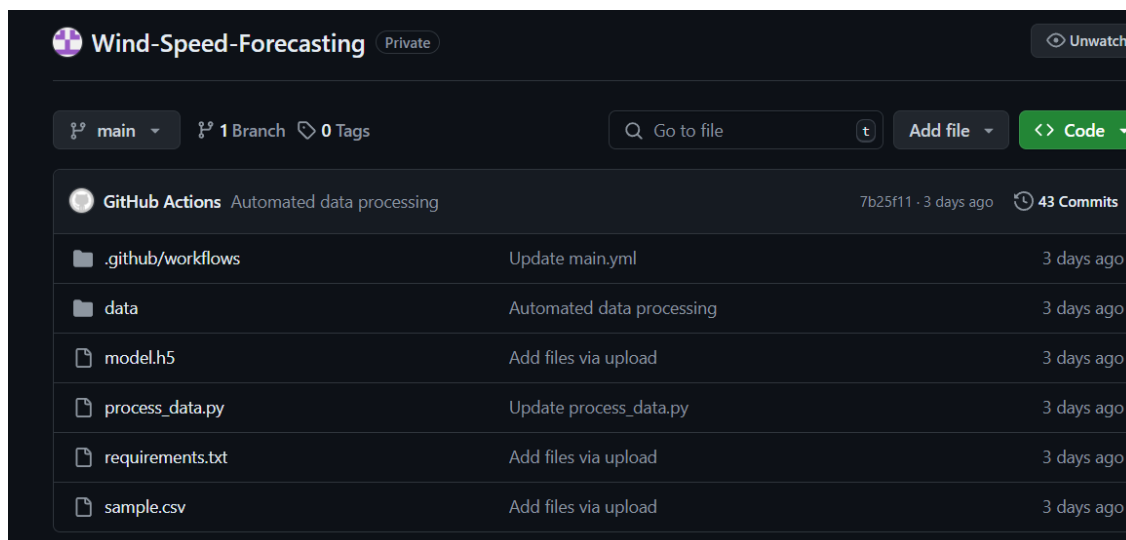


Fig.7.2.1

Workflow(yml) file:

```
name: Process CSV Files

on:
  push:
    branches:
      - main # Trigger on push to main branch
  pull_request:
    branches:
      - main # Trigger on pull requests to main branch

permissions:
  contents: write # Grant write permissions to the workflow

jobs:
  process-data:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout Repository
        uses: actions/checkout@v2

      - name: Set up Python
        uses: actions/setup-python@v2
        with:
          python-version: '3.x' # Replace with your Python version

      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip
          pip install pandas numpy tensorflow # Install necessary dependencies

      - name: Execute Python Script
        run: python process_data.py # Ensure this path is correct
```

Fig.7.2.2

Actions Output:

The screenshot displays the GitHub Actions interface for a workflow named 'process-data'. On the left, a sidebar shows the 'Summary' tab selected, with links for 'Jobs', 'Run details', 'Usage', and 'Workflow file'. The 'Jobs' section lists 'process-data' as the active job. The main panel shows the execution details of the 'process-data' job, which succeeded 3 days ago in 1m 0s. The job consists of three steps: 'Execute Python Script', 'Commit and Push Changes', and 'Post Set up Python'. The 'Execute Python Script' step is expanded, showing a log of commands and their outputs. The log includes line numbers (26-50) and timestamps (e.g., 11ms/step, 12ms/step). The output shows the installation of dependencies and the execution of the 'process_data.py' script, which saves the output to 'data/output/completed_sample.csv'. The final output shows the file size (1244 bytes) and the files in the output directory ('result_csv', 'completed_sample.csv').

Fig,7.2.3

VIII: SOFTWARE AND TOOLS USED

Python:



A versatile programming language used for various purposes such as data analysis, machine learning, and web development. Python's extensive libraries and frameworks make it a suitable choice for implementing the wind speed forecasting model.

Pandas:



A powerful data manipulation and analysis library for Python. It provides data structures like DataFrame for handling large datasets efficiently.

NumPy:



A fundamental package for scientific computing with Python. It provides support for arrays, matrices, and a collection of mathematical functions to operate on these data structures.

Matplotlib:



A plotting library for creating static, animated, and interactive visualizations in Python. It is used for visualizing the results of the wind speed forecasting model.

Keras (with TensorFlow backend):



A high-level neural networks API written in Python. Keras is used to build and train the CNN model for wind speed forecasting, while TensorFlow serves as the backend for performing the computations.

Streamlit:



An open-source app framework for creating and sharing beautiful, custom web apps for machine learning and data science. Streamlit is used to build the web application that displays the wind speed predictions.

