A Mini Project Report

on

Stock Market Price Predictor

*carried out as part of the **AI Lab DS3230***

Submitted

by

Deivyansh Singh
219309064
Aditya Shaswat
219309110

Submitted
to
Prof. Abhishek Dwivedi

MANIPAL UNIVERSITY
JAIPUR

Section B

Group No.: 30

School of Information Technology
Department of Data Science and Engineering

MANIPAL UNIVERSITY JAIPUR
RAJASTHAN, INDIA

April 2024

ABSTRACT

Importance and Objective: Predicting stock prices is crucial for informed investment decisions. This project aimed to develop a Long Short-Term Memory (LSTM) model to forecast closing prices for HDFC Bank stock (HDFCBANK.NS) on the National Stock Exchange of India.

Methodology: The project utilized Python libraries like yfinance and TensorFlow (Keras) to:

- Downloaded historical daily closing prices for HDFC Bank from 2012-01-01 to 2023-12-31.
- Pre-process the data by calculating moving averages (100-day and 200-day) and splitting it into training and testing sets.
- Employ MinMaxScaler for normalization and create sequences for LSTM input.
- Build a multi-layered LSTM model with dropout regularization for improved generalization.
- Train the model on the training set for 50 epochs using the Adam optimizer and mean squared error loss function.
- Evaluate the model's performance on the testing set and visualize the predicted prices against the actual closing prices.

Results: The LSTM model successfully learned patterns in the historical data and generated price predictions for the testing period. The visualization presented a comparison between the predicted and actual closing prices, allowing for an assessment of the model's accuracy.

Significance: This project demonstrates the potential of LSTM models for stock price prediction. By further refining the model and incorporating additional technical indicators, investors can gain valuable insights to support their investment strategies.

TABLE OF CONTENTS

1. Introduction, Theory, and Objectives

1.1. Introduction

This project explores the concept of time series forecasting applied to stock prices. It utilizes a Long Short-Term Memory (LSTM) neural network to predict future closing prices for a particular stock based on historical data.

Motivation: Stock price prediction is a challenging yet valuable task for investors and financial analysts. By analysing historical trends and patterns, investors can make informed decisions about buying, selling, or holding stocks.

1.2. Data Acquisition and Pre-processing

The code leverages two main concepts:

1. **Moving Averages:** The code calculates two moving averages, one for 100 days and another for 200 days. Moving averages are used to smooth out short-term price fluctuations and identify potential trends.
2. **LSTM Networks:** LSTM networks are a type of recurrent neural network (RNN) particularly well-suited for time series forecasting. LSTMs can learn long-term dependencies in data, which is crucial for predicting future stock prices.

1.3. Pre-processing

The main objectives achieved in this project are:

- **Data Acquisition**: Downloaded historical closing price data for a chosen stock (HDFCBANK.NS) using the yfinance library.
- **Data Pre-processing**: Cleaned and prepared the data by handling missing values and splitting it into training and testing sets.
- **Feature Engineering**: Implemented MinMaxScaler to normalize the data between 0 and 1.
- **Model Development**: Created a multi-layered LSTM model to predict future closing prices based on the previous 100 days of data.
- **Model Training**: Trained the model on the training data set for 50 epochs using the Adam optimizer and mean squared error loss function.
- **Model Evaluation**: Evaluated the model's performance by comparing its predicted closing prices with the actual closing prices of the testing data set.
- **Model Saving**: Saved the trained model for future use in stock price prediction.

1.4 Recurrent Neural Networks (RNN) and Long Short Term Memory (LSTM)

Long Short-Term Memory (LSTM) is one of many types of Recurrent Neural Network RNN, it's also capable of catching data from past stages and use it for future predictions. In general, an Artificial Neural Network (ANN) consists of three layers:

1) Input layer,

2) Hidden layers,

3) Output layer.

