MULTIVARIATE AIR QUALITY FORECASTING USING DEEP LEARNING MODELS

A Case Study on Delhi Pollution

Project Report

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology
in
Electrical and Electronics Engineering

by

Akash P R (*Roll No.*: **B210775EE**)

Under the supervision of

Dr. Shihabudheen K. V.

Department of Electrical Engineering
NATIONAL INSTITUTE OF TECHNOLOGY CALICUT
April 2025

CERTIFICATE

This is to certify that the report entitled "**Multivariate Air Quality Forecasting Using Deep Learning Models: A Case Study on Delhi Pollution**" is a bonafide record of the **project** done by **Akash P R** under my supervision, in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Electrical and Electronics Engineering** from **National Institute of Technology Calicut**, and this work has not been submitted elsewhere for the award of a degree.

Dr. Shihabudheen K V
Asst. Professor
Dept. of Electrical Engineering

Dr. Sindhu T K
Professor & Head
Dept. of Electrical Engineering

Place : NIT Calicut
Date : 12 April 2025

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Dr. Shihabudheen K V for his invaluable support, guidance, and expertise throughout this project. I extend our appreciation to the faculty members of the Department of Electrical and Electronics Engineering for providing a conducive learning environment. I acknowledge our peers for their collaboration and constructive feedback during the preparation of this project. Thank you all for your contributions to the successful completion of this project report on Multivariate Air Quality Forecasting Using Deep Learning Models.

ABSTRACT

Air pollution remains one of the most critical environmental challenges in urban centers, particularly in rapidly growing cities like Delhi. This project presents a deep learning-based approach for multivariate air quality forecasting using time series data. By leveraging historical pollutant concentrations and meteorological parameters, we implement Long Short-Term Memory (LSTM) networks to predict PM2.5 levels — a key indicator of urban air pollution.

The dataset, sourced from the Delhi Air Quality dataset on Kaggle, undergoes comprehensive preprocessing including missing value imputation, normalization, and the creation of time-lagged features and moving averages. Exploratory data analysis (EDA) reveals significant temporal patterns and correlations among pollutants. The LSTM model is trained on a supervised learning structure with early stopping and model checkpointing to avoid overfitting. Evaluation using MAE, RMSE, and R^2 score indicates strong predictive performance. This study demonstrates the viability of deep learning techniques for environmental forecasting and supports informed policy-making for air pollution control.

CHAPTER 1
INTRODUCTION

Air quality forecasting is becoming increasingly vital for public health and environmental management, especially in densely populated urban areas. Among major Indian cities, **Delhi consistently ranks among the most polluted**, with elevated concentrations of fine particulate matter (PM2.5 and PM10) and other harmful gases like NO_2, CO, and O_3. These pollutants not only pose serious health risks but also lead to environmental degradation and economic losses.

Recent advances in deep learning have made it possible to model complex temporal dependencies in environmental data. Traditional statistical models often fall short in capturing the nonlinear interactions between meteorological variables and pollutant levels. In contrast, **Recurrent Neural Networks (RNNs)**—especially **Long Short-Term Memory (LSTM)** networks—are well-suited for modeling sequential time series data with long-range dependencies.

This project, titled **"Multivariate Air Quality Forecasting Using Deep Learning Models: A Case Study on Delhi Pollution"**, aims to:

- **Forecast future PM2.5 levels** using a multivariate LSTM model trained on historical data of multiple air pollutants and weather indicators.
- **Evaluate model performance** using appropriate regression metrics (MAE, RMSE, R^2).

By accurately forecasting pollutant levels, this project aims to support proactive pollution control measures and provide timely alerts to the public, thereby contributing to smarter urban environmental management.

