Stock Price Prediction Using LSTM Neural Network

Ayush Behera
03323802722

Problem Statement

The project's objective is to construct a machine learning model for predicting future stock prices using a Long Short-Term Memory (LSTM) neural network. LSTMs, a form of Recurrent Neural Network (RNN), are suitable for time series data due to their ability to retain patterns over extended sequences. This model will process historical stock price data to generate predictions that can guide investment decisions. Stock markets are inherently volatile, influenced by many dynamic factors, and accurate price prediction is challenging due to their stochastic nature. Traditional statistical models often struggle with nonlinear patterns, whereas deep learning approaches like LSTMs can effectively learn complex temporal dependencies in data.

Objectives

The main objective is to develop a stock price prediction system using LSTM neural networks and make it accessible, interactive, and practical through a Streamlit web application.

Specifically, the objectives are:

- Learn from historical price trends. Predict future stock prices with improved accuracy.
- Build a next-day price forecast for selected tickers with a clear scope and limits.
- Deliver Interactive Visualization & App Development with an intuitive Streamlit UX: simple controls, and interactive charts.
- Communicate uncertainty (confidence cues/bands) and interpretation.
- Performance Analysis and Validation

Dataset

Source: Yahoo Finance via **yfinance** Python library. Provides historical stock market data for any ticker (e.g., AAPL, TSLA, GOOG)

Key terminologies:

- Date: Trading date (daily, weekly, or chosen interval)
- Open: first traded price of the session at the chosen interval.
- High: highest traded price during the session/interval.
- Low: lowest traded price during the session/interval.
- Close: last traded price of the session/interval.
- Volume: shares traded during the session/interval.
- Adjusted Close: Close adjusted for dividends/splits to maintain continuity for backtests and return series.

THE SOLUTION

Data Collection

Preprocessing and Cleaning

Training LSTM Model

Testing

Streamlit App

Data Collection

Source: Yahoo Finance via the Python yfinance library, returning OHLCV price series for chosen tickers and dates. Provides free and reliable access to historical market data

Import yfinance in Python

Use yf.download(ticker, start, end) to fetch data

We used Google's data for the model training and testing.

```python
import yfinance as yf
```

```python
start = '2014-01-01'
end = '2024-12-31'
stock = 'GOOG'

data = yf.download(stock, start, end)
```

Price	Close	High	Low	Open	Volume
Ticker	GOOG	GOOG	GOOG	GOOG	GOOG
Date					
2014-01-02	27.560261	27.674898	27.439930	27.618199	73129082
2014-01-03	27.359213	27.654593	27.357480	27.606808	66917888
2014-01-06	27.664251	27.702380	27.394868	27.557538	71037271
2014-01-07	28.197571	28.218122	27.759330	27.854405	102486711
2014-01-08	28.256250	28.407036	28.059659	28.374353	90036218
...	...	...	...	...	...
2024-12-23	195.531937	196.030768	191.182126	193.576511	15235900
2024-12-24	197.108261	197.208018	194.741796	195.711524	6809800
2024-12-26	196.639359	197.696879	195.412222	196.280199	7907900
2024-12-27	193.586487	196.340046	191.523327	196.010815	14693000
2024-12-30	192.239655	193.327103	189.915098	190.418923	12209500

2767 rows × 5 columns

Data Preprocessing and Cleaning

Why Preprocessing? Stock data is raw and noisy and models require normalized, structured input for better learning.

Steps in Preprocessing:

1. Check for null/NaN values. Drop or interpolate missing rows

2. Focus on Close Price (target variable)

3. Train–Test Split. 80% training data and 20% testing data

4. Normalization. Apply MinMaxScaler; scale prices to range [0,1]. Ensures stable LSTM training

Training LSTM Model

Model Architecture:

Sequential Model built using Keras

Layers:

- LSTM Layer(s): captures time dependencies in stock prices
- Dropout Layer(s): prevents overfitting
- Dense Layer: fully connected layer to output prediction (1 value: next-day price)

Optimizer: Adam (efficient gradient descent)

Loss Function: Mean Squared Error (MSE)

Hyperparameters:

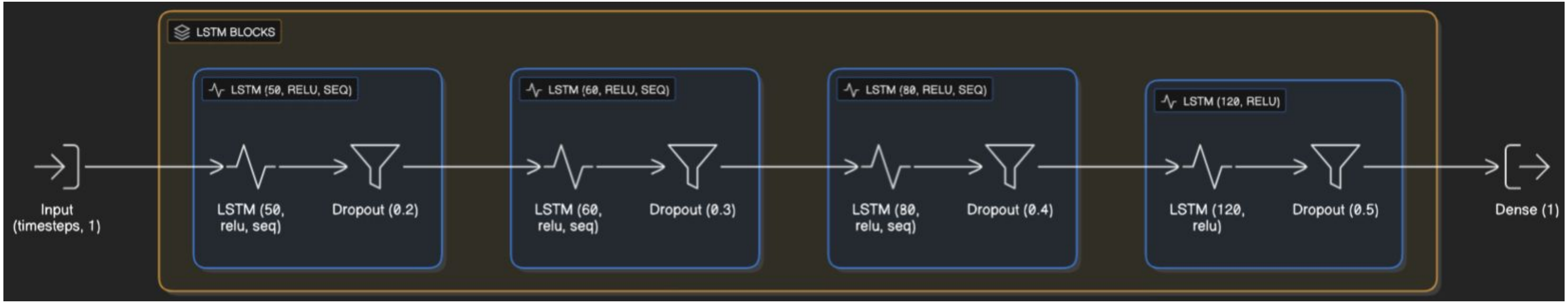
- Epochs = 50
- Batch size = 32
- Input shape = (100 timesteps, 1 feature)

Training LSTM Model (cont.)

Process:

1. Feed sequences of 100 past days (x) to predict next day price (y)
2. Model weights updated using backpropagation through time

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 100, 50)	10,400
dropout (Dropout)	(None, 100, 50)	0
lstm_1 (LSTM)	(None, 100, 60)	26,640
dropout_1 (Dropout)	(None, 100, 60)	0
lstm_2 (LSTM)	(None, 100, 80)	45,120
dropout_2 (Dropout)	(None, 100, 80)	0
lstm_3 (LSTM)	(None, 120)	96,480
dropout_3 (Dropout)	(None, 120)	0
dense (Dense)	(None, 1)	121



Testing

Process:

1. Combine last 100 days of training data with test data for continuity. Normalize using same MinMaxScaler
2. For each day in test set, take previous 100 days as input (x). True next-day price is output (y)
3. Model predicts next-day stock prices from test sequences. Rescale predictions back to actual values
4. Predictions follow actual stock price trends closely. Enables visualization of Predicted vs Actual performance on test dataset.

This phase ensures that the trained model is generalized and performs well on unseen stock data.

Streamlit App

The app is a lightweight Streamlit front end that wraps the trained model to fetch data, run inference, and render KPIs and charts interactively. Streamlit reruns the script on each interaction, so the design uses per-session state to keep selections/results consistent across reruns.

Moving Averages:

Price vs MA50 - Price vs MA50 vs MA100 - Price vs MA100 vs MA200 - zoomable, time-based comparisons for trend insight.

Stock Settings:

- Dropdown to select stock ticker (e.g., NVDA, GOOG, TSLA, AAPL etc.)
- Option to manually enter a symbol
- Start date input for historical data retrieval

Price History:

- Company details: name, sector, data points
- Current price display with real-time change indicator
- Interactive price history chart (zoomable, time-based trends) i.e., a long-range price history chart for quick situational awareness.