Docker:



A platform that enables developers to automate the deployment of applications inside lightweight, portable containers. Docker is used to containerize the Streamlit application, ensuring consistent environments across different stages of deployment.

GitHub Actions:



A CI/CD (Continuous Integration/Continuous Deployment) platform that allows automation of workflows directly from GitHub. GitHub Actions is used to automate the data processing and model prediction workflow whenever a new dataset is uploaded to the repository.

Kalman Filter (from pykalman):

A Python library implementing Kalman Filter and Kalman Smoother. It is used for interpolating missing values in the wind speed data.

Scikit-learn:



A machine learning library for Python. It provides simple and efficient tools for data mining and data analysis. In this project, it is used for evaluating the model's performance using metrics like Mean Squared Error (MSE) and Mean Absolute Error (MAE).

Git:



A version control system used for tracking changes in source code during software development. Git is used to manage the project's source code and coordinate work among multiple developers.

TensorBoard:

Visualization toolkit for TensorFlow to visualize model graphs, plot quantitative metrics about the execution of your model, and show additional data like images that pass through it during execution.

IX: RESULT

Model Output

9.1 Time Series Plot

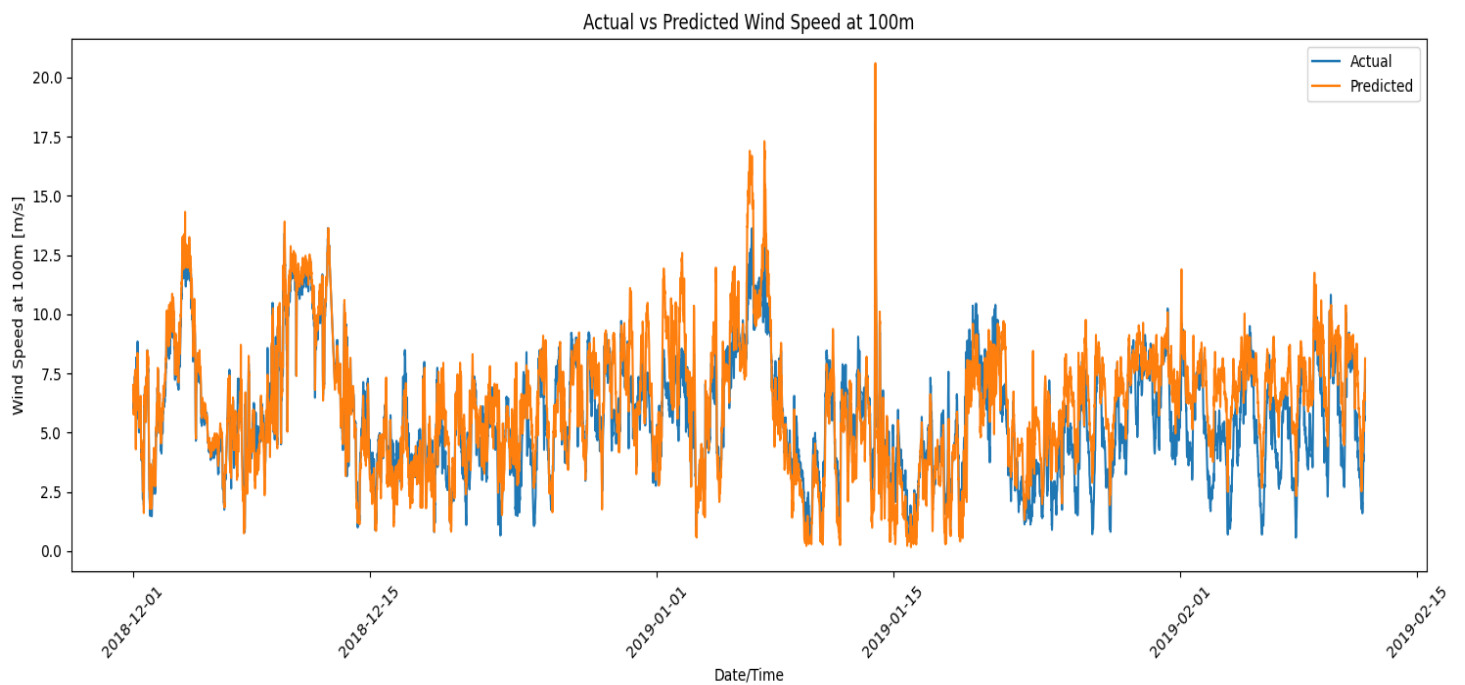


Fig.9.1.1

The plot reveals the deviations between the actual values and the model's predictions.

9.2 Box Plot by Hour of Day:

Analyze the variability of wind speeds by plotting box plots for actual and predicted wind speeds grouped by hour of the day.