CHAPTER 2

DATA COLLECTION AND PREPROCESSING

2.1 DATASET SOURCE

The data used in this study was obtained from the publicly available *Air Quality Data in India (2015 - 2020)* hosted on Kaggle. This dataset comprises time-stamped records of various atmospheric pollutant concentrations and meteorological parameters recorded at monitoring stations across multiple cities through India. The features include major air pollutants such as PM2.5, PM10, NO, NO_2, NOx, NH_3, CO, SO_2, O_3, Benzene, and Toluene, as well as weather-related variables like temperature, humidity, wind speed, and wind direction. The dataset provides a comprehensive view of Delhi's air quality over several years, with measurements captured at an hourly frequency, making it highly suitable for time series analysis and forecasting

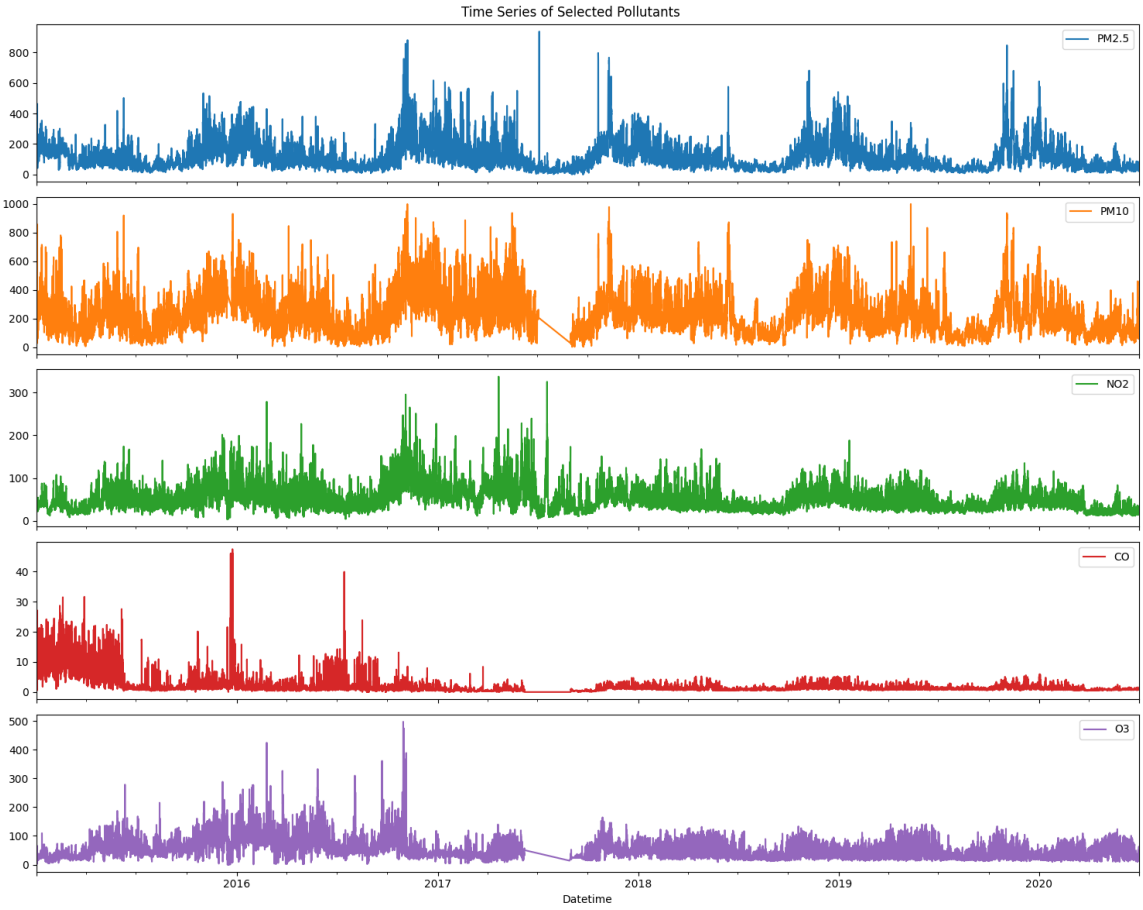
Link to dataset used: https://www.kaggle.com/datasets/rohanrao/air-quality-data-in-india/data?select=city_hour.csv

