

1. System Overview

The system is divided into two main components:

1. **Prediction Engine (`LZomato_updated.py`):** A Deep Learning model (LSTM) that fetches historical data and predicts the next 7 days of stock price "High" and "Low" values.
 2. **Hardware Link (`send_to_esp32.py`):** A bridge script that reads the predictions and sends them to an external ESP32 device via a USB serial port (COM7).
-

2. Technical Component: The Prediction Engine

The model utilizes a **Long Short-Term Memory (LSTM)** neural network, which is specifically designed for time-series forecasting.

Data Flow

1. **Data Acquisition:** Uses `yfinance` to download historical data for a specific ticker (default: ETERNAL.NS).
2. **Preprocessing:** * Filters features to High, Low, and Close.
 - Normalizes data using `MinMaxScaler` to a range of $[0, 1]$.
 - Creates a sliding window of **60 days** of past data to predict the next day.

3. Model Architecture:

- **Layer 1:** LSTM (100 units) with return sequences enabled.
- **Layer 2:** Dropout (20%) to prevent overfitting.
- **Layer 3:** LSTM (100 units).
- **Layer 4:** Dropout (20%).
- **Layer 5:** Dense (50 units).
- **Layer 6:** Dense (2 units) – Outputs the predicted **High** and **Low**.

Outputs

- `next7days_high_low.csv`: Contains the date, predicted high, and predicted low.
 - `next7days_range.csv`: Contains a randomized close price estimate for range visualization.
-

3. Technical Component: The Hardware Link

This script acts as the communication interface between the PC and the ESP32.

Execution Logic

- **Serial Configuration:** Opens a connection on COM7 at a baud rate of 115200.
 - **Data Parsing:** Reads the CSV file generated by the prediction engine.
 - **Transmission:**
 - Iterates through each row (representing one future day).
 - Formats the string as: YYYY-MM-DD,Low-High\n.
 - Sends the encoded string to the ESP32.
 - **Interval:** Implements a time.sleep(4) delay to ensure the ESP32 has enough time to process and display the text on its screen (e.g., OLED or LCD).
-

4. Setup & Usage

Prerequisites

- **Python Libraries:** numpy, pandas, yfinance, scikit-learn, tensorflow, pyserial.
- **Hardware:** ESP32 connected via USB.

Execution Steps

1. Run Prediction: ``bash

```
python LZomato_updated.py
```

Wait for the model to train (20 epochs) and save the CSV.

2. **Upload to Hardware:**

Bash

```
python send_to_esp32.py
```

Ensure your ESP32 is flashed with code to read Serial input and display it.

5. Potential Improvements

- **Dynamic Port Detection:** Update send_to_esp32.py to automatically detect the COM port instead of hardcoding COM7.
- **Model Refinement:** Increase the number of epochs or add more technical indicators (like Moving Averages) to the input features.

- **Error Handling:** Add a try-except block around the serial write process to handle accidental cable disconnections.

How to Modify the Data Range

In the provided script, the `yf.download` function controls the timeframe of the historical data retrieved.

- **Locate the Download Line:** Look for the line where `ticker` is used in the `yf.download` function.
 - **Change start and end:** Update the strings to your desired dates in YYYY-MM-DD format.
 - **Example:** To train on data from January 2018 to December 2023, change the line to: `df = yf.download(ticker, start="2018-01-01", end="2023-12-31", progress=False)`.
-

Impact of Range Changes on the Model

Adjusting the training range significantly affects the LSTM's performance and data structure:

- **Data Volume:** Increasing the range provides more records, which generally helps the LSTM learn long-term patterns better.
- **Relevance:** Financial markets change over time; using data from too far in the past may introduce "noise" that is no longer relevant to current market conditions.
- **Look-back Consistency:** The `look_back` variable is set to 60. This means that regardless of the total range, the model always looks at the previous 60 days to predict the next day.
- **Prediction Alignment:** The prediction logic in **Step 5** starts from the very last available date in your downloaded range (`df.index[-1]`).

Important Note on the End Date

In your current script, the end date is set to `"2025-11-09"`. Because that date is in the future relative to the current time, `yfinance` will simply download all available data up to the **most recent trading day**. If you want a specific historical cutoff, ensure the `end` date is set to a day that has already passed.