# Assignment 4 – Time Series Prediction using Recurrent Neural Networks (RNNs)- stock market analysis or weather forecasting

Name : Aayush Bokde

Roll No: 382016

PRN: 22310316

## Problem Statement

Implement a Recurrent Neural Network (RNN) in Python using Keras and TensorFlow to predict stock market prices based on historical data. The model is trained on Apple (AAPL) stock closing prices from 2018 to 2023 to forecast future values.

## Objectives

* To understand the working of Recurrent Neural Networks (RNNs) for time series forecasting.
* To preprocess stock market data using normalization and sequence generation.
* To build, compile, and train an RNN model for stock price prediction.
* To evaluate model performance using predicted vs. actual prices.
* To visualize learning curves and stock price predictions.

## Requirements

* Operating System: Windows/Linux/MacOS
* Python Version: 3.x
* Tools: Jupyter Notebook / Anaconda / Google Colab
* Libraries Used: TensorFlow, Keras, NumPy, Pandas, Matplotlib, Scikit-Learn, yFinance

## Theory

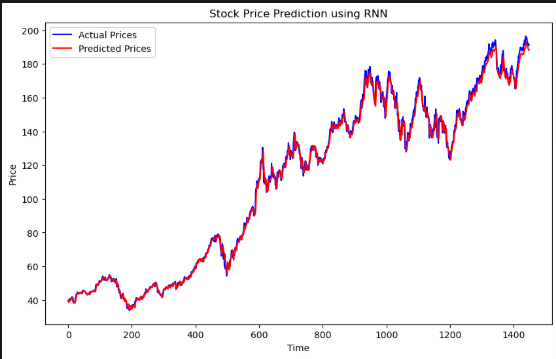
Stock price prediction is a time series forecasting task where past stock values are used to estimate future prices. Traditional machine learning methods struggle with sequential dependencies, hence Recurrent Neural Networks (RNNs) are used as they can maintain memory of past information.  
  
Input Layer: Takes stock price sequences (past 60 days).  
RNN Layer (SimpleRNN): Learns temporal dependencies between stock prices.  
Dense Layer: Produces the next-day stock price prediction.  
Output: Predicted stock closing price.

## Methodology

1. Data Acquisition: Apple (AAPL) stock prices downloaded using yfinance.
2. Data Preparation: Extract closing prices, normalize data, and create sequences (60 days → next day).
3. Model Architecture: SimpleRNN layer with 50 units, Dense layer with 1 neuron.
4. Model Compilation: Optimizer = Adam, Loss Function = Mean Squared Error (MSE).
5. Model Training: 20 epochs, batch size 32, validation split 20%.
6. Model Evaluation: Predictions generated on test set and compared with actual prices.
7. Visualization: Plot actual vs. predicted stock prices.

## Graphs and Visualizations

• Predicted vs. Actual Stock Prices: Line graph comparing real and predicted values.



## Advantages

* RNNs are well-suited for sequential/time series data.
* Captures short-term dependencies between stock values.
* Easy to implement using Keras.

## Limitations

* RNNs struggle with long-term dependencies (vanishing gradient problem).
* Stock markets are influenced by external factors (news, sentiment) not captured in historical data.
* Predictions may not generalize well without more features.

## Applications

* Stock market forecasting and algorithmic trading.
* Weather and climate prediction.
* Demand forecasting in retail and supply chain.
* Predictive maintenance in IoT systems.

## Working / Algorithm

1. Import required libraries.
2. Load stock data using yfinance.
3. Preprocess data (normalize, sequence generation).
4. Build RNN model (SimpleRNN + Dense).
5. Compile model (Adam optimizer, MSE loss).
6. Train model on training data.
7. Evaluate and predict stock prices.
8. Plot results for analysis.

## Conclusion

A Recurrent Neural Network was implemented to predict stock prices of Apple (AAPL) based on past data. The model successfully captured short-term trends and demonstrated how RNNs can be applied for time series forecasting. While predictions aligned closely with actual prices, improvements can be made using more advanced models like LSTMs/GRUs and incorporating external market indicators.