2.2 DATA FILTERING AND MISSING DATA

To ensure consistency and usability for time series modeling, the raw dataset underwent several preprocessing steps. The datetime column was parsed and set as the index to facilitate chronological operations. Since some records were missing or recorded at irregular intervals, the data was resampled to a consistent hourly frequency. Missing values were then handled using a combination of forward-fill methods and linear interpolation. In cases where pollutant readings were missing for extended periods, the corresponding entries were dropped to prevent skewing the model's learning process. This careful handling of missing data helped preserve temporal continuity and ensured the dataset's readiness for modeling.

2.3 FEATURE SELECTION

Given the project's primary objective of forecasting PM2.5 concentrations, a subset of relevant features was selected based on their correlation with the target variable and their overall contribution to pollution dynamics. The selected features—PM2.5, PM10, NO_2, CO, and O_3—were chosen for their strong predictive power and physical relevance in air quality modeling. These pollutants are known to interact chemically and environmentally, making them suitable inputs for a multivariate forecasting model. To further enhance the dataset, time series feature engineering was applied, including the addition of lag features (previous time step values) and moving averages to capture short-term temporal dependencies. All features were normalized using the MinMaxScaler to ensure they fell within the 0–1 range, which is essential for efficient neural network training and convergence.



CHAPTER 3

MODEL IMPLEMENTATION

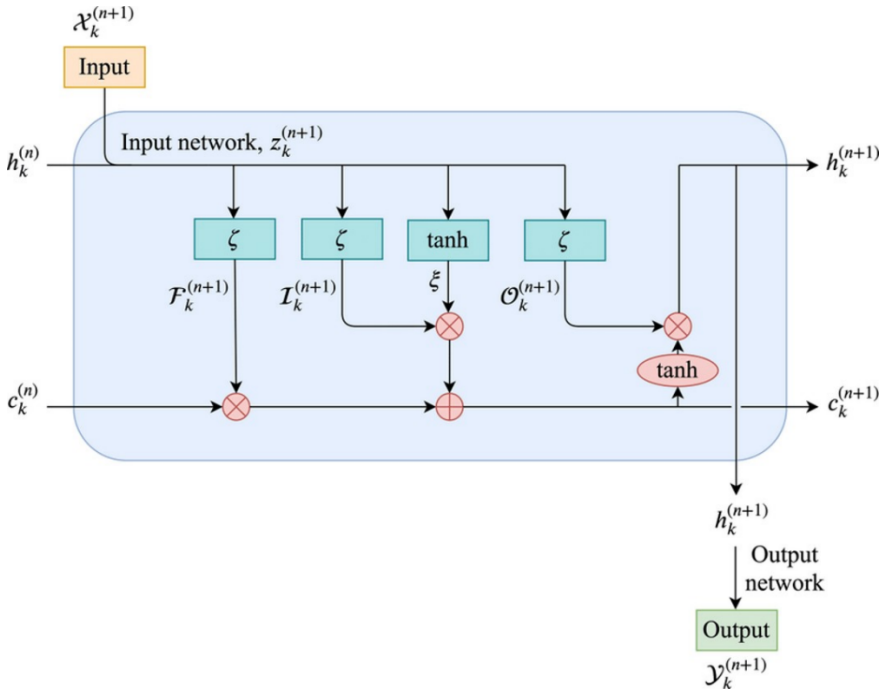
3.1 MODEL COMPARISONS

3.1.1 Recurrent Neural Networks (RNNs)

RNNs process sequential data by maintaining a hidden state that updates as the input sequence progresses. However, the challenges of vanishing and exploding gradients limited their application in handling longer sequences, leading to reduced forecasting accuracy.

3.1.2 Long Short-Term Memory Networks (LSTMs)

LSTMs, an advanced variant of RNNs, address these challenges by incorporating gating mechanisms, such as forget, input, and output gates. This allows LSTMs to selectively retain and discard information, resulting in better modelling of long-term dependencies and higher accuracy in temporal data forecasting.