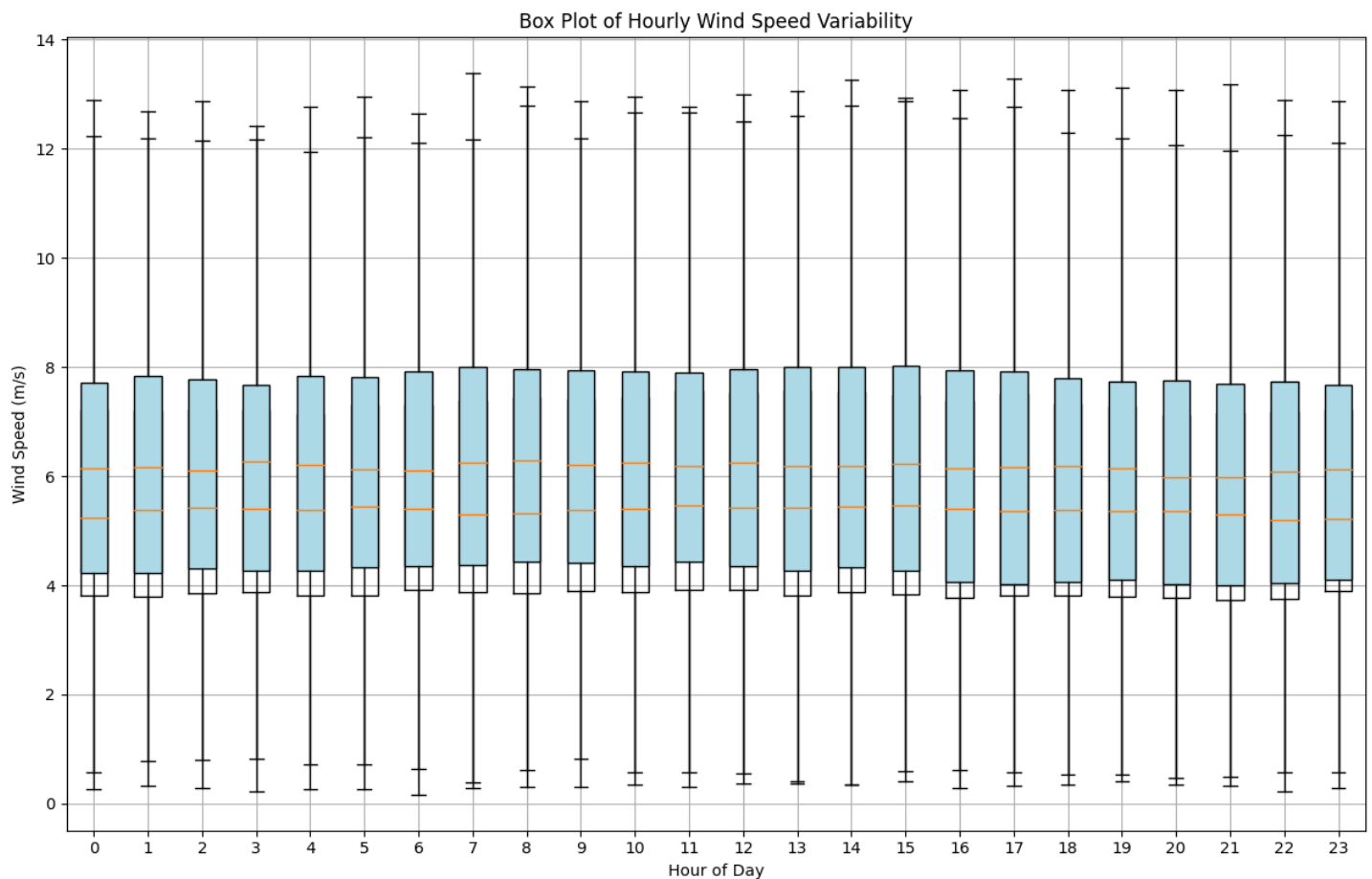


Fig.9.2.1

9.3 Cumulative Distribution Function (CDF) Plot:

Visualizing the cumulative distribution of actual and predicted wind speeds.

