**Microsoft Stock Price Prediction using Stacked LSTM Model**

**Project Overview:**

This project aims to predict Microsoft (MSFT) stock prices from January 01, 2014, to October 17, 2024, using a **Stacked Long Short-Term Memory (LSTM)** neural network. The model is trained on historical stock price data and forecasts future prices based on identified patterns. The Stacked LSTM architecture is particularly well-suited for financial time series data, capturing both short-term and long-term temporal dependencies.

**Key Features:**

- Historical stock prices of Microsoft (MSFT) from 2014-01-01 to 2024-10-17

- Stacked LSTM model for capturing complex temporal patterns

- Performance Metrics:

- Training RMSE: 7.22

- Testing RMSE: 30.03

- Forecast: The model predicts a significant drop in MSFT stock prices for the next 30 days

**Key Objectives:**

1. Build a stacked LSTM model to predict future stock prices.
2. Evaluate the model’s performance using various metrics like **RMSE (Root Mean Squared Error)**.
3. Visualize both actual and predicted stock prices.

**Table of Contents:**

1. [Installation](#installation)
2. [Data Preprocessing](#data-preprocessing)
3. [Model Architecture](#model-architecture)
4. [Results](#results)
5. [Visualization](#visualization)
6. [Conclusion](#conclusion)

**1. Installation:**

To run this project, you'll need the following libraries:

* TensorFlow
* Keras
* NumPy
* Pandas
* Matplotlib
* Scikit-learn

**2. Data Preprocessing:**

* **Data Source and Description:** The dataset consists of Microsoft stock prices from January 2014 to October 2024 was obtained from yfinance using some snippets code. The primary feature used for prediction is the **closing price**, which is often the most reflective of the stock’s overall performance for the day.

Key features include:

* **Open**: Opening price of the stock.
* **High**: Highest price of the stock during the day.
* **Low**: Lowest price of stock during the day.
* **Close**: Closing price of the stock (target variable).
* **Volume**: The total volume of shares traded during the day.
* **Missing Data:** There was no missing data.
* **Feature Selection:** The Close column was selected for analysis and chart plotted.
* **Normalization:** The closing stock prices are normalized between 0 and 1 to make the model training more efficient using the MinMaxScaler.
* **Train-Test Split:** The dataset is split into a training set and a testing set to evaluate model generalization.

**3. Model Architecture and Breakdown**

The code initializes a Sequential LSTM model for time series prediction. It consists of four LSTM layers with increasing units (50, 60, 80, and 120) and corresponding dropout layers to prevent overfitting. The first three LSTM layers return sequences, while the final layer outputs a single value for regression. The model is compiled using mean squared error as the loss function and the Adam optimizer for efficient training. This setup is designed to capture temporal patterns in the data while maintaining generalization.

**4. Results:**

* **Training RMSE:** 7.22
* **Testing RMSE:** 30.03

This indicates that the model performs much better on the training data compared to the test data, showing a significant gap in RMSE. This is a common sign of overfitting, where the model is too closely tailored to the training data and struggles to generalize to new, unseen data (test set).

* **Prediction for the Next 30 Days:** The model predicts a **slight significant drop** in MSFT stock prices over the next month. **(October 18, 2024 – November 18, 2024)**

**5. Visualization:**

The project includes various plots to visualize both past stock trends and future predictions:

* **Historical Stock Prices vs. Predictions**
* **Next 30 Days Forecast**

**6. Conclusion:**

* The forecast indicates a slight decline in prices for the next 30 days. **(October 18, 2024 – November 18, 2024)**
* Stacked LSTMs offer the ability to model complex temporal relationships in stock market data, making them well-suited for predicting stock prices, market trends, and other time series tasks. They provide more robust and flexible models, especially when there are long-term dependencies and multiple influencing factors. However, careful tuning and regularization are needed to balance complexity and performance.

**7. Future Work:**

* **Model Tuning:** Further hyperparameter optimization can improve the accuracy of the model.
* **Feature Engineering:** Incorporating additional features (e.g., market sentiment, trading volume) could enhance predictive performance.
* **Other Stocks:** Extending this methodology to other stocks or assets can help test the model's versatility.