3.2 IMPLEMENTATION OF LSTM

The Long Short-Term Memory (LSTM) model is designed to capture temporal dependencies in sequential data, making it well-suited for forecasting tasks like solar power generation. Unlike traditional feedforward networks, LSTMs are capable of retaining important information over long sequences, mitigating issues like vanishing gradients, and modelling time-dependent patterns in the data. The LSTM model consists of several layers designed to process and predict time-series data. The following table summarizes the architecture of the LSTM model.

Layer	**Output Shape**	**Parameters**
Input Layer	(None, *time_steps=24*, *n_features*)	0
LSTM	(None, 24,64)	23,040
Dropout	(None, 24,64)	0
LSTM	(None, 32)	12,416
Dropout	(None, 32)	0
Dense	(None,1)	33

Table 5.1: LSTM Model Architecture

This LSTM-based model processes the temporal data sequentially and makes predictions based on learned dependencies. It is particularly effective for time series forecasting tasks, where the output is dependent on past time steps. The architecture combines the advantages of the LSTM's ability to model sequential dependencies with dropout regularization to enhance generalization performance.

3.3 MODEL OPTIMIZATION

To enhance performance, several optimization techniques were applied during training. This included implementing early stopping to prevent overfitting and adopting dropout layers to improve generalization. The learning rate was monitored and adjusted dynamically during training to ensure smooth convergence.

