**Assignment No:** 3

**Problem Statement:** Implement Time Series Prediction using Recurrent Neural Networks (RNNs) for stock market or weather forecasting.

**Objective:** - To understand the architecture and working of Recurrent Neural Networks. - To learn how to preprocess time series data for training RNNs. - To implement an RNN model using Keras and TensorFlow for predicting future values. - To evaluate model performance using validation data. - To visualize training loss and prediction results over epochs.

**S/W Packages and H/W apparatus used:** - Operating System: Windows/Linux/MacOS - Kernel: Python 3.x - Tools: Jupyter Notebook, Anaconda, or Google Colab - Hardware: CPU with minimum 4GB RAM; optional GPU for faster processing - Libraries: TensorFlow, Keras, NumPy, Matplotlib, Pandas, scikit-learn

**Theory:** Recurrent Neural Networks (RNNs) are deep learning algorithms designed for sequential data such as time series. They maintain internal memory, enabling them to capture temporal dependencies and patterns in sequences, making them suitable for predicting stock prices or weather metrics.

**Structure:** - Input Layer: Receives sequences of historical data. - RNN Layers: Process sequences and learn temporal dependencies. - Dense Layer: Maps RNN outputs to predicted values. - Output Layer: Produces continuous predictions for the next time step(s).

**Activation Functions:** - ReLU or tanh for hidden layers - Linear for output layer

**Backpropagation Through Time (BPTT):** Gradients are calculated across time steps and used to update network weights to minimize loss.

**Methodology:** 1. Data Acquisition: Load historical stock or weather data. 2. Data Preparation: - Normalize values between 0 and 1. - Create sequences of fixed time steps. 3. Model Architecture: - Sequential model using Keras. - First RNN layer with 50 units and return sequences. - Second RNN layer with 50 units. - Dense layer with 1 unit for output. 4. Model Compilation: Adam optimizer, Mean Squared Error loss. 5. Model Training: Fit the model on training data, validate on test data. 6. Model Evaluation: Measure performance using RMSE on test dataset. 7. Prediction Visualization: Plot actual vs predicted values to assess model performance.

**Advantages:** - Captures temporal dependencies in sequential data. - Suitable for forecasting continuous values. - Can handle variable-length sequences.

**Limitations:** - Requires sufficient historical data. - Can be computationally expensive for long sequences. - Risk of vanishing/exploding gradients. - Sensitive to hyperparameters and sequence length.

**Applications:** - Stock price prediction - Weather forecasting - Energy consumption forecasting - Sales and demand prediction

**Working / Algorithm:** 1. Load Dataset: Historical stock or weather data is loaded and split into training and test sets. 2. Preprocess Data: Normalize values to [0,1] and create sequences for input. 3. Visualize Data: Plot historical data to understand trends. 4. Define RNN Model Architecture: - RNN layers with 50 units each. - Dense layer with 1 output unit. 5. Compile the Model: Use Adam optimizer and Mean Squared Error loss. 6. Train the Model: Train for multiple epochs while validating on test data. 7. Evaluate the Model: Compute RMSE to assess prediction accuracy. 8. Visualize Predictions: Plot actual vs predicted values to compare performance.

**Conclusion:** Recurrent Neural Networks are powerful for time series forecasting, effectively learning patterns and temporal dependencies. Proper preprocessing, sequence design, and hyperparameter tuning are critical for accurate predictions in stock market or weather datasets.