In a neural network that only contains one hidden layer the number of nodes in the input layer always depend on the dimension of the data, the nodes of the input layer connect to the hidden layer via links called 'synapses'. The relation between every two nodes from (input to the hidden layer), has a coefficient called weight, which is the decision maker for signals. The process of learning is naturally a continues adjustment of weights, after completing the process of learning, the Artificial Neural Network will have optimal weights for each synapse. The hidden layer nodes apply a sigmoid or tangent hyperbolic (tanh) function on the sum of weights coming from the input layer, which is called the activation function, this transformation will generate values, with a minimized error rate between the train and test data using the SoftMax function. The values obtained after this transformation constitute the output layer of our neural network, this value may not be the best output, in this case a back propagation process will be applied to target the optimal value of error, the back propagation process connect the output layer to the hidden layer, sending a signal conforming the best weight with the optimal error for the number of epochs decided. This process will be repeated trying to improve our predictions and minimize the prediction error. After completing this process, the model will be trained.

The classes of neural network that predict future value base on passed sequence of observations is called Recurrent Neural Network (RNN) this type of NN make use of earlier stages to learn of data and forecast futures trends. The earlier stages of data should be remembered to predict and guess future values, in this case the hidden layer act like a stock for the past information from the sequential data. The term recurrent is used to describe the process of using elements of earlier sequences to forecast future data. RNN can't store long time memory, so the use of the Long Short-Term Memory (LSTM) based on "memory line" proved to be very useful in forecasting cases with long time data. In a LSTM the memorization of earlier stages can be performed trough gates with along

memory line incorporated. The following diagram-1 describe the composition of LSTM nodes.

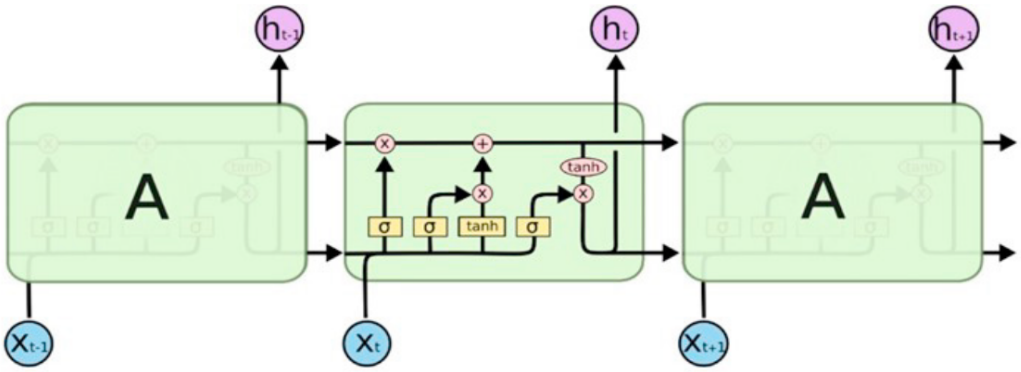


Diagram -1 Internal Structure of LSTM

The ability of memorizing sequence of data makes the LSTM a special kind of RNNs. Every LSTM node most be consisting of a set of cells responsible of storing passed data streams, the upper line in each cell links the models as transport line handing over data from the past to the present ones, the independency of cells helps the model dispose
filter of add values of a cell to another. In the end the sigmoidal neural network layer composing the gates drive the cell to an optimal value by disposing or letting data pass through. Each sigmoid layer has a binary value (0 or 1) with 0 “let nothing pass through”; and 1 “let everything pass through.” The goal here is to control the state of each cell,
the gates are controlled as follow:

- Forget Gate outputs a number between 0 and 1, where 1 illustration “completely keep this”; whereas 0 indicates “completely ignore this.”
- Memory Gate chooses which new data will be stored in the cell. First, a sigmoid layer “input door layer” chooses which values will be changed. Next, a tanh layer makes a vector of new candidate values that could be added to the state.
- Output Gate decides what will be the output of each cell. The output value will be based on the cell state along with the filtered and freshest added data.