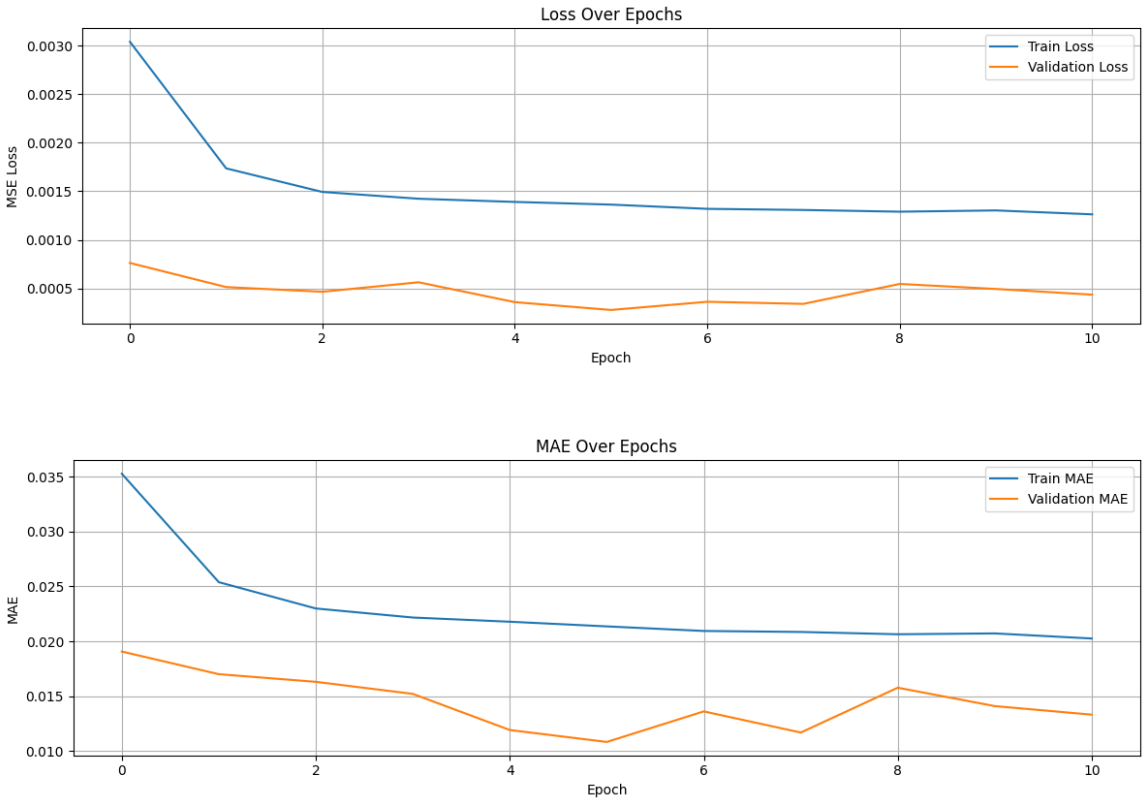


Figure 3.3: Loss variation with Epochs

CHAPTER 4

RESULTS AND EVALUATION

The evaluation of the LSTM model with four dimensional input focused on its ability to forecast Air Quality with high accuracy. Various metrics were employed to gauge its performance, including Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and the R^2 score. The model's training, validation, and test results are comprehensively analyzed, providing insights into its predictive reliability, generalization capabilities, and potential areas for refinement.

4.1 TRAINING AND VALIDATION PERFORMANCE

The model was set to be trained over 100 epochs with a learning rate of 0.001. A key observation during the training process was the steady decline in both training and validation losses. This demonstrates effective learning without significant overfitting.

The training RMSE, computed as the square root of the mean squared error over the training set, indicated a consistent reduction in prediction errors as the model parameters optimized over epochs. Validation loss exhibited a similar downward trend, highlighting the model's ability to generalize well to unseen validation data.

4.2 TEST PERFORMANCE EVALUATION

The model's performance on unseen data was assessed using various statistical metrics, each providing a unique perspective on its predictive accuracy. Below is a detailed analysis of the evaluation metrics.

4.2.1 Root Mean Squared Error (RMSE)

RMSE measures the square root of the average squared differences between predicted and actual values. It provides insight into the magnitude of prediction errors, with lower values indicating better accuracy.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}$$

Where:

- n : Number of observations
- y_i : Actual value
- $\hat{y}_i$: Predicted value

4.2.2 Mean Absolute Error (MAE)

MAE represents the average absolute differences between predicted and actual values. It quantifies the model's accuracy in terms of actual power units, providing an intuitive interpretation of error magnitude.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right|$$

4.2.3 R^2 Score

The R^2 score, also known as the coefficient of determination, measures how well the predicted values explain the variance in the actual values. It ranges from 0 to 1, where higher values indicate better performance.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2}$$