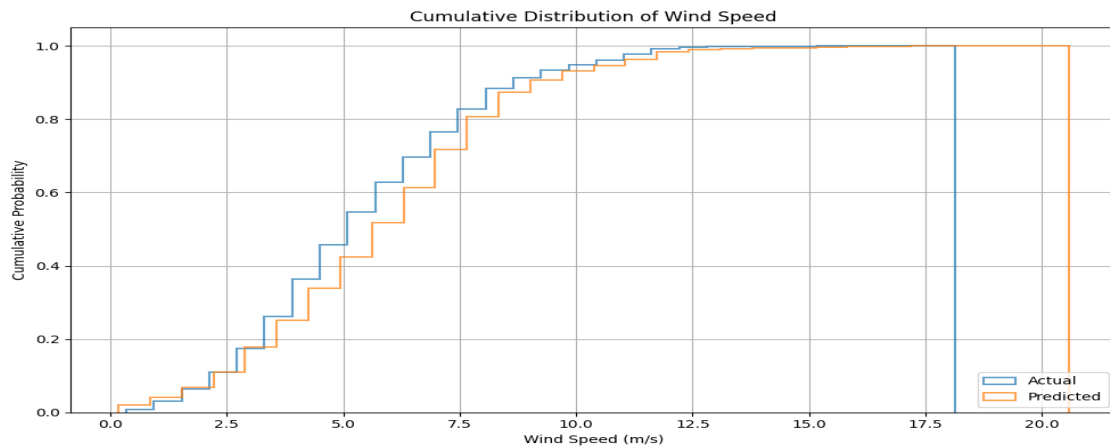


Fig.9.3.1

9.4 Monthly Average Variation

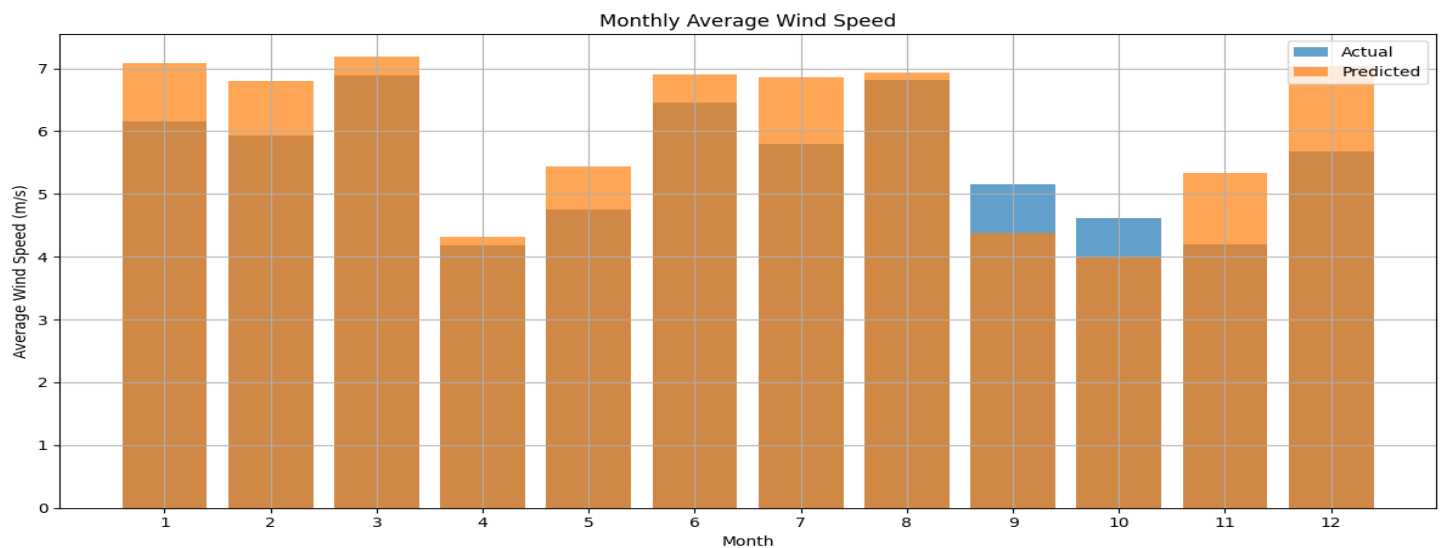


Fig.9.4.1

9.5 Seasonal Decomposition Plot

Decompose actual and predicted wind speeds into trend, seasonal, and residual components using seasonal decomposition analysis.