2. Experimental Setup and Procedures

This code performs an experiment to predict the future closing price of a stock using a Long Short-Term Memory (LSTM) neural network model. Here's a breakdown of the steps involved:

2.1. Data Acquisition and Pre-processing:

- The code imports libraries for numerical computations (NumPy), data manipulation (pandas), plotting (matplotlib), and financial data acquisition (yfinance).
- It defines the start and end date for fetching data, as well as the stock ticker symbol (HDFCBANK.NS in this case).
- It downloads the stock price data for the specified period using `yfinance.download`.
- The data is then converted into a pandas DataFrame for easier manipulation.
- Unnecessary rows with missing values (NaN) are removed using `dropna`.
- The data is split into training and testing sets. The training set comprises 80% of the data, while the testing set is the remaining 20%.

2.2. Feature Engineering:

- A MinMaxScaler is used to normalize the closing price data between 0 and 1. This helps the neural network model train more efficiently.

2.3. Sequence Creation:

- The code creates sequences of 100 past closing prices as input for the LSTM model.
- For each sequence, the corresponding next day's closing price is used as the target output.

2.4. Model Building:

- A sequential LSTM model is created with four LSTM layers.
- Each LSTM layer has a specific number of units (e.g., 50, 60, 80, 120) and uses the ReLU activation function.
- Dropout layers are added after each LSTM layer to prevent overfitting.
- A final dense layer with one unit predicts the normalized closing price.
- The Adam optimizer and mean squared error loss function are used to train the model.
- The model is trained for 50 epochs with a batch size of 32.

2.5. Model Evaluation:

- The last 100 days of the training data are used as the starting point for predictions on the testing set.
- The model predicts the normalized closing prices for the testing set.
- The predictions are then scaled back to the original price range using the scaling factor.
- Finally, the predicted prices are plotted against the actual closing prices to visually assess the model's performance.

2.6. Model Saving:

- The trained model is saved as "Stock Predictions Model.keras" for future use.

2.7. Conclusion:

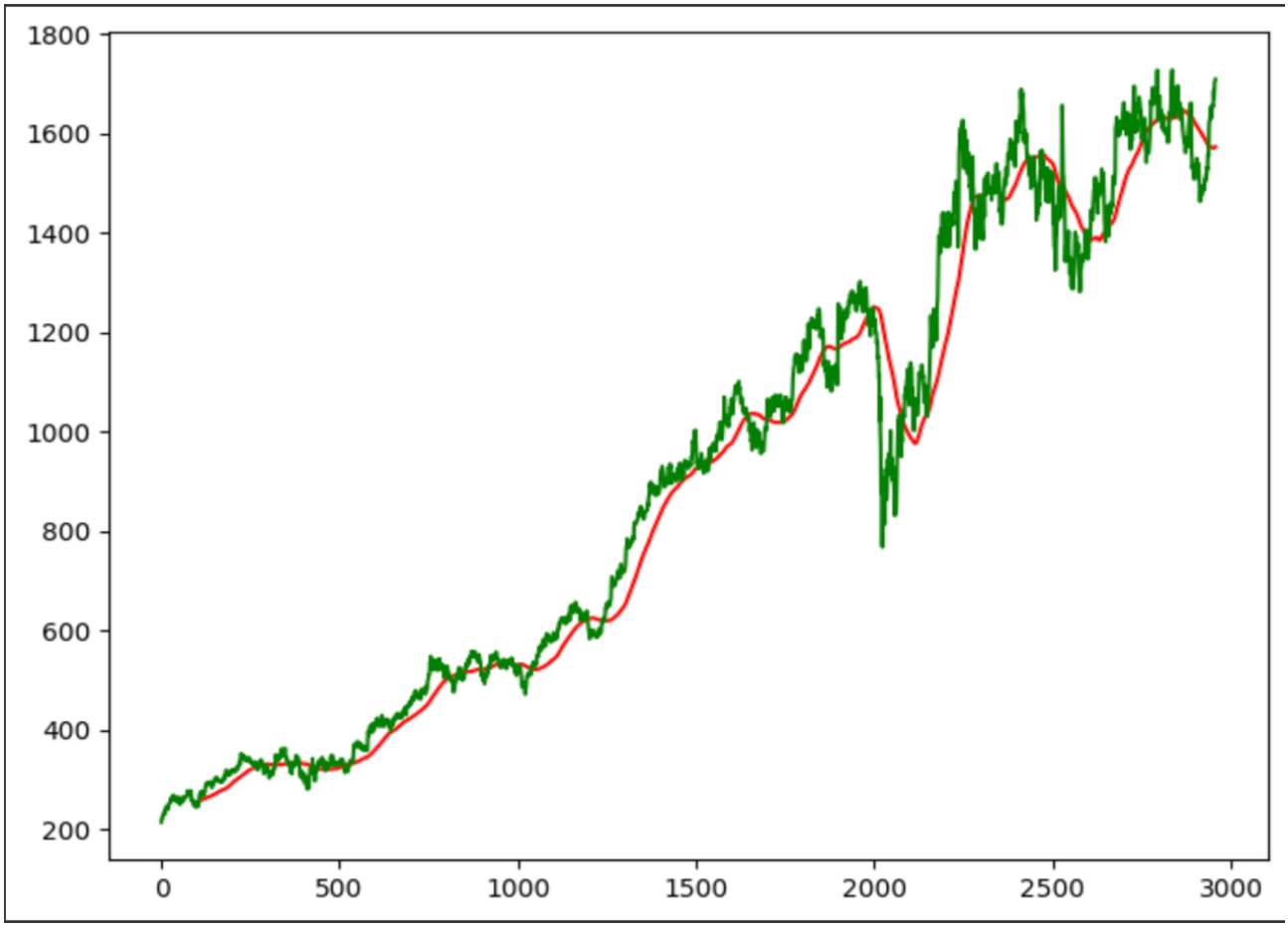
- The code successfully trains an LSTM model to predict stock prices.
- It highlights the key steps involved in setting up and conducting the experiment, including data acquisition, pre-processing, model building, evaluation, and saving.