Where, $\bar{y}$: Mean of the actual values

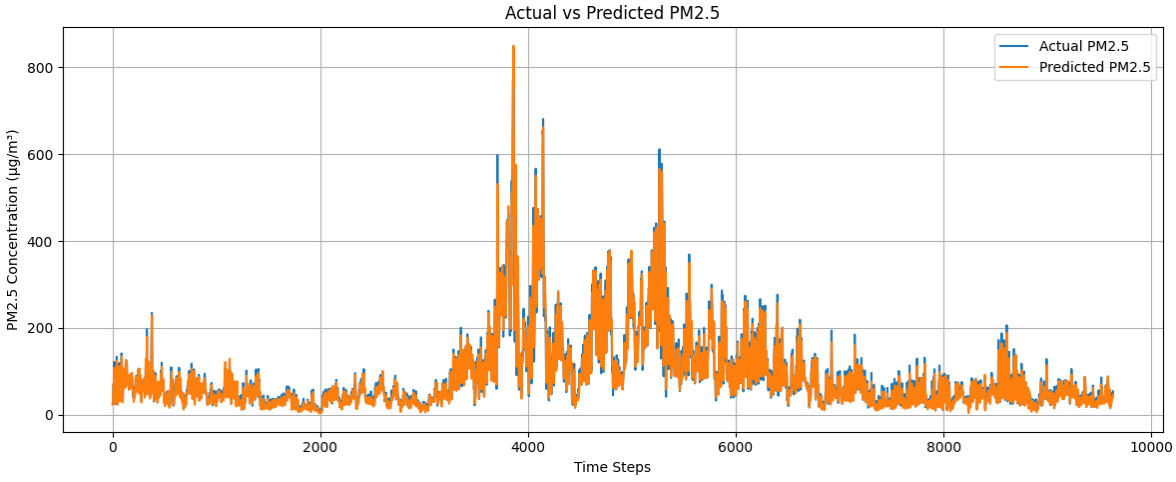
4.2.4 Test Dataset Analysis

RMSE (µg/m³)	MAE (µg/m³)	R² Score
14.14	8.92	0.9746

Table 4.1: Result analysis on test dataset

4.3 VISUALIZATION

The LSTM model's performance is showcased through visualization. This model demonstrated the most accurate predictions, achieving optimal results in terms of mean squared error (MSE) and R-squared. The visualization of this model allows for a clearer understanding of its predictive capabilities.



CHAPTER 5

CODE

```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import zscore
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score

df = pd.read_csv('city_hour.csv')
df

#Filtering only Delhi rows from the Kaggle Dataset
df = df[df['City'] == 'Delhi'].copy()
print(df.head())
print(df.info())

df['Datetime'] = pd.to_datetime(df['Datetime'])
df = df.set_index('Datetime')  # To Make datetime the index for time-
series modeling
df = df.sort_index()  # To Ensure it's in time order

print(df.isnull().sum()) #for checking the missing and null data

missing_percent = df.isnull().mean() * 100 # Missing percentage
print(missing_percent.sort_values(ascending=False))

df = df.interpolate(method='time')  # or 'linear' Interpolation of
missing data
df = df.fillna(method='ffill').fillna(method='bfill')  # backup

selected_features = ['PM2.5', 'PM10', 'NO2', 'CO', 'O3']
df = df[selected_features]

print(df.isnull().sum()) #for checking the missing and null data

df.plot(subplots=True, figsize=(15, 12), title='Time Series of Selected
Pollutants')
plt.tight_layout()
plt.show()

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(df)

# Convert back to DataFrame for easier handling
```

```
scaled_df = pd.DataFrame(scaled_data, index=df.index,
columns=df.columns)

# Add lag features
for col in df.columns:
    scaled_df[f'{col}_lag1'] = scaled_df[col].shift(1)
    scaled_df[f'{col}_lag2'] = scaled_df[col].shift(2)

# Add moving average features
for col in df.columns:
    scaled_df[f'{col}_ma3'] = scaled_df[col].rolling(window=3).mean()
    scaled_df[f'{col}_ma7'] = scaled_df[col].rolling(window=7).mean()

# Drop rows with NaNs after shifting/rolling
scaled_df.dropna(inplace=True)

import numpy as np
def create_sliding_window(data, target_col='PM2.5', window_size=24,
forecast_horizon=1):
    X, y = [], []
    for i in range(len(data) - window_size - forecast_horizon + 1):
        X.append(data.iloc[i:i+window_size].values)
        y.append(data.iloc[i+window_size+forecast_horizon-1]
[target_col])
    return np.array(X), np.array(y)

X, y = create_sliding_window(scaled_df, target_col='PM2.5',
window_size=24, forecast_horizon=1)

split = int(0.8 * len(X))
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]

import matplotlib.pyplot as plt

def plot_moving_averages(df, feature, ma_windows=[3, 7]):
    plt.figure(figsize=(15, 5))
    plt.plot(df.index, df[feature], label=f'{feature} (Original)',
alpha=0.6)

    for window in ma_windows:
        if f'{feature}_ma{window}' in df.columns:
            plt.plot(df.index, df[f'{feature}_ma{window}'],
label=f'MA{window}')

    plt.title(f'{feature} with Moving Averages')
    plt.xlabel('Time')
    plt.ylabel('Normalized Value')
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

```
# PM2.5
plot_moving_averages(scaled_df, 'PM2.5')

