Predicting Google Stock Prices Using Recurrent Neural Networks

Aditya Anant Deshpande
Master of AI and ML a-1910571

***Abstract*—Stock price prediction is a significant and challenging problem in the financial domain. This paper explores the application of Recurrent Neural Networks (RNNs), particularly Vanilla RNNs and Long Short-Term Memory (LSTM) networks, to predict Google stock prices using historical stock data. By leveraging past trends, the models aim to predict the next day's open, high, low, close, and volume prices. The proposed methods were evaluated using a publicly available Kaggle dataset. The results demonstrate the comparative performance of these models and their potential in financial forecasting.**

I. Introduction

Predicting stock prices is a critical task in the field of finance and trading, as accurate forecasts can lead to significant financial gains and risk mitigation. However, the highly volatile, nonlinear, and dynamic nature of stock markets poses considerable challenges to conventional predictive modeling techniques. The prices of stocks are influenced by numerous factors, including economic indicators, investor sentiment, and geopolitical events, making the prediction task inherently complex.

Traditional methods for stock price prediction, such as statistical models, are often unable to capture the intricate temporal dependencies present in the data. In recent years, advancements in machine learning and deep learning have opened new avenues for modeling time-series data, providing tools that can better handle the sequential nature of stock market data.

Recurrent Neural Networks (RNNs) are a class of neural networks specifically designed for sequential data. Unlike traditional feedforward neural networks, RNNs maintain an internal state that enables them to retain information about previous inputs, making them suitable for time-series forecasting tasks. Despite their advantages, Vanilla RNNs are susceptible to issues like vanishing gradients, which hinder their ability to capture long-term dependencies in data.

To address these limitations, Long Short-Term Memory (LSTM) networks were introduced. LSTMs are an extension of RNNs with specialized gating mechanisms that allow them to retain information over extended time steps, making them more robust for tasks involving long-term temporal dependencies.

This paper explores the use of RNNs and LSTMs for predicting Google stock prices. Specifically, the goal is to predict key stock attributes, including the opening price, highest price, lowest price, closing price, and traded volume, using historical data. The contributions of this paper include:

- A systematic approach to preprocessing and normalizing stock price data for deep learning models.
- Implementation and evaluation of Vanilla RNN and LSTM architectures for stock price prediction.
- A detailed analysis of model performance, highlighting their strengths and limitations in capturing temporal patterns.

The remainder of the paper is organized as follows: Section II provides a detailed overview of related work and competing approaches. Section III describes the methodology, including data preprocessing, model architectures, and training strategies. Section IV presents experimental results, followed by conclusions and future work in Section VI.

II. Background

The task of stock price prediction has been extensively studied, with methods ranging from traditional statistical approaches to modern machine learning and deep learning algorithms. This section reviews key approaches and their comparative strengths and limitations.

A. Statistical Methods

Early attempts at stock price prediction relied heavily on statistical models, such as:

- **Autoregressive Integrated Moving Average (ARIMA):** ARIMA models are effective for analyzing and forecasting univariate time series data. They rely on linear assumptions, making them unsuitable for capturing nonlinear relationships in stock prices.
- **GARCH Models:** Generalized Autoregressive Conditional Heteroskedasticity (GARCH) models are used to model volatility clustering in financial data. However, like ARIMA, they struggle with nonlinearities and interdependencies in the data.

B. Machine Learning Methods

The advent of machine learning introduced more sophisticated techniques for stock price prediction, including:

- **Support Vector Machines (SVM):** SVMs can classify stock trends (e.g., price increase or decrease) but are limited in modeling complex sequential dependencies.
- **Random Forests and Gradient Boosting:** These ensemble learning methods are powerful for feature-based prediction but fail to explicitly capture the temporal nature of stock data.