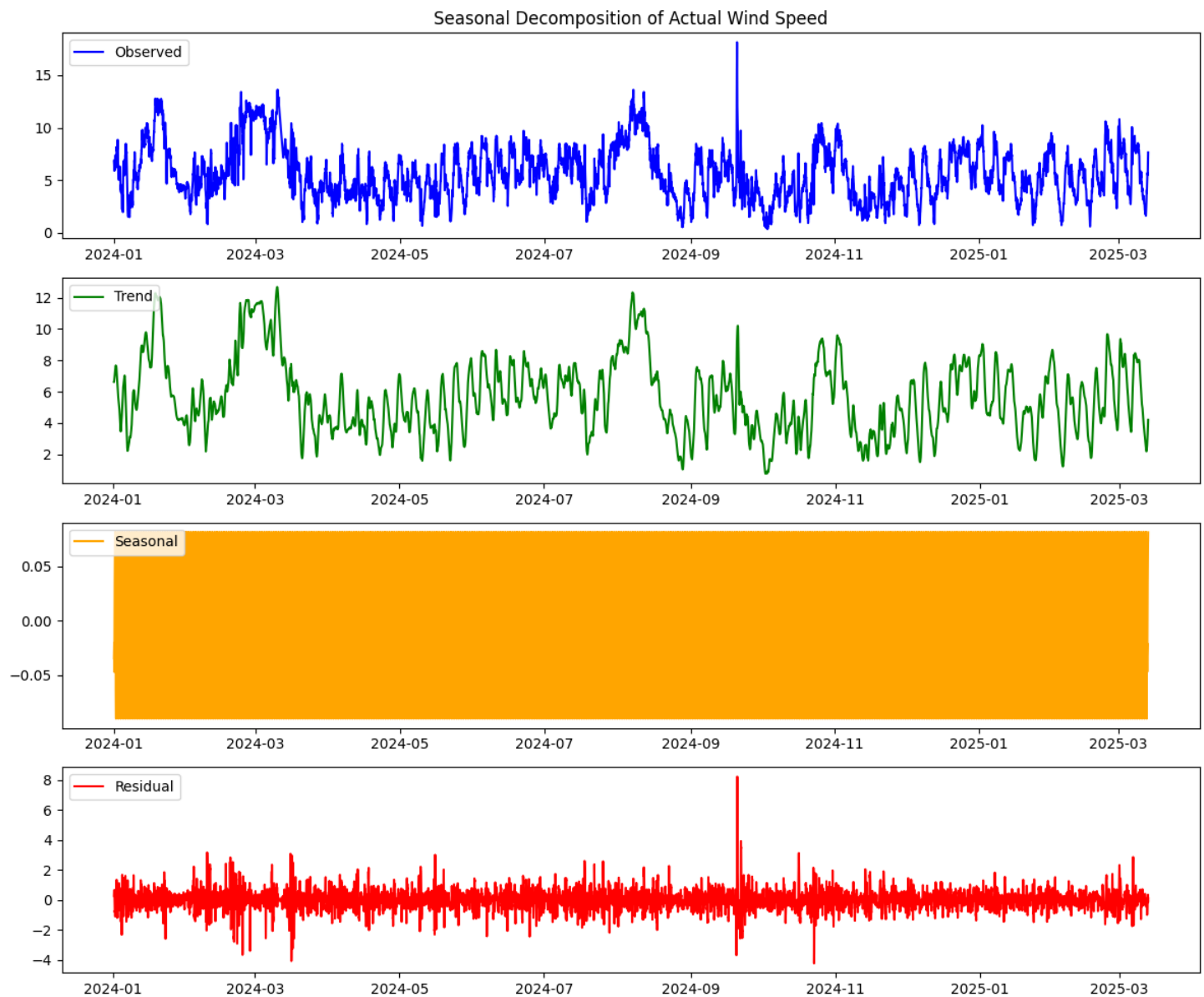


Fig.9.5.1

9.6 Streamlit Web Application

The Streamlit web application efficiently handles missing data in the '100m' wind speed column using a trained Convolutional Neural Network (CNN). It seamlessly predicts and fills these missing values based on uploaded datasets. Users can download an output CSV file containing the original input data augmented with predicted wind speeds, facilitating straightforward integration and further analysis.



Fig.9.6.1 UI

A screenshot of the "Input Data" section of the Streamlit application. It shows a table with 8 columns: an index column, "Date/Time", "100m Avg[m/s]", "80m Avg [m/s]", "50m Avg [m/s]", "20m Avg [m/s]", "10m Avg [m/s]", and "Pres". There are 10 rows of data. The "100m Avg[m/s]" column contains the value "None" for all rows, indicating missing data. The other columns contain numerical values. Above the table, there's a file upload section showing a file named "sample.csv" (1.0KB) has been uploaded. There's a "Browse files" button and a close button (X) for the file.