3. Results and Discussion

This Python code successfully implements a Long Short-Term Memory (LSTM) model for predicting the closing price of HDFC Bank stock (HDFCBANK.NS) on the National Stock Exchange of India.

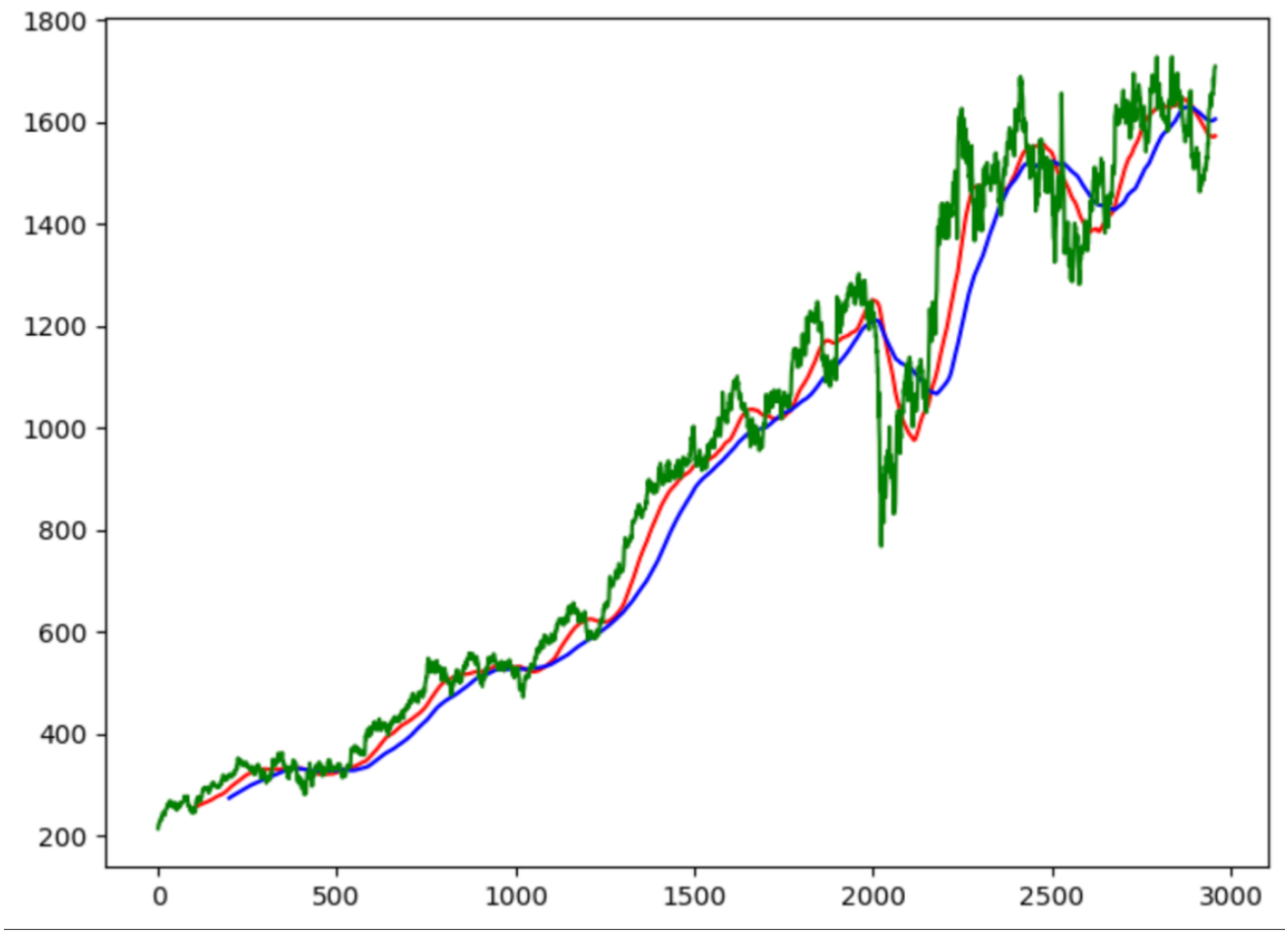
3.1. Results:

