**Name:** Suyog Bodke  
**Roll No:** 381006  
**PRN:** 22310072

**Assignment No: -** 4  
**Title: -** Time Series Prediction using RNN – Stock Market Analysis or Weather Forecasting

**Problem Statement:**

Implement **time series prediction** using **Recurrent Neural Networks (RNNs)** such as LSTM or GRU for applications like **stock market analysis** or **weather forecasting**.

**Objective:**

* To learn time series modeling using RNN architectures (LSTM/GRU).
* To apply preprocessing (scaling, sequence generation) on time-dependent data.
* To train an RNN model for predicting future values.
* To evaluate the model using test loss and visualize predictions.
* To forecast future time steps (next 20 days of weather temperature).

**S/W Packages and H/W apparatus used:**

* **Operating System:** Windows/Linux/MacOS
* **Kernel:** Python 3.x
* **Tools:** Jupyter Notebook / Google Colab
* **Hardware:** GPU (optional)
* **Libraries:** Pandas, NumPy, Matplotlib, scikit-learn, TensorFlow/Keras

**Theory:**

**Time Series Prediction** involves forecasting future values based on past data.  
Traditional models (ARIMA, Exponential Smoothing) capture linear dependencies, but they fail with complex, nonlinear patterns.

**Recurrent Neural Networks (RNNs)** and their variants (LSTM, GRU) are widely used because:

* They capture temporal dependencies using memory cells.
* LSTMs (Long Short-Term Memory) solve vanishing gradient problems by maintaining long-term context.
* GRUs (Gated Recurrent Units) simplify computations with fewer parameters.

For this assignment:

* Input: Past **10 days’ temperature (Temp3pm)**
* Output: Next **1-day forecast**

**Methodology:**

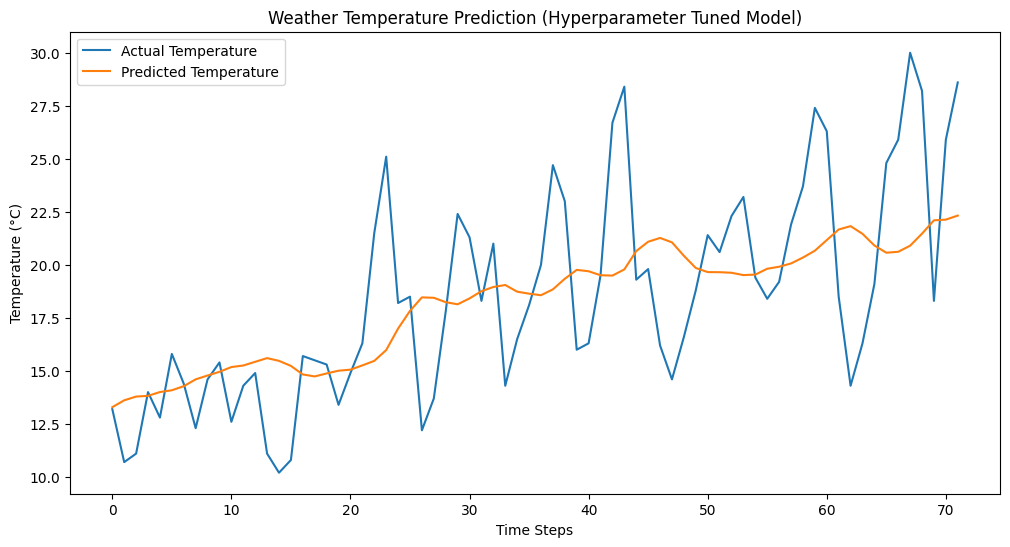
1. **Dataset Loading:** Weather dataset with temperature values.
2. **Preprocessing:**
   * Handle missing values with forward fill.
   * Normalize data using MinMaxScaler.
   * Generate input sequences (10 timesteps → next value).
3. **Data Splitting:** 80% train, 20% test.
4. **Model Architecture:**
   * LSTM(50 units) + Dropout
   * LSTM(50 units) + Dropout
   * Dense output (1 step forecast)
5. **Compilation:** Optimizer = Adam, Loss = Mean Squared Error.
6. **Training:** 10 epochs, batch size = 32.
7. **Evaluation:** Test loss calculation.
8. **Forecasting:** Predict next 20 days’ temperature.
9. **Visualization:** Plot actual vs. predicted temperatures.

**Results:**

* **Training Loss:** Converged to ~0.03
* **Validation Loss:** ~0.015
* **Test Loss:** **0.0187**
* **Next 20 Days Forecast:** Ranged between ~22.0°C and 23.8°C

Graph:

* Line graph showing **actual vs. predicted temperature**.



**Advantages:**

* Captures temporal dependencies effectively.
* Can be applied to both univariate and multivariate time series.
* Outperforms traditional forecasting methods for nonlinear data.

**Limitations:**

* Requires more computational power.
* Sensitive to hyperparameters (sequence length, units, dropout).
* Predictions may drift for long horizons.

**Applications:**

* Stock market trend prediction.
* Weather forecasting (temperature, rainfall, wind speed).
* Energy demand forecasting.
* Sales and financial time series predictions.

**Working / Algorithm:**

1. Preprocess time series data.
2. Scale and convert into supervised learning format (X, y).
3. Build LSTM-based RNN model.
4. Train and evaluate model.
5. Forecast next 20 steps.
6. Plot actual vs. predicted results.

**Conclusion:**

The LSTM-based RNN model successfully forecasted **future weather temperature values** with a test loss of **0.0187**. The model captured the time dependencies well, making it suitable for real-world forecasting tasks like **stock market analysis** and **weather prediction**.