**STOCK PRICE PREDICTION USING LSTM**

**TEAM** : Arvind\_mera\_sathi

We have implemented a stock price prediction model using LSTM model, the process involves:

1**. Data Loading**: It loads two CSV files, `train.csv` and `test.csv`, into Pandas DataFrames.

2. **Data Transformation**:

* *Pivoting*: The `train` DataFrame is pivoted based on the columns 'Date' and 'Company', and selects columns 'Open', 'High', 'Low', 'Volume', and 'Id' for each company. This rearranges the data into a more structured format.
* *Flattening MultiIndex*: After pivoting, the columns become MultiIndex columns. The code flattens these columns into single-level columns for easier manipulation.
* *Resetting Index*: The index of the pivoted DataFrame is reset for better structure and organization.

3. **Feature and Target Preparation**:

* The closing price data, `df\_closing`, is separated from the rest of the features as it's the target variable.
* Columns 'Id' and 'Date' are dropped from the feature DataFrame (`df\_pivoted`) as they are not necessary for modeling.

Now we have prepared our Data and now we will further process this data into out LSTM model for predicting future closing price of stocks.

Now we use time series forecasting model using Long Short-Term Memory (LSTM) neural networks with Keras. Here's a breakdown :

1. **Importing Libraries**: The script starts by importing necessary libraries including NumPy for numerical operations, Pandas for data manipulation, MinMaxScaler from scikit-learn for data normalization, and Keras modules for building and training neural networks.

2. **Data Preparation**:

* The closing price data (`y`) is converted to a NumPy array.
* The data is scaled using MinMaxScaler to bring all values within the range of 0 and 1, which can help neural networks converge faster.

3. **Model Configuration**:

* `window\_size` is defined as 200, representing the number of previous days' data used as input features.
* -`prediction\_days` is defined as 96, indicating the number of days to predict ahead.
* An empty list `predictions\_l` is initialized to store predictions for each company.

4. **Model Training and Prediction**:

* The script iterates over each company's data.
* For each company:
* The training data is created by sliding a window of `window\_size` over the scaled dataset. Each window represents a set of input features (`x\_train`) and the corresponding target value (`y\_train`).
* An LSTM model is defined using Keras Sequential API, consisting of two LSTM layers followed by Dense layers for output.
* The model is compiled with Adam optimizer and mean squared error loss function.
* The model is trained using the training data for one epoch.
* Test data is prepared by selecting the last `window\_size` data points from the scaled dataset for the current company.
* Predictions are made using the trained model on the test data.
* Predictions are appended to the `predictions\_l` list.

After this step , we extract the weight of this LSTM model using Keras , and save them into a text file.

After that:

1. **Reshaping Predictions**: The `predictions\_l` list contains the predictions for each company for the next 96 days. The code reshapes this list into a NumPy array of shape (96, 6), where each row represents the predictions for a single day across all companies.

2.**Inverse Scaling**: The predictions are then inverse-scaled using the `scaler.inverse\_transform()` method to convert them back to the original scale of the closing prices.

3**. Flattening Predictions**: The predictions are flattened into a single list `fl` for further processing.

4. **Updating DataFrame**: The code reads the test dataset and initializes a new 'Close' column with zeros. It then iterates over the rows of the DataFrame and assigns the predicted closing prices to the 'Close' column.

5. **Data Preparation**: The DataFrame is then modified by dropping the 'Company' and 'Date' columns and resetting the index.

6. **Plotting Predictions and Converting to CSV**: For visualization, the code plots the predicted prices for each company separately using Matplotlib.

**Here's why LSTM is chosen:**

The LSTM (Long Short-Term Memory) model is chosen because it's well-suited for analyzing sequences, making it ideal for predicting stock prices, which are influenced by past trends. LSTM's ability to remember long-term patterns, handle non-linear relationships, and adapt to varying data lengths makes it effective for capturing the complexities of financial markets. Its architecture helps mitigate issues like the vanishing gradient problem, commonly encountered in deep learning models. Overall, LSTM is selected for its capability to accurately model the temporal dynamics of stock price data.