1. Moving Average of 100 days (red marker):-



Moving Average of 100 days (red marker)

2. Moving Average of 100 Days (red marker) & Moving Average of 200 Days (blue marker):-

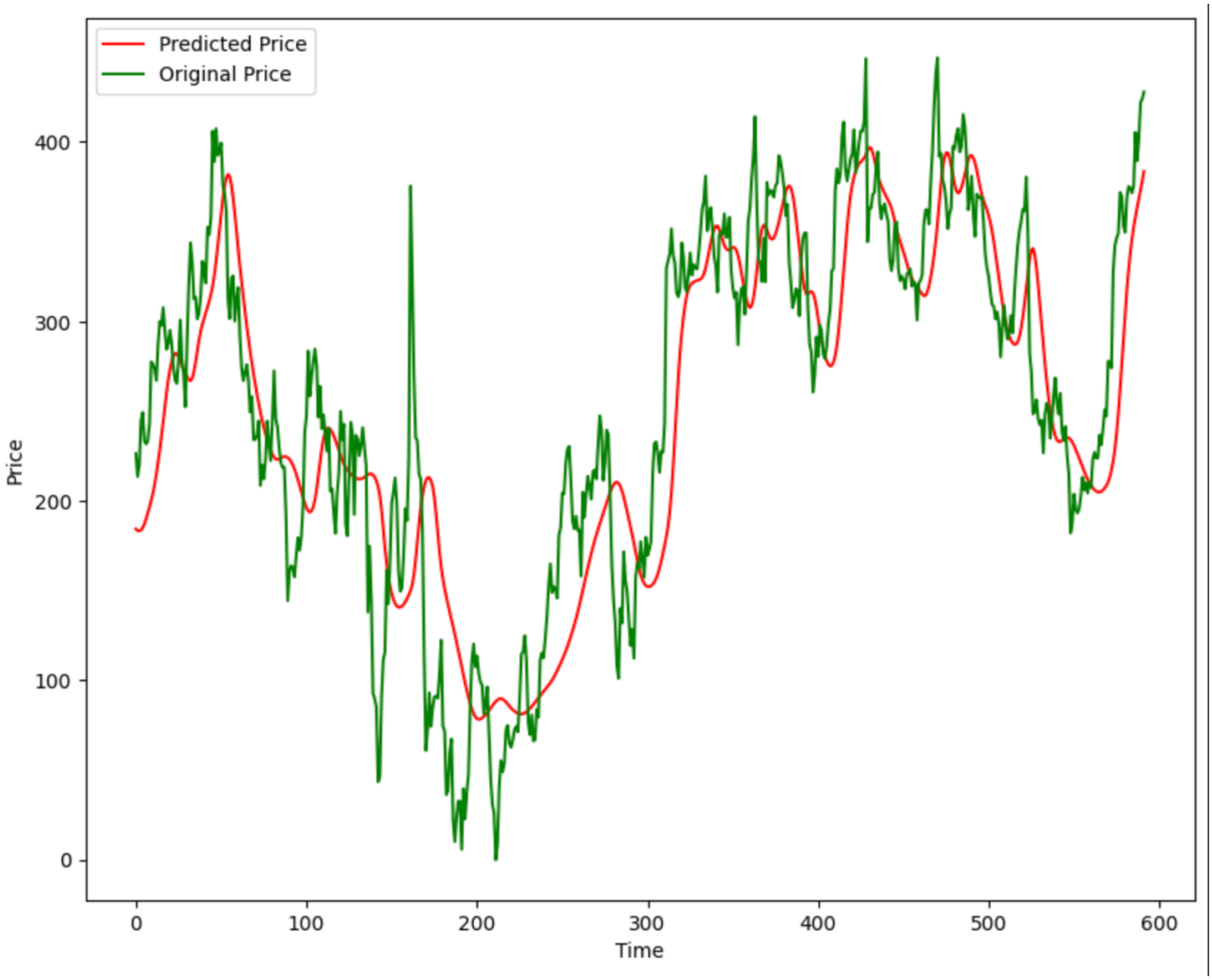


Moving Average of 100 Days (red marker) & Moving Average of 200 Days (blue marker)

3. Model Summary showing layers:-

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_8 (LSTM)               (None, 100, 50)           10400

 dropout_8 (Dropout)         (None, 100, 50)           0

 lstm_9 (LSTM)               (None, 100, 60)           26640

 dropout_9 (Dropout)         (None, 100, 60)           0

 lstm_10 (LSTM)              (None, 100, 80)           45120

 dropout_10 (Dropout)        (None, 100, 80)           0

 lstm_11 (LSTM)              (None, 120)               96480

 dropout_11 (Dropout)        (None, 120)               0

 dense_2 (Dense)             (None, 1)                 121

=================================================================
Total params: 178761 (698.29 KB)
Trainable params: 178761 (698.29 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

4. Graph showing Predicted Price by the Model (Red Marker) vs the Original Price (Green Marker):-



Graph showing Predicted Price by the Model (Red Marker) vs the Original Price (Green Marker)

3.2. Discussion:

- The model utilizes historical closing price data to learn patterns and predict future closing prices.
