Smart Rain Prediction System Using Humidity Data and ML on an ESP32 based weather station

EE2021E Assignment: Python Application Development

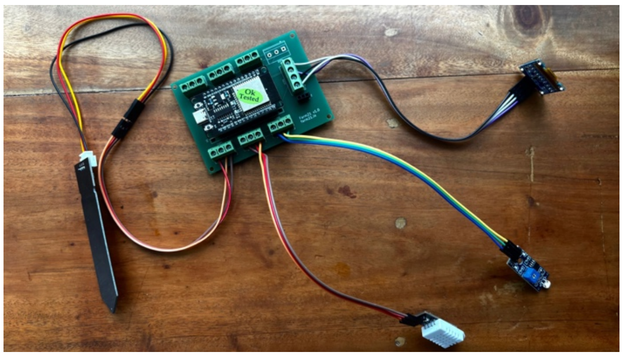
Submitted by: Johan Sanju Sebastian (B240051EE)

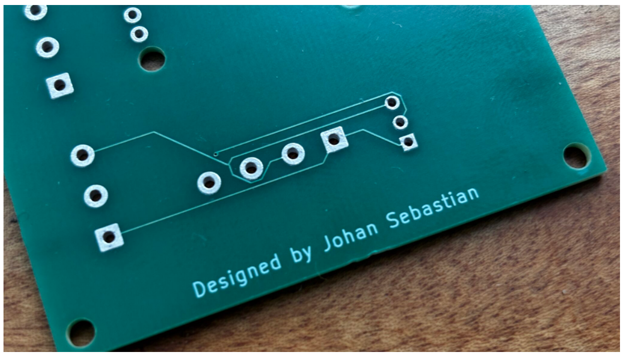
1. Application Selection:

- Sensor-Based Applications: Data acquisition, real-time monitoring, and environmental sensing.
- Machine Learning: Projects involving forecasting, classification, or clustering, using datasets with up to 1,000 samples and a maximum of 20 features.

2. Project Overview

- This project aims to use humidity and temperature data, then teach a model to predict what could come next, then use that to guess rain chances, and finally draw plots showing what happened.
- The program uses a neural network called an LSTM or "Long Short-Term Memory." designed for working with sequence data, like weather readings over time, text, or sensor signals.
- What makes LSTM different from most neural networks is its ability to "remember" information from earlier time steps for a long duration. This is very useful for recognizing patterns that change over time, such as the way temperature and humidity might be leading indicators for rain.
- The project makes use of a custom designed (*designed by myself) PCB based weather station that contains an ESP32 microcontroller, DHT22 (humidity and temperature sensor), a capacitive soil moisture sensor and LDR.





- For the purpose of this project, we only use the DHT22 sensor as we only require humidity and temperature data, while the soil moisture sensor could've been incorporated, it would increase the number of data points for the model making it slower.

3. Code Logic

1. Data Logging

- To start off the project we use a data logger python file to automatically log the humidity and temperature data from the serial monitor of the ESP32 and then store it in a csv file.