C. Deep Learning Methods

Deep learning has revolutionized the field of time-series prediction by offering methods that can learn hierarchical and sequential patterns directly from data:

- **Feedforward Neural Networks:** Although capable of learning complex patterns, they lack the temporal memory required for sequential data.
- **Recurrent Neural Networks (RNNs):** RNNs introduced the concept of maintaining a hidden state across time steps, making them ideal for time-series tasks. However, Vanilla RNNs suffer from vanishing gradients, limiting their ability to capture long-term dependencies.
- **Long Short-Term Memory (LSTM):** LSTMs address the vanishing gradient problem by introducing gating mechanisms that regulate the flow of information. These networks are capable of modeling long-term dependencies and are widely used for stock price prediction.

D. Comparison of Methods

The strengths and weaknesses of various approaches highlight the trade-offs in selecting a model for stock price prediction. While statistical methods are interpretable and computationally efficient, they lack the flexibility to model complex patterns. Machine learning models offer improved predictive power but require feature engineering and struggle with sequential dependencies. Deep learning models, particularly RNNs and LSTMs, excel in capturing temporal relationships but are computationally intensive and require large datasets for training.

This paper builds on the foundation of deep learning methods, focusing on the application of RNNs and LSTMs for stock price prediction. By comparing the performance of these models, we aim to provide insights into their suitability for financial forecasting tasks.

III. Methodology

A. Dataset Overview

The dataset consists of stock market data with features including *Open*, *High*, *Low*, *Close*, and *Volume*. Two subsets of the data were used: a training dataset spanning the years 2012 to 2016 and a test dataset containing data from the year 2017.

1) Training Dataset: The training data contains 1,258 observations. A summary of its descriptive statistics is presented below:

- **Open**: Mean value of 533.71 with a standard deviation of 151.90. Minimum and maximum values are 279.12 and 816.68, respectively.
- **High**: Mean value of 537.88 with a standard deviation of 153.01. Minimum and maximum values are 281.21 and 816.68, respectively.
- **Low**: Mean value of 529.01 with a standard deviation of 150.55. Minimum and maximum values are 277.22 and 805.14, respectively.
- **Close**: Range varies from 620.76 to 663.59 for the first 10 entries.
- **Volume**: Recorded in units of thousands, with the first 10 entries ranging from 3,764,400 to 11,688,800.

```
train_data.head(10)
✓ 0.0s
```

	Date	Open	High	Low	Close	Volume
0	1/3/2012	325.25	332.83	324.97	663.59	7,380,500
1	1/4/2012	331.27	333.87	329.08	666.45	5,749,400
2	1/5/2012	329.83	330.75	326.89	657.21	6,590,300
3	1/6/2012	328.34	328.77	323.68	648.24	5,405,900
4	1/9/2012	322.04	322.29	309.46	620.76	11,688,800
5	1/10/2012	313.70	315.72	307.30	621.43	8,824,000
6	1/11/2012	310.59	313.52	309.40	624.25	4,817,800
7	1/12/2012	314.43	315.26	312.08	627.92	3,764,400
8	1/13/2012	311.96	312.30	309.37	623.28	4,631,800
9	1/17/2012	314.81	314.81	311.67	626.86	3,832,800

```
train_data.describe()
✓ 0.0s
```

	Open	High	Low
count	1258.000000	1258.000000	1258.000000
mean	533.709833	537.880223	529.007409
std	151.904442	153.008811	150.552807
min	279.120000	281.210000	277.220000
25%	404.115000	406.765000	401.765000
50%	537.470000	540.750000	532.990000
75%	654.922500	662.587500	644.800000
max	816.680000	816.680000	805.140000

Fig. 1. Training dataset preview and descriptive statistics.

2) Test Dataset: The test data comprises 20 observations. A summary of its descriptive statistics is presented below:

- **Open**: Mean value of 807.53 with a standard deviation of 15.13. Minimum and maximum values are 778.81 and 837.81, respectively.
- **High**: Mean value of 811.93 with a standard deviation of 14.38. Minimum and maximum values are 789.63 and 841.95, respectively.
- **Low**: Mean value of 801.95 with a standard deviation of 13.28. Minimum and maximum values are 775.80 and 827.01, respectively.
- **Close**: Range varies from 786.14 to 835.67 for the first 10 entries.
- **Volume**: Recorded in units of thousands, with the first 10 entries ranging from 1,073,000 to 1,657,300.

These descriptive statistics highlight the characteristics of the data and form the basis for further analysis and model training.

B. Data Preprocessing

The dataset consists of daily stock prices, including open, high, low, close, and volume values. The preprocessing steps

```
test_data.head(10)
```
✓ 0.0s

	Date	Open	High	Low	Close	Volume
0	1/3/2017	778.81	789.63	775.80	786.14	1,657,300
1	1/4/2017	788.36	791.34	783.16	786.90	1,073,000
2	1/5/2017	786.08	794.48	785.02	794.02	1,335,200
3	1/6/2017	795.26	807.90	792.20	806.15	1,640,200
4	1/9/2017	806.40	809.97	802.83	806.65	1,272,400
5	1/10/2017	807.86	809.13	803.51	804.79	1,176,800
6	1/11/2017	805.00	808.15	801.37	807.91	1,065,900
7	1/12/2017	807.14	807.39	799.17	806.36	1,353,100
8	1/13/2017	807.48	811.22	806.69	807.88	1,099,200
9	1/17/2017	807.08	807.14	800.37	804.61	1,362,100

```
test_data.describe()
```
✓ 0.0s

	Open	High	Low	Close
count	20.000000	20.000000	20.000000	20.000000
mean	807.526000	811.926500	801.949500	807.904500
std	15.125428	14.381198	13.278607	13.210088
min	778.810000	789.630000	775.800000	786.140000
25%	802.965000	806.735000	797.427500	802.282500
50%	806.995000	808.640000	801.530000	806.110000
75%	809.560000	817.097500	804.477500	810.760000
max	837.810000	841.950000	827.010000	835.670000

Fig. 2. Test dataset preview and descriptive statistics.

included:

- **Cleaning:** Converting strings to float and handling missing values.
- **Normalization:** Scaling values to a range of [0, 1] using MinMaxScaler.
- **Sequence Creation:** Splitting the data into sequences of $N = 30$ days to predict the next day's prices ($M = 1$).

C. Model Architectures

Vanilla RNN: Implemented with 50 units and a dense output layer to predict all five features for the next day. The `tanh` activation function was used to capture non-linearities.

LSTM: Included 50 units with forget gates to capture long-term dependencies. The output layer mirrored the structure of the RNN for consistency.

D. Training and Validation

The dataset was split into training (80%) and validation (20%) sets. Models were trained for 20 epochs using the Adam optimizer with a learning rate of 0.001.

IV. Experimental Analysis

A. Evaluation Metrics

The models were evaluated using Mean Squared Error (MSE) and Mean Absolute Error (MAE).

B. Results

TABLE I
Performance Comparison of Models

Model	MSE	MAE
Vanilla RNN	0.001255	0.028243
LSTM	0.0008150	0.023237

C. Observations

- LSTM outperformed Vanilla RNN in both MSE and MAE, indicating better handling of temporal dependencies.
- Both models effectively predicted the overall trend but showed some deviation in capturing sudden spikes.

V. Code

The implementation is publicly available on GitHub: https://github.com/aadi654/DLF_Assignments/tree/main/Assignment%203. The repository includes:

- Data preprocessing scripts.
- Model training and evaluation code.
- Visualization of results.

VI. Conclusion

This study demonstrates the application of RNNs and LSTMs for Google stock price prediction. While LSTMs proved more effective in capturing temporal patterns, both models highlighted the potential of neural networks in financial forecasting. Future work can explore:

- Incorporating additional features (e.g., macroeconomic indicators).
- Testing on multi-step predictions.
- Optimizing hyperparameters for improved performance.

References

1) Kaggle dataset: https://www.kaggle.com/rahulsah06/google-stock-price.