**Assignment 4**

**Problem Statement:** Time Series Prediction using RNN – Stock Market Analysis.

**Libraries:**

The following libraries are used in this implementation:

* **NumPy**: For handling array-based computations.
* **Pandas**: To read and manipulate time series data.
* **Matplotlib**: For plotting the stock price predictions.
* **scikit-learn (MinMaxScaler)**: To scale the stock price data.
* **Keras (Sequential, Dense, SimpleRNN)**: To build and train the RNN model.

**Theory:**

Recurrent Neural Networks (RNNs) are designed to recognize patterns in sequences of data, which makes them highly effective for time series prediction tasks. Unlike feedforward neural networks, RNNs have connections that form cycles, enabling the network to "remember" previous outputs and use them as inputs for future predictions. This feature makes RNNs especially useful in scenarios where historical data is essential for predicting future events, such as in stock market analysis.

The SimpleRNN layer is the basic building block of RNNs. In this architecture:

* Units: The number of neurons in the RNN cell.
* Input Shape: The number of time steps (sequence length) and features in each step.

Time series prediction involves forecasting future values based on previously observed data points in a sequence. In the context of stock market analysis or weather forecasting, time series prediction aims to predict future stock prices or weather conditions using historical data. Recurrent Neural Networks (RNNs) are a type of neural network that excels in processing sequential data by utilizing internal memory to capture dependencies between time steps.

1. Recurrent Neural Networks (RNNs)

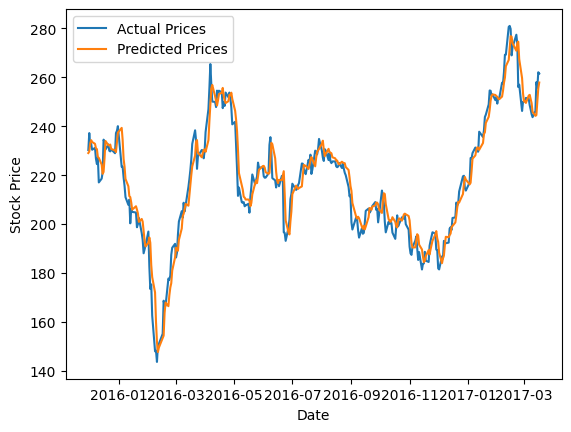
RNNs are designed to work with sequential data, making them particularly useful for tasks involving time series, such as stock price prediction, weather forecasting, and natural language processing. Unlike traditional neural networks, which assume that all inputs (and outputs) are independent of each other, RNNs consider the time dependencies between the data points. This ability to "remember" information from previous time steps allows RNNs to model time series data more effectively.

**2. Stock Market Time Series**

Time series data for stock market analysis typically consists of a sequence of prices (e.g., opening, closing, high, low) indexed by time (usually in days). The key challenge is to model the temporal dependencies between these prices to forecast future prices. In this implementation, we focus on using the closing price of a stock, which represents the final price of the stock at the end of the trading day.

**Methodology:**

1. **Data Preprocessing**:
   * The stock data (Tesla in this case) is loaded and the Close prices are extracted.
   * The MinMaxScaler is used to normalize the data between 0 and 1 to improve the convergence of the model.
2. **Sequence Generation**:
   * A sliding window technique is used to generate sequences of stock prices. This creates an input sequence of n days to predict the price on the n+1 day.
3. **RNN Model**:
   * A simple RNN model is built using the Keras Sequential API. It consists of one RNN layer with 50 units followed by a dense layer to predict the stock price.
4. **Training**:
   * The model is trained on 80% of the dataset and evaluated on the remaining 20%. The loss function used is Mean Squared Error (MSE).
5. **Prediction**:
   * The model uses the last 60 days of training data to predict the next 20 days of stock prices.
6. **Evaluation and Visualization**:
   * The predicted prices are compared with the actual prices and visualized using a line plot to see how closely the model predictions match the actual data.



**Advantages:**

1. **Temporal Dependency**: RNNs effectively capture the temporal dependencies in time series data, making them well-suited for stock price predictions.
2. **Sequential Data Handling**: RNNs can handle input sequences of variable lengths, which is essential for real-world time series data.
3. **Ease of Integration**: RNNs can be easily integrated into a range of time-dependent prediction problems beyond stock market analysis, such as weather forecasting, anomaly detection, etc.

**Disadvantages:**

1. **Vanishing Gradient**: RNNs can suffer from vanishing gradient problems, which makes it hard for them to remember long sequences of data.
2. **Limited to Short Sequences**: For longer sequences, simple RNNs struggle, and more advanced architectures like LSTMs or GRUs are preferred.
3. **Data Sensitivity**: RNNs are sensitive to data normalization and preprocessing. Small changes in data can result in large variances in output.
4. **Computational Complexity**: RNNs are computationally expensive compared to traditional feedforward neural networks due to their recurrent nature.

**Conclusion:**

Time series prediction using RNNs for stock market analysis offers a powerful approach to model temporal dependencies. While RNNs can be used to predict future stock prices based on historical data, their limitations like vanishing gradients and handling long sequences need to be addressed. For more complex tasks, architectures like LSTMs or GRUs may offer better performance, especially when dealing with long-term dependencies. ​