2. Data Sorting and Model Building

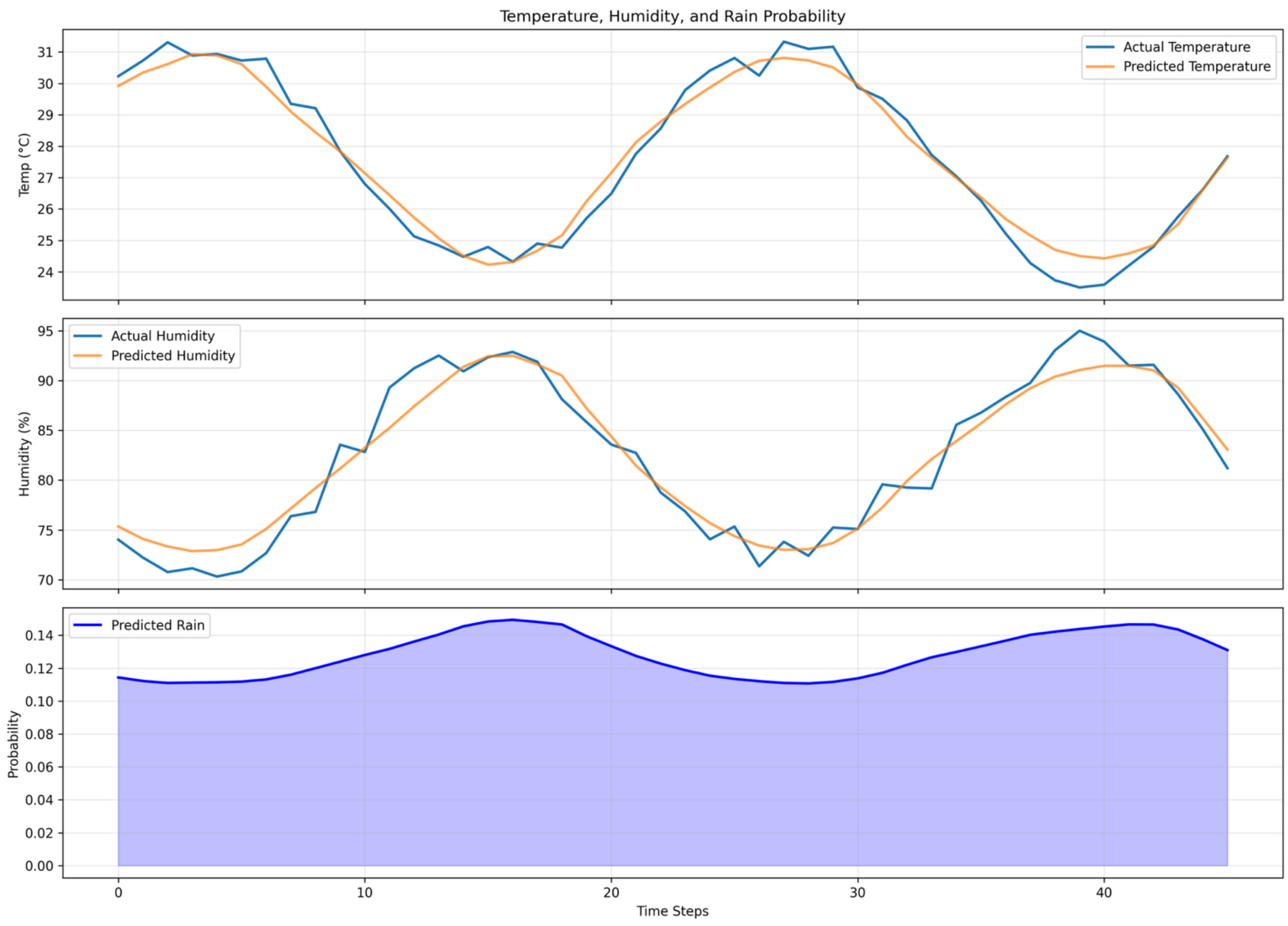
- The second part of the program then reads data from a CSV file containing temperature and humidity readings. It then normalizes the data to the (0,1) range for better neural network performance.
- Then it converts raw time-series data into overlapping sequences suitable for LSTM, with each sequence used to predict the next value (It also splits data into different sets for training and testing).
- Then the program uses the trained LSTM to predict temperature and humidity for test sequences. Then inversely transforms predictions back to their real-world scale for interpretability.

3. Historical Weather Fetching

- Then the program downloads historical weather data (temperature, humidity, precipitation) for the specified location from where real data was logged from Open-Meteo's API.
- It then processes API data to mark time points where rainfall (precipitation > 0 mm) occurred.
- It then trains a logistic regression model to predict the probability of rain using historical temperature and humidity as input features.

4. **Rain Probability Forecasting from LSTM**

- As mentioned above, the program feeds the LSTM's predicted temperature and humidity to the logistic regression model to forecast rain probability for each time step.
- Then for ease of visualization and understanding, we create a combined figure with three subplots:
    - Actual vs predicted temperature
    - Actual vs predicted humidity
    - Predicted probability of rain from the logistic model



5. Data Utilized

1. **Local CSV Data (humidity_data.csv)**
    - Purpose: Used to train an LSTM neural network
    - Variables Obtained:
        - Temperature
        - Humidity
        - Time

2. **Historical Weather Data (from Open-Meteo API)**
    - Retrieved via the function fetch_historical_weather()
    - Variables fetched hourly:
        - Temperature
        - Humidity
        - Rainfall amount (Precipitation)
    - Constraint to check for rain:
        - rain = 1 if precipitation > 0, else 0 (rain/no rain)
    - Purpose: Used to train a logistic regression model that predicts rain probability based on temperature and humidity.

6. Algorithms and Techniques Implemented

1. **Long Short-Term Memory (LSTM) Neural Network**
    - Model type: Sequential
    - Loss: Mean Squared Error (mse)
    - Metric: Mean Absolute Error (mae)
    - Early stopping: stops training when validation loss stops improving

2. **Logistic Regression**
    - Model type: LogisticRegression(max_iter=1000)
    - Input features: ['temperature', 'humidity']
    - Output: Probability that rain will occur (predict_proba())