	Date/Time	100m Avg[m/s]	80m Avg [m/s]	50m Avg [m/s]	20m Avg [m/s]	10m Avg [m/s]	Pres
0	01-12-2018 00:00	None	5.17	7.12	4.82	4.14	
1	01-12-2018 00:10	None	5.3	7.34	4.89	4.16	
2	01-12-2018 00:20	None	5.32	7.39	4.81	4.16	
3	01-12-2018 00:30	None	5.32	7.41	4.78	4.16	
4	01-12-2018 00:40	None	5.35	7.37	4.81	4.16	
5	01-12-2018 00:50	None	5.34	7.4	4.78	4.16	
6	01-12-2018 01:00	None	5.29	7.18	4.69	4.16	
7	01-12-2018 01:10	None	5.45	6.99	4.58	4.16	
8	01-12-2018 01:20	None	5.59	6.45	3.89	4.16	
9	01-12-2018 01:30	None	5.86	5.85	3.36	4.16	

Fig.9.6.2 Input Data

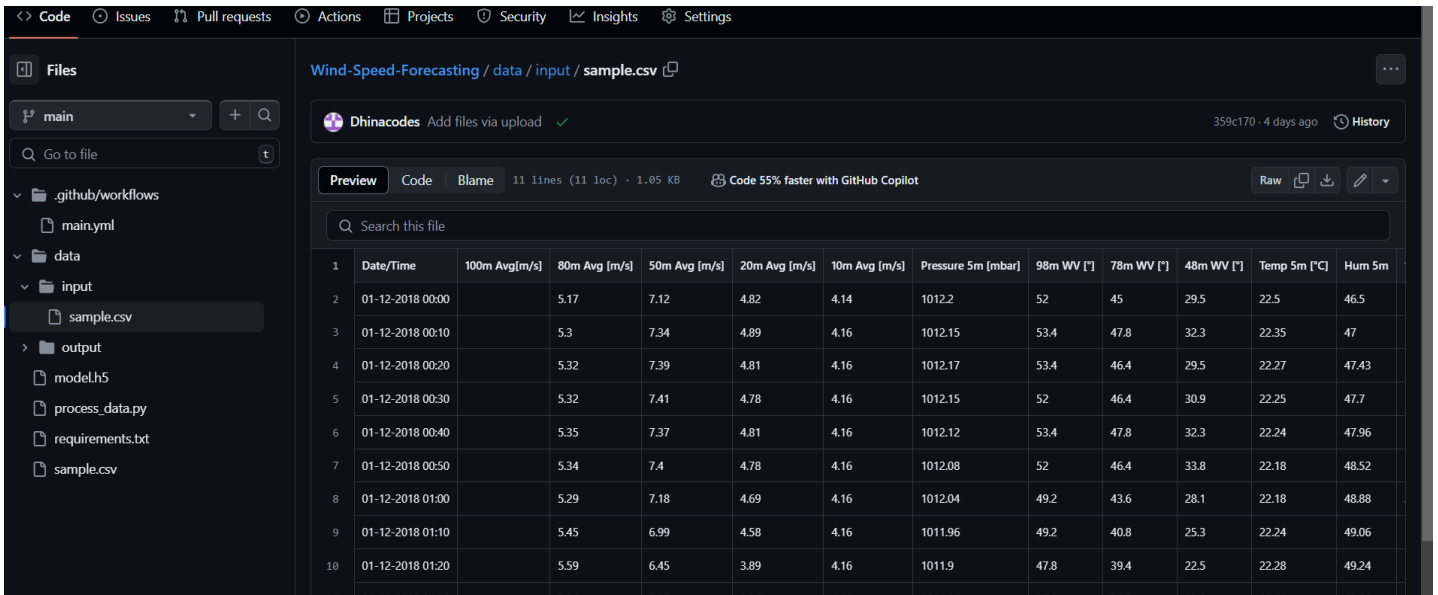
Data with Predictions:

	Date/Time	100m Avg[m/s]	80m Avg [m/s]	50m Avg [m/s]	20m Avg [m/s]	10m Avg [m/s]	Pres
0	01-12-2018 00:00	9.7177	5.17	7.12	4.82	4.14	
1	01-12-2018 00:10	6.0578	5.3	7.34	4.89	4.16	
2	01-12-2018 00:20	6.628	5.32	7.39	4.81	4.16	
3	01-12-2018 00:30	8.5387	5.32	7.41	4.78	4.16	
4	01-12-2018 00:40	11.2815	5.35	7.37	4.81	4.16	
5	01-12-2018 00:50	11.7982	5.34	7.4	4.78	4.16	
6	01-12-2018 01:00	8.3243	5.29	7.18	4.69	4.16	
7	01-12-2018 01:10	11.1621	5.45	6.99	4.58	4.16	
8	01-12-2018 01:20	8.9647	5.59	6.45	3.89	4.16	
9	01-12-2018 01:30	8.2864	5.86	5.85	3.36	4.16	