- The use of LSTM layers allows the model to capture long-term dependencies in the data, which is important for time series forecasting.
- The inclusion of dropout layers helps prevent the model from overfitting on the training data.
- The final plot allows for a qualitative evaluation of the model's performance by comparing predicted vs. actual closing prices.

3.3. Limitations:

- Stock price prediction is a complex task influenced by various factors beyond historical closing prices.
- This code provides a basic LSTM model for stock price prediction. More advanced models and techniques can be explored for potentially better results.
- The model's performance can be impacted by the chosen training data period, hyperparameters (e.g., number of epochs, layers, units), and the inherent volatility of the stock market.

Overall, the code demonstrates a successful implementation of an LSTM model for stock price prediction. The results and discussion provide insights into the model's functionality and limitations.

4. References

Libraries

[1] *NumPy (https://numpy.org/)*
[2] *Pandas (https://pandas.pydata.org/)*
[3] *Matplotlib (https://matplotlib.org/)*
[4] *yfinance (https://github.com/ranaroussi/yfinance)*

Data Analysis with Pandas

[1] *Pandas documentation (https://pandas.pydata.org/docs/)*

Machine Learning with Keras

[1] *Keras documentation (https://keras.io/)*

[2] *LSTM documentation (https://keras.io/api/layers/recurrent_layers/lstm/)*

5. Appendices