# CO
plot_moving_averages(scaled_df, 'CO')

# NO2
plot_moving_averages(scaled_df, 'NO2')

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping

model = Sequential()

# First LSTM Layer
model.add(LSTM(64, return_sequences=True,
input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.2))

# Second LSTM Layer
model.add(LSTM(32))
model.add(Dropout(0.2))

# Output Layer
model.add(Dense(1))  # predicting PM2.5

# Compile Model
model.compile(optimizer='adam', loss='mean_squared_error',
metrics=['mae'])

model.summary()

early_stop = EarlyStopping(
    monitor='val_loss',      # we can also use 'val_mae'
    patience=5,              # Wait 5 epochs after no improvement
    restore_best_weights=True
)

history = model.fit(
    X_train, y_train,
    epochs=100,
    batch_size=32,
    validation_split=0.2,
    callbacks=[early_stop],    # <-- Added callback
    verbose=1
)

loss, mae = model.evaluate(X_test, y_test)
print(f'Test Loss: {loss:.4f}, Test MAE: {mae:.4f}')
```

```
y_pred = model.predict(X_test)

# Inverse transform to original scale for y_test
y_test_inv = scaler.inverse_transform(np.repeat(y_test.reshape(-1, 1),
len(selected_features), axis=1))[:, 0] # Extract PM2.5

# Inverse transform to original scale for y_pred
y_pred_inv = scaler.inverse_transform(np.repeat(y_pred,
len(selected_features), axis=1))[:, 0] # Extract PM2.5

plt.figure(figsize=(12, 5))
plt.plot(y_test_inv, label='Actual PM2.5')
plt.plot(y_pred_inv, label='Predicted PM2.5')
plt.title('Actual vs Predicted PM2.5')
plt.xlabel('Time Steps')
plt.ylabel('PM2.5 Concentration (μg/m³)')
plt.legend()
plt.tight_layout()
plt.grid(True)
plt.show()

mae = mean_absolute_error(y_test_inv, y_pred_inv)
rmse = np.sqrt(mean_squared_error(y_test_inv, y_pred_inv))
r2 = r2_score(y_test_inv, y_pred_inv)

print(f"MAE  : {mae:.2f}")
print(f"RMSE : {rmse:.2f}")
print(f"R²   : {r2:.4f}")

# Loss plot
plt.figure(figsize=(12, 4))
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss Over Epochs')
plt.xlabel('Epoch')
plt.ylabel('MSE Loss')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
# MAE plot
plt.figure(figsize=(12, 4))
plt.plot(history.history['mae'], label='Train MAE')
plt.plot(history.history['val_mae'], label='Validation MAE')
plt.title('MAE Over Epochs')
plt.xlabel('Epoch')
plt.ylabel('MAE')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

CHAPTER 6

CONCLUSION

This project presented a deep learning-based approach for forecasting air pollution levels in Delhi, focusing specifically on PM2.5 concentrations due to their critical impact on public health. By leveraging a multivariate Long Short-Term Memory (LSTM) model trained on historical air quality and meteorological data, we were able to capture complex temporal dependencies and pollutant interactions. The preprocessing phase involved rigorous data cleaning, feature engineering, and normalization, which laid a strong foundation for effective model training.

The model demonstrated excellent predictive performance, achieving a Mean Absolute Error (MAE) of 8.92, a Root Mean Squared Error (RMSE) of 14.14, and a high R^2 score of 0.9746 on the test set. These results indicate that the model was able to explain over 97% of the variance in PM2.5 concentrations, validating its effectiveness in capturing the underlying patterns in the data. Visualizations of actual versus predicted values further confirmed the model's accuracy and robustness.

Overall, this study illustrates the potential of LSTM networks in air quality forecasting and highlights the importance of incorporating multiple pollutants and environmental variables into predictive models. With further optimization and integration into a real-time dashboard, such models can serve as powerful tools for early warning systems, public health advisories, and policy planning aimed at reducing air pollution in urban environments.