Download completed CSV

Fig.9.6.3 Output

9.7 Automation Result



Wind-Speed-Forecasting / data / input / sample.csv

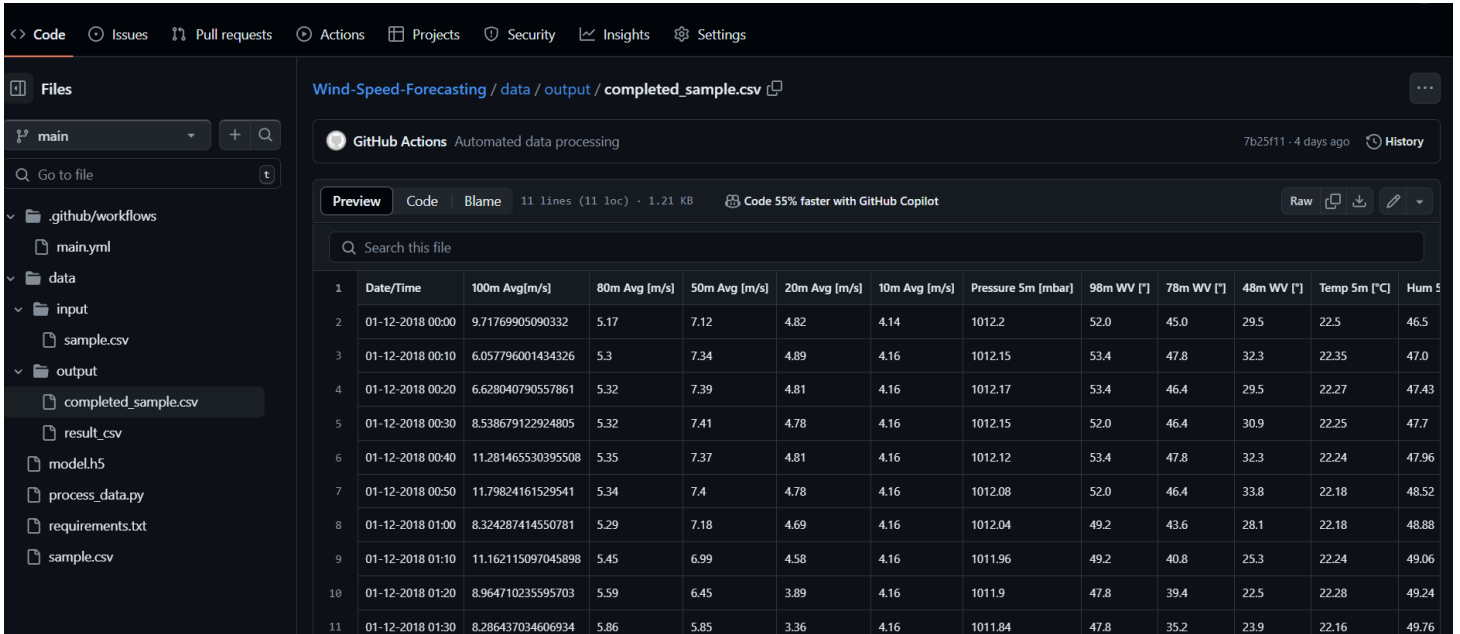
Dhinacodes Add files via upload 359c170 · 4 days ago History

Preview Code Blame 11 lines (11 loc) · 1.05 KB Code 55% faster with GitHub Copilot

Search this file

	Date/Time	100m Avg[m/s]	80m Avg [m/s]	50m Avg [m/s]	20m Avg [m/s]	10m Avg [m/s]	Pressure 5m [mbar]	98m WV [°]	78m WV [°]	48m WV [°]	Temp 5m [°C]	Hum 5m
1	01-12-2018 00:00		5.17	7.12	4.82	4.14	1012.2	52	45	29.5	22.5	46.5
2	01-12-2018 00:10		5.3	7.34	4.89	4.16	1012.15	53.4	47.8	32.3	22.35	47
3	01-12-2018 00:20		5.32	7.39	4.81	4.16	1012.17	53.4	46.4	29.5	22.27	47.43
4	01-12-2018 00:30		5.32	7.41	4.78	4.16	1012.15	52	46.4	30.9	22.25	47.7
5	01-12-2018 00:40		5.35	7.37	4.81	4.16	1012.12	53.4	47.8	32.3	22.24	47.96
6	01-12-2018 00:50		5.34	7.4	4.78	4.16	1012.08	52	46.4	33.8	22.18	48.52
7	01-12-2018 01:00		5.29	7.18	4.69	4.16	1012.04	49.2	43.6	28.1	22.18	48.88
8	01-12-2018 01:10		5.45	6.99	4.58	4.16	1011.96	49.2	40.8	25.3	22.24	49.06
9	01-12-2018 01:20		5.59	6.45	3.89	4.16	1011.9	47.8	39.4	22.5	22.28	49.24
10	01-12-2018 01:30		5.86	5.85	3.36	4.16	1011.84	47.8	35.2	23.9	22.16	49.76