Model Performance:
Key Performance Indicators (KPIs) are derived from the same dataset powering the model inputs. They are as follows:

- R^2 Score for accuracy
- MAE (Mean Absolute Error) for error measure
- Model rating indicator (e.g., Good,)

Prediction vs. Actual:

- Compares model predictions with realized closes on a time-aligned plot, alongside headline metrics (e.g., R^2 and MAE) to communicate fit quality.
- Clear comparison using line charts (LSTM predictions). Helps evaluate prediction reliability and trend following

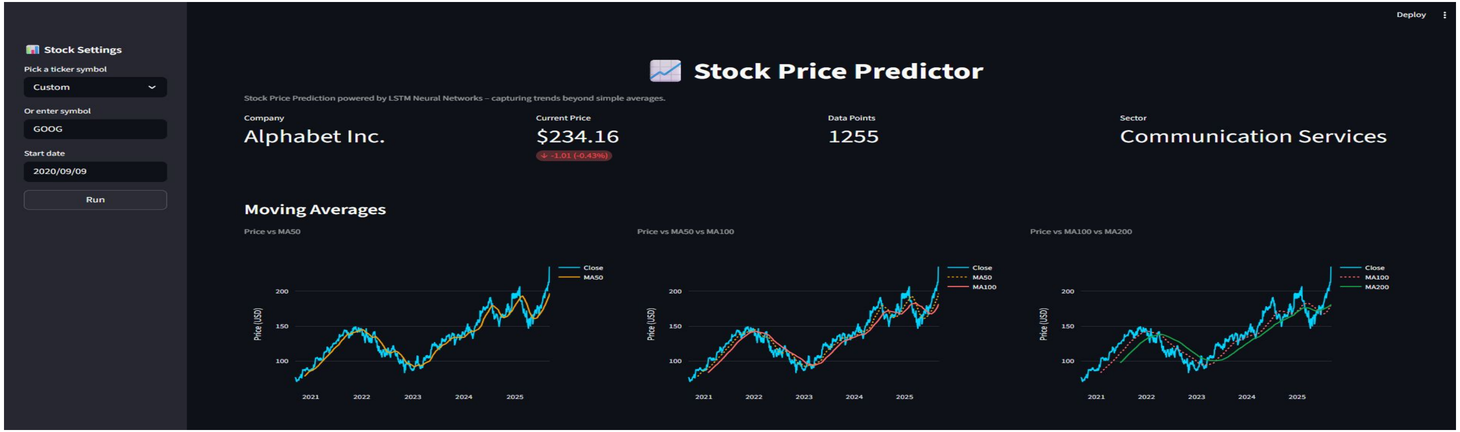
Next-day prediction:

- Displays current closing price,
- Next-day closing price forecast with confidence level
- Lookback period parameter (days of past data used)

Volumes Traded:

- Volume analysis chart with 7-day SMA overlay
- Highlights liquidity and trading activity trends

Preview of Streamlit App



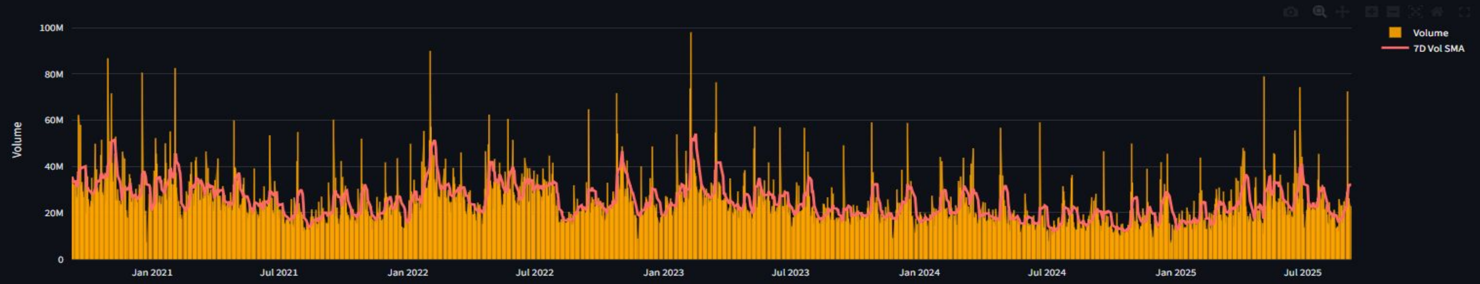
Predictions vs Actual

Next Day Prediction

Current Close	Predicted Next Close	Lookback (days)
$234.16	$196.23 (↓ -37.93)	100

Prediction Confidence: **Medium** (based on test R^2 of 0.603)

Volumes Traded



Dataset from Yahoo Finance API.

Thank you!