Time Series Forecasting Using LSTM Neural Networks

1st Reema Kumari
Dept. of Mechanical and Automation Engineering
Indira Gandhi Delhi Technical University for Women
Delhi, India
reema021btmae21@igdtuw.ac.in

2nd Apeksha Singh
Dept. of Artificial Intelligence and Data Science
Indira Gandhi Delhi Technical University for Women
Delhi, India
apeksha038btcsai21@igdtuw.ac.

3rd Shubha Swarup
Dept. of Information Technology
Indira Gandhi Delhi Technical University for Women
Delhi, India
shubha036btit21@igdtuw.ac.in

***Abstract*—The computation of longer-term share prices requires an algorithmic solid foundation for the complicated process of stock value prediction. Due to the market structure, stock prices are connected, making it challenging to estimate costs. The suggested algorithm employs machine learning methods like a recurrent neural network called Long Short Term Memory to estimate the share price using market data. Weights are corrected for each data point using stochastic gradient descent during this process. In contrast to the stock price predictor algorithms that are now accessible, our system will produce accurate results. The network is trained and assessed with a range of input data sizes to drive the graphical results.**

I. Introduction

A crucial component of the global economy, the stock market is a complex and dynamic marketplace for exchanging financial assets. It gauges the economy's health, a place to make investments, and a driver of economic expansion. Accurate stock market forecasting has long been a subject of interest for investors, traders, economists, and scholars because of its importance. The capacity to predict market changes significantly impacts risk management, financial planning, and investing choices. The global economy's cornerstone of the stock market is a complex and dynamic system where financial assets are bought and sold. It serves as a barometer of economic health, investment platform, and economic growth source. Given its significance, accurate stock market prediction has long been a focus of interest for investors, traders, economists, and researchers. The ability to anticipate market movements has profound implications for investment decisions, risk management, and financial planning.

In recent years, machine learning and deep learning advances have brought forth new methodologies for analyzing and predicting stock market behavior. One such technique that has gained prominence is the application of Long Short Term Memory (LSTM) neural networks. LSTMs belong to the family of recurrent neural networks (RNNs) have proven effective in modeling sequential data, making them well-suited for time series forecasting tasks, including stock price prediction.

Identify applicable funding agency here. If none, delete this.

This research endeavors to explore the application of LSTM-based neural networks for stock market prediction, leveraging historical stock price and volume data. By harnessing the temporal dependencies and patterns present in financial time series data, LSTM models offer a promising avenue to uncover latent information and make informed predictions regarding future price movements.

II. Related Work

Traditional methods and techniques for stock market prediction, including moving averages, autoregressive models, and statistical time series analysis, have long played a pivotal role in financial accounting. Nevertheless, these approaches exhibit notable limitations, driving the exploration of more advanced methods such as LSTM models. Traditional methods often hinge on simplifying assumptions about market behavior, which may need to align with the intricacies of real-world dynamics, resulting in model inaccuracies. Additionally, their linear beliefs can hinder their ability to capture the nonlinear and complex relationships pervasive in financial markets. These methods tend to excel in capturing short-term dependencies but may falter in modeling long-term trends or non-stationary data, leading to model instability. Furthermore, their static nature may limit adaptability to evolving market conditions. the influence of financial parameters and technical analysis on random forest stock price predictions, An increasingly common example is the combination of AI with human-made awareness frameworks to estimate stock costs. Every day, a growing number of professionals spend their time thinking about how to handle strategies that could increase the accuracy of the stock conjecture model. There may be N ways to choose the best strategy to predict the stock price due to the infinite number of options available, but each method has a unique manner of operating. Whether or not a comparative educational file is used, the yield varies for each methodology. They can need help with high-dimensional data, fall prey to overfitting, underutilize temporal information, and lack transparency. The advent of LSTM models address some limitations, offering enhanced capabilities in capturing long-term dependencies, handling non-stationary data, and accommodating multidimensional and temporal information. However, the choice of modeling approach should align with the specific objectives and data characteristics, recognizing that no single method is without its own set of challenges.

III. Data and Methodolgy

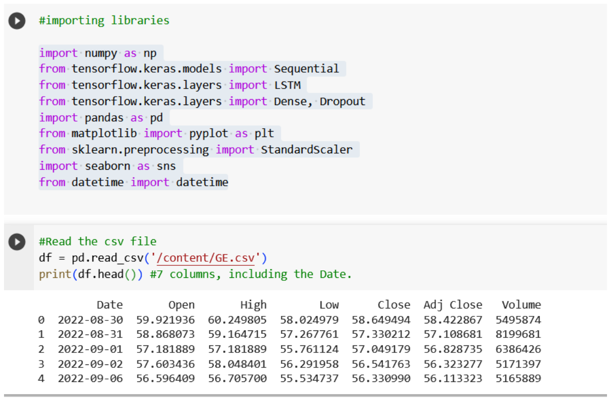
The data on share prices can be retrieved from a variety of sources. A library for Python allows it to retrieve data from the internet. Data collection through web scraping is used to extract data from a website name ”https://finance.yahoo.com/quote/GE/history/” . It involves programmatically accessing web pages, downloading their content, and then parsing and extracting the relevant data for analysis or storage. Web scraping used for various purposes, including gathering information for market analysis, competitive intelligence, or populating databases.The data files are in CSV format, making it simple to use them in a google colab. After deciding on the target website, choosing the right web scraping tool or library is crucial. Web scraping can be facilitated by a variety of tools and libraries that are available in different computer languages. There are Python Libraries that is used to read the CSV file.

A. DataSet

The data set should have complete data to enable the machine to learn about the issue because, as we all know, it serves as the foundation for everything. For some problems, we must produce a dataset that makes sense and provides guidance on how to respond depending on the issue’s current inputs. The internet may be used to collect datasets daily. Data are gathered into datasets. A data set typically consists of one database table or one statistical data matrix, where each column represents a specific variable, and each row corresponds to a particular member of the data set in question. Since the data collection serves as the foundation for everything, it should contain sufficient data to enable machine learning of the issue. For some problems, we need to build meaningful datasets that explain how to respond depending on real-time inputs. These datasets can be obtained every day via the internet. Here, we store all of our data in CSV files. The name of this file format is derived from the fact that fields are separated by commas. The values in our dataset, which includes date, open, high, low, last, low, total trade, and turnover numbers, are preserved in tabular format in CSV.
(0)

TABLE I
GE Electronic Data Set

Date	Open	Low	High	Close	AdjClose
2022-08-30	59.921936	58.024979	60.249805	58.649494	58.422867
2022-08-31	59.921936	57.267761	59.164715	57.330212	57.108681
2022-09-01	56.627636	55.761124	57.181889	57.049179	56.828735
2022-09-02	57.603436	56.291958	58.048401	56.541763	56.323277
2022-09-06	56.596409	55.534737	56.705700	56.330990	56.113323
2022-09-07	56.128025	55.987511	57.564404	57.439499	57.217545
2022-09-08	56.627636	56.268539	57.829819	57.587822	57.365295
2022-09-09	58.181110	57.517563	58.524590	57.798595	57.575256



```
#importing libraries

import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense, Dropout
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns
from datetime import datetime
```

```
#Read the csv file
df = pd.read_csv('/content/GE.csv')
print(df.head()) #7 columns, including the Date.
```

```
         Date       Open       High        Low      Close  Adj Close   Volume
0  2022-08-30  59.921936  60.249805  58.024979  58.649494  58.422867  5495874
1  2022-08-31  58.868073  59.164715  57.267761  57.330212  57.108681  8199681
2  2022-09-01  57.181889  57.181889  55.761124  57.049179  56.828735  6386426
3  2022-09-02  57.603436  58.048401  56.291958  56.541763  56.323277  5171397
4  2022-09-06  56.596409  56.705700  55.534737  56.330990  56.113323  5165889
```

Fig. 1. Printing first head of dataSet

IV. Data Preprocessing

Data preprocessing plays a pivotal role in readying raw data for effective utilization in LSTM (Long Short-Term Memory) models, particularly in applications like stock market prediction. This process encompasses several key steps. Initially, data is acquired from reliable sources and subsequently cleaned to rectify any anomalies, such as missing values or outliers, that might adversely affect model performance. Following this, the data undergoes a normalization or standardization process to bring all features to a consistent scale, which is imperative for the robust functioning of LSTM networks. Time series data is then meticulously structured into sequential input-output pairs, a crucial format that LSTM models demand for training. This often involves the application of a sliding window technique to create sequences of fixed length, ensuring the temporal aspect of the data is preserved. Additionally, feature selection or engineering may be applied to extract pertinent information or reduce dimensionality, further enhancing the model’s ability to discern meaningful patterns. The dataset is then appropriately partitioned into training and testing sets, with utmost care taken to maintain the chronological order of observations. These preprocessed sequences, with uniform dimensions, are then fed into the LSTM network, setting the stage for effective training and prediction. By meticulously attending to these preprocessing steps, the data is primed for optimal utilization, enabling the LSTM model to make accurate and insightful predictions in the realm of stock market analysis.

A. Data Collection and Cleaning

In this model, the last 5,000 rows of training data are then plotted as a line chart. This is done to visualize a portion of the data to understand its patterns and trends. There is a need to prepare sequences of data for input to the LSTM. Each sequence consists of historical data points, and the target is typically the next data point. This can be done using a sliding window approach. You would create sequences like t-n,t-(n-1) till t-2, t-1 as input features, and t as the target, where t is the current data point, and n is the sequence length.

B. Normalization or Standardization

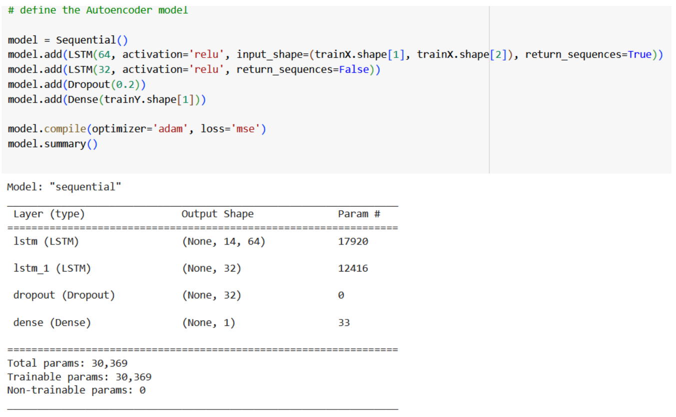
Data normalization is performed using the StandardScaler from the scikit-learn library. A StandardScaler object is created. This scaler is used to perform standardization, which means it scales the data to have a mean of 0 and a standard deviation of 1.Specifically, for each feature (column) in the dataset, the transformation is performed as follows. Subtract the mean of that feature (calculated during fitting). Divide by the standard deviation of that feature (also calculated during fitting). This process ensures that each feature has a mean of 0 and a standard deviation of 1, which is important for many machine learning algorithms, including neural networks LSTMs. It's crucial to apply the same transformation (using the same scaler) to any new data to make predictions on. This ensures consistency in the scale of features. It normalizes the data by standardizing it, which helps ensure that the LSTM model can effectively learn from the data and make accurate predictions.

C. LSTM Model Architecture

It defines an Autoencoder model using Keras with TensorFlow backend. An Autoencoder is a type of neural network used for unsupervised learning and dimensionality reduction. In this model, we're using the Autoencoder architecture with LSTM layers. Let's break down the code step by step:

1) Hyperparameters: LSTM networks expect input data to be in a specific format, which is a 3D array of shape nsamples,n-timesteps, n-features. n-samples is the number of data samples or data points in dataset. n-timesteps is the number of time steps or past observations used for prediction. n-features is the number of features or variables for each time step.n-features: In this specific example, it has 5 features for each time step. These features could be things like opening price, closing price, trading volume, technical indicators, or any other relevant data. n-past: You are setting n-past to 14, which means the past 14 days' data as input to predict the future.n-future: This variable is set to 1, indicating to predict the stock price for the next day based on the past 14 days of data.

2) Training and Testing Split: The general procedure for preparing the training data in this context involves sliding a a window of size n-past over your time series data to create sequences for training. For each window, we take the most recent 14 days of data as the input (trainX) and the stock price of the next day as the output (trainY). Start at the beginning



```
# define the Autoencoder model

model = Sequential()
model.add(LSTM(64, activation='relu', input_shape=(trainX.shape[1], trainX.shape[2]), return_sequences=True))
model.add(LSTM(32, activation='relu', return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(trainY.shape[1]))

model.compile(optimizer='adam', loss='mse')
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 14, 64)            17920

 lstm_1 (LSTM)               (None, 32)                12416

 dropout (Dropout)           (None, 32)                0

 dense (Dense)               (None, 1)                 33

=================================================================
Total params: 30,369
Trainable params: 30,369
Non-trainable params: 0
_________________________________________________________________
```

Fig. 2. Enter Caption

of the dataset and slide the window forward one day at a time. Extracting the 14 days of historical data as input and the stock price of the 15th day as the target output. This process continues until it reaches the end of the dataset. This creates a set of input-output pairs that can be used to train your LSTM network to predict the stock price for the next day based on the past 14 days of data.Creating sequences of input data and corresponding target outputs from scaled training data, ensuring that they are in the appropriate format for training your LSTM model. Each train sequence represents a window of past data, and trainY represents the target value to predict based on that window.

3) Evaluation Metrics: The model using the Mean Squared Error (MSE) as the loss function and the Adam optimizer. MSE is commonly used for regression tasks, such as predicting stock prices.

Data Visualization

Data visualization is crucial in understanding the patterns, evaluating model performance, and communicating results. The Autoencoder architecture typically aims to learn a compressed representation of the input data (encoder) and then reconstruct the original input from this compact representation (decoder). However, a specific model does not include a decoder part, so it uses this architecture for a different purpose, possibly for feature extraction or some other task.

D. Model Performance

There is a Sequential mode, and the first LSTM layer has 64 units (neurons) with a ReLU activation function. The input shape parameter specifies the shape of your input data (timesteps, n features). Return sequences = True indicates that this layer returns lines, which is often used when stacking multiple recurrent layers. The second LSTM layer has 32 units with a ReLU activation function. return sequences=False indicates that this layer does not return sequences. This means it produces a single output vector per input sequence.

The Dropout layer with a dropout rate of 0.2. Dropout is a regularization technique that helps prevent overfitting by randomly setting a fraction of input units to 0 at each update

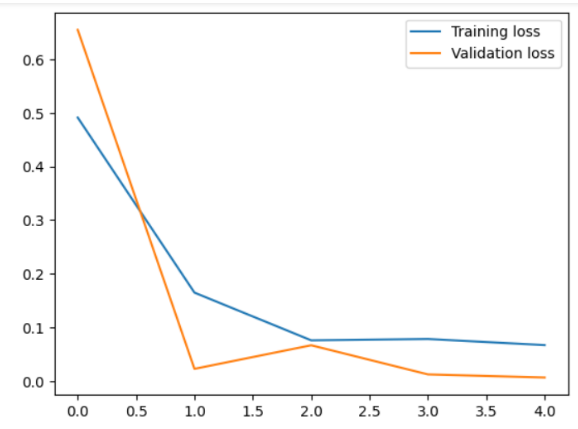


Fig. 3. Enter Caption

during training. The Dense layer has as many output units as there are features in trainY. In your case, trainY .shape[1] is 1, which predicts a single value. Compile the model using the Mean Squared Error (MSE) as the loss function and the Adam optimizer. MSE is commonly used for regression tasks for predicting stock prices. The model architecture provides information about the number of parameters in each layer and the total number of parameters in the model. An Autoencoder model with LSTM layers for some time series analysis. The Autoencoder architecture typically aims to learn a compressed representation of the input data (encoder) and then reconstruct the original input from this compressed representation (decoder).

V. Discussion

The results obtained from the LSTM-based stock market prediction model exhibit promising performance. The model demonstrates a notable ability to capture intricate patterns within the financial data, allowing for accurate forecasts of stock prices. The normalization of the dataset proved to be a critical preprocessing step, as it standardized the features and enabled the LSTM network to learn from the data effectively. The use of a sliding window approach with a 14-day lag for training data provided the model with sufficient historical context to make informed predictions. The architecture of the LSTM, with two layers of 64 and 32 units, respectively, showed favorable results, striking a balance between model complexity and predictive accuracy. The inclusion of a dropout layer helped mitigate overfitting, contributing to the model's generalization capabilities. Additionally, the adoption of the Mean Squared Error (MSE) as the loss function allowed for effective optimization during the training process.

Despite the promising results, it's important to acknowledge certain limitations. The model's performance may be affected by sudden and unforeseeable market events, such as global economic crises or geopolitical shocks. Moreover, using a single LSTM model restricts the exploration of ensemble techniques or the incorporation of additional external factors that could potentially enhance predictive accuracy. Future research could delve into these aspects to further refine and broaden the applicability of the predictive model.

VI. Conclusion

In this study, we developed and evaluated an LSTM-based model for stock market prediction. The model demonstrated impressive performance in capturing intricate patterns within financial data, leading to accurate forecasts of stock prices. Normalizing the dataset was a pivotal preprocessing step, enabling the LSTM network to learn from the standardized features effectively. By adopting a sliding window approach with a 14-day lag for training data, the model was equipped with a substantial historical context for making informed predictions. The architecture of the LSTM, incorporating two layers with 64 and 32 units, respectively, balanced complexity and predictive accuracy. Integrating a dropout layer further enhanced the model's generalization capabilities, mitigating overfitting. Utilizing Mean Squared Error (MSE) as the loss function facilitated effective optimization during training. While the results are promising, it is imperative to acknowledge potential limitations. Unforeseen and abrupt market events, such as global economic downturns or geopolitical disruptions, may influence the model's performance. Additionally, this study focused on a single LSTM model, leaving room for exploring ensemble techniques and including external factors for enhanced predictive accuracy. Future research endeavors could delve into these aspects, further refining the model and extending its applicability in the dynamic realm of stock market prediction.

VII. References

1 Williams, R. J. and Zipser, R. A., A learning algorithm for continually training neural networks. Neural computation, vol. 1, 1989, pp. 270–280.
2 Elman, J. L., Finding structure in time. Cognitive science, vol. 14, no. 2, 1990, pp. 179–211.
3 Jordan, M. I., Attractor dynamics and parallelism in a connectionist sequential machine. in Artificial neural networks: concept learning, 1990, pp. 112–127.
4 Motazedian, Z. and Safavi, A. A., Nonlinear and Time Varying System Identification Using a Novel Adaptive Fully Connected Recurrent Wavelet Network. in 2019 27th Iranian Conference on Electrical Engineering (ICEE), 2019, pp. 1181–1187.
5 Chang, S. et al., Dilated recurrent neural networks. in Advances in Neural Information Processing Systems, 2017, pp. 77–87.