Fig.9.7.1 Input file



Wind-Speed-Forecasting / data / output / completed_sample.csv

GitHub Actions Automated data processing 7b25f11 · 4 days ago History

Preview Code Blame 11 lines (11 loc) · 1.21 KB Code 55% faster with GitHub Copilot

Search this file

	Date/Time	100m Avg[m/s]	80m Avg [m/s]	50m Avg [m/s]	20m Avg [m/s]	10m Avg [m/s]	Pressure 5m [mbar]	98m WV [°]	78m WV [°]	48m WV [°]	Temp 5m [°C]	Hum 5m
1	01-12-2018 00:00	9.71769905090332	5.17	7.12	4.82	4.14	1012.2	52.0	45.0	29.5	22.5	46.5
2	01-12-2018 00:10	6.057796001434326	5.3	7.34	4.89	4.16	1012.15	53.4	47.8	32.3	22.35	47.0
3	01-12-2018 00:20	6.628040790557861	5.32	7.39	4.81	4.16	1012.17	53.4	46.4	29.5	22.27	47.43
4	01-12-2018 00:30	8.538679122924805	5.32	7.41	4.78	4.16	1012.15	52.0	46.4	30.9	22.25	47.7
5	01-12-2018 00:40	11.281465530395508	5.35	7.37	4.81	4.16	1012.12	53.4	47.8	32.3	22.24	47.96
6	01-12-2018 00:50	11.79824161529541	5.34	7.4	4.78	4.16	1012.08	52.0	46.4	33.8	22.18	48.52
7	01-12-2018 01:00	8.324287414550781	5.29	7.18	4.69	4.16	1012.04	49.2	43.6	28.1	22.18	48.88
8	01-12-2018 01:10	11.162115097045898	5.45	6.99	4.58	4.16	1011.96	49.2	40.8	25.3	22.24	49.06
9	01-12-2018 01:20	8.964710235595703	5.59	6.45	3.89	4.16	1011.9	47.8	39.4	22.5	22.28	49.24
10	01-12-2018 01:30	8.286437034606934	5.86	5.85	3.36	4.16	1011.84	47.8	35.2	23.9	22.16	49.76

Fig.9.7.2 Output file

X: CONCLUSION

The completion of this project has yielded compelling results, demonstrating the effectiveness of advanced methodologies and tools applied to wind speed forecasting. Key outcomes include:

1. **CNN Model Performance:** The Convolutional Neural Network (CNN) model, developed using TensorFlow and Keras, exhibited robust performance in predicting wind speeds. Trained on historical data and refined through iterative processes, the CNN model accurately forecasted wind speeds, validated through metrics such as Mean Squared Error (MSE) and Root Mean Squared Error (RMSE).
2. **Streamlit Web Application:** A Streamlit-based web application was created to facilitate user-friendly wind speed predictions. Integrated with Docker for efficient containerization and GitHub Workflows for automated deployment, the application ensured scalability and ease of use. Users could upload datasets for wind speed prediction, and the application processed these inputs, leveraging the trained CNN model to deliver accurate forecasts.
3. **GitHub Workflows Automation:** Automation via GitHub Workflows streamlined the deployment and operational processes of the Streamlit application. Upon uploading datasets to the designated repository, GitHub Workflows orchestrated the prediction and output generation workflows seamlessly. Screenshots from GitHub Workflows illustrate the automated data processing and prediction phases.
4. **Forecast Visualization:** Utilizing visualization tools like Matplotlib, forecasted wind speed trends were graphically represented. These visualizations provided insightful perspectives on predicted variations over time, aiding in decision-making and operational planning.

In conclusion, this project underscores the effectiveness of integrating advanced machine learning techniques with modern software engineering practices. By delivering accurate wind speed forecasts and empowering users with accessible prediction tools, this endeavour contributes significantly to enhancing decision-making in meteorology and related fields.

XI: REFERENCES

- [1] Brower, M. [2012]. Wind Resource Assessment: A Practical Guide to Developing a Wind Project.
- [2] "Deep learning applications in weather forecasting and climate change: A review" by Karpatne et al. (2017)
- [3] "A deep learning model for wind speed forecasting using convolutional neural networks" by Jia et al. (2019).
- [4] Manual for Real-Time Quality Control of Wind Data by IOOS (2014).
- [5] Wind Resource Assessment Handbook by NREL (1997).
- [6] "Wind speed forecasting: A review of the state-of-the-art, machine learning approaches, and research challenges" by Sharawi et al. (2021)
- [7] "Building and deploying a machine learning web app using Flask and Heroku" by Panchal (2021).
- [8] "Docker for Data Science: An evaluation of its strengths and weaknesses" by Brouwer et al. (2020).
- [9] GitHub Workflows Documentation