

```
In [1]: import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import ipywidgets as widgets
from IPython.display import display
from sklearn.preprocessing import MinMaxScaler
import tensorflow as tf
```

```

In [2]: # Function to load and preprocess data
def load_and_preprocess_data(folder_path):
    csv_files = [file for file in os.listdir(folder_path) if file.endswith('.csv')]
    combined_data = pd.concat([pd.read_csv(os.path.join(folder_path, file)) for file in csv_files])
    combined_data['ts'] = pd.to_datetime(combined_data['ts'])
    combined_data.set_index('ts', inplace=True)
    return combined_data

# Function to train LSTM model
def train_lstm_model(combined_data, seq_length=30):
    data = combined_data[['humidity1', 'temperature1', 'humidity2', 'temperature2']]
    scaler = MinMaxScaler()
    scaled_data = scaler.fit_transform(data)

    X, y = [], []
    for i in range(len(scaled_data) - seq_length):
        X.append(scaled_data[i:i + seq_length])
        y.append(scaled_data[i + seq_length])
    X, y = np.array(X), np.array(y)

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

    model = tf.keras.models.Sequential([
        tf.keras.layers.LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.LSTM(units=50, return_sequences=False),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.Dense(units=16),
        tf.keras.layers.Dense(units=len(data.columns))
    ])
    model.compile(optimizer='adam', loss='mse')
    history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_test, y_test))
    mse = model.evaluate(X_test, y_test)
    predictions = model.predict(X_test)
    predictions = scaler.inverse_transform(predictions)
    y_test = scaler.inverse_transform(y_test)
    return predictions, y_test, history

# Function to update plots based on user input
def update_plots(folder_path):
    combined_data = load_and_preprocess_data(folder_path)
    predictions, y_test, history = train_lstm_model(combined_data)
    plt.figure(figsize=(12, 6))
    plt.plot(y_test, label='Actual')
    plt.plot(predictions, label='Predicted')
    plt.xlabel('Time')
    plt.ylabel('Temperature')
    plt.title('Actual vs Predicted Temperature')
    plt.legend()
    plt.show()

# Widget for folder path input
folder_path_widget = widgets.Text(value=r'D:\Training Dataset\Training Dataset')

# Button to trigger update

```



```
C:\Users\lenovo\anaconda3\Lib\site-packages\keras\src\layers\rnn\rnn.py:204:  
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. Wh  
en using Sequential models, prefer using an `Input(shape)` object as the firs  
t layer in the model instead.
```

```
super().__init__(**kwargs)
```

```
134/134 ————— 1s 4ms/step - loss: 0.0011
```

```
134/134 ————— 1s 5ms/step
```

