RNN Project

