# **Assignment 4: Time Series Prediction using RNN (LSTM) – Google Stock Price Analysis**

## **Problem Statement**

Stock market prices are highly volatile and influenced by numerous factors, making prediction a challenging task. Traditional statistical models (e.g., ARIMA) fail to capture complex **temporal dependencies** in sequential financial data.

This assignment focuses on using **Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM)** to predict Google stock prices based on historical data.

## **Objective**

* To understand **time series forecasting** using deep learning.
* To implement **LSTM-based RNN** for predicting future stock prices.
* To evaluate how well the model learns long-term dependencies in financial data.

## **Requirements**

* **Operating System**: Windows / Linux / macOS (Google Colab recommended)
* **IDE / Platform**: Jupyter Notebook / Google Colab

### **Libraries and Packages Used**

* **NumPy** → Numerical computations.
* **Pandas** → Data manipulation & preprocessing.
* **Matplotlib / Seaborn** → Data visualization.
* **Scikit-learn** → Feature scaling.
* **TensorFlow / Keras** → Building and training LSTM model.

## **Theory**

### **Definition**

* **Recurrent Neural Networks (RNNs)** are deep learning models designed to work with **sequential data** by maintaining memory of past inputs.
* **LSTMs (Long Short-Term Memory networks)** are a special type of RNN that solves the **vanishing gradient problem**, allowing the model to learn long-term dependencies in time series data.

### **Structure**

1. **Input Layer** – Sequences of stock prices (e.g., past 60 days).
2. **LSTM Layers** – Capture temporal patterns and dependencies.
3. **Dense Layer** – Maps LSTM outputs to the predicted stock price.
4. **Output Layer** – Single neuron for final prediction (stock price).

## **Methodology**

1. **Data Collection**
   * Use **Google Stock Price dataset** (Train & Test files).
   * Select relevant column (e.g., "Open" price).
2. **Preprocessing**
   * Normalize stock price data using **MinMaxScaler**.
   * Convert time series into supervised learning format (past 60 days → next day prediction).
   * Split into training and test sets.
3. **Model Building**
   * Define an **LSTM-based RNN** using Keras.
   * Compile model with optimizer (adam) and loss function (mean\_squared\_error).
4. **Training & Prediction**
   * Train model on training dataset.
   * Predict on test dataset.
   * Rescale predictions back to original values.
5. **Evaluation**
   * Plot predicted vs actual stock prices to visualize performance.

## **Advantages**

* Captures **long-term temporal dependencies** in stock data.
* Better than traditional models (ARIMA, regression) for sequential forecasting.
* Flexible and generalizable to other time series (weather, sales, traffic, etc.).

## **Limitations**

* Requires **large datasets** for accurate learning.
* Computationally expensive and time-consuming.
* Susceptible to **overfitting** if not regularized properly.
* Stock prices are influenced by external factors (news, economy, politics), which are not captured in pure historical data.

## **Working / Algorithm**

1. Load stock price dataset.
2. Preprocess: scale values & create input sequences.
3. Build LSTM model with stacked layers.
4. Train model on historical stock data.
5. Predict stock prices on test set.
6. Compare **actual vs predicted** prices with visualization.

## **Conclusion**

The LSTM-based RNN effectively predicts stock prices by learning from historical data, demonstrating the power of deep learning in **time series forecasting**. While predictions capture general trends, they cannot fully account for external unpredictable factors. This makes the approach valuable for **trend analysis**, but not for guaranteed financial decision-making.