Assignment 4

**1. Aim**

To implement a Recurrent Neural Network (RNN) model for time series prediction using stock market data or weather forecasting data and evaluate the accuracy of the predictions.

**2. Objectives**

* To understand the theory behind time series analysis and prediction using RNNs.
* To implement an RNN model using Python (TensorFlow/Keras) for stock market or weather forecasting data.
* To preprocess the data and train the RNN model to predict future values based on historical data.
* To evaluate and optimize the performance of the model using appropriate metrics like Mean Absolute Error (MAE) or Mean Squared Error (MSE).

**3. Theory**

**Time Series Prediction** refers to the use of models to predict future values based on previously observed data points in time. Common examples of time series data include stock prices and weather data, where patterns from the past can help predict future trends.

**Recurrent Neural Networks (RNNs)** are a class of neural networks specifically designed for sequence data, such as time series data. The main feature of RNNs is their "memory" capability, which allows them to retain information from previous time steps and use that information in the current prediction.

* **RNN Architecture:** Unlike feedforward neural networks, RNNs have loops that allow information to persist. This makes them particularly effective in tasks where the order of inputs is critical, such as time series prediction.
* **Long Short-Term Memory (LSTM):** A special type of RNN, LSTMs are capable of learning long-term dependencies. They are widely used in time series predictions due to their ability to retain information over long sequences.

**Applications:**

* **Stock Market Analysis:** Predicting future stock prices based on historical stock data.
* **Weather Forecasting:** Predicting future temperature, humidity, or precipitation based on historical weather data.

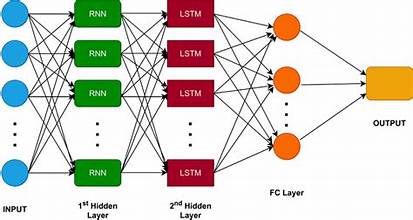
**4. Working/Algorithm Used**

**4.1 Data Collection and Preprocessing:**

1. **Data Collection:**
   * Stock market data or weather data is collected from sources like Yahoo Finance (for stock prices) or meteorological data websites (for weather forecasting).
2. **Data Preprocessing:**
   * Normalize the data to bring all values between 0 and 1, improving the performance of the neural network.
   * Create sequences of data where each sequence consists of a fixed number of time steps (e.g., previous 60 days for stock prediction).
   * Split the dataset into training, validation, and test sets.

**4.2 RNN/LSTM Model Architecture:**

1. **Building the Model:**
   * Use the Sequential() model from Keras.
   * Add an LSTM layer with a specific number of units (neurons), which will capture the temporal dependencies in the data.
   * Optionally, add Dropout layers to prevent overfitting.
   * Add a Dense layer at the end for predicting the next value in the series.
2. **Compilation:**
   * Use the Mean Squared Error (MSE) or Mean Absolute Error (MAE) as the loss function for regression tasks.
   * Use the Adam optimizer for better convergence.

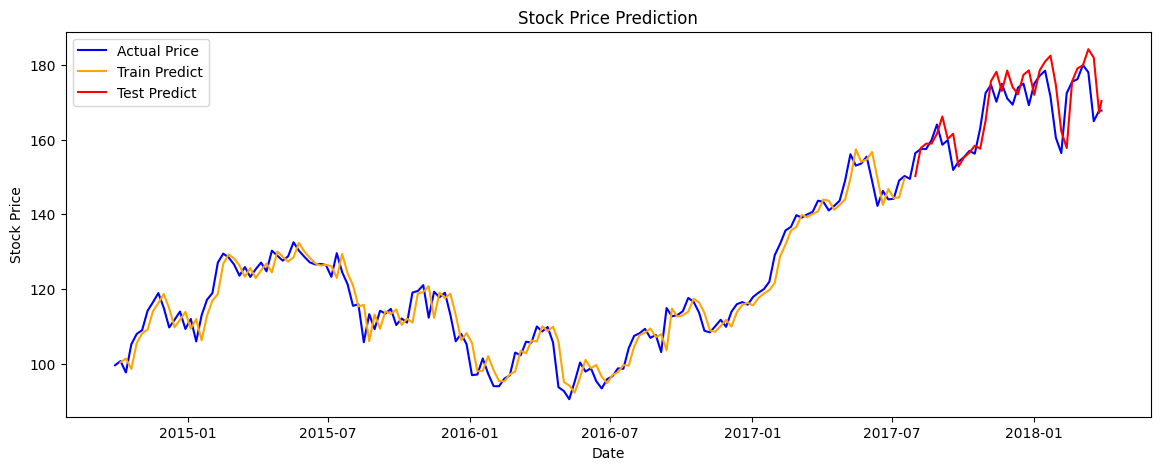


**4.3 Model Training and Prediction:**

1. **Training:**
   * Train the RNN model on the training dataset using the fit() function in Keras.
   * Monitor the performance on the validation dataset during training to detect overfitting.
2. **Prediction:**
   * Once the model is trained, use it to predict future values of the time series, such as the stock price or temperature for the next day.

**4.5 Evaluation:**

* The model is evaluated using metrics like **Mean Absolute Error (MAE)** or **Mean Squared Error (MSE)** on the test dataset.
* The predicted values are compared to the actual values to assess the performance of the RNN model.



**5. Conclusion**

In this project, an RNN (specifically an LSTM) was implemented for time series prediction. The model was trained using historical stock market or weather data, and it successfully predicted future values with a reasonable degree of accuracy. LSTMs demonstrated their ability to capture temporal dependencies in the data. However, further improvements can be made by tuning hyperparameters, increasing the dataset size, and experimenting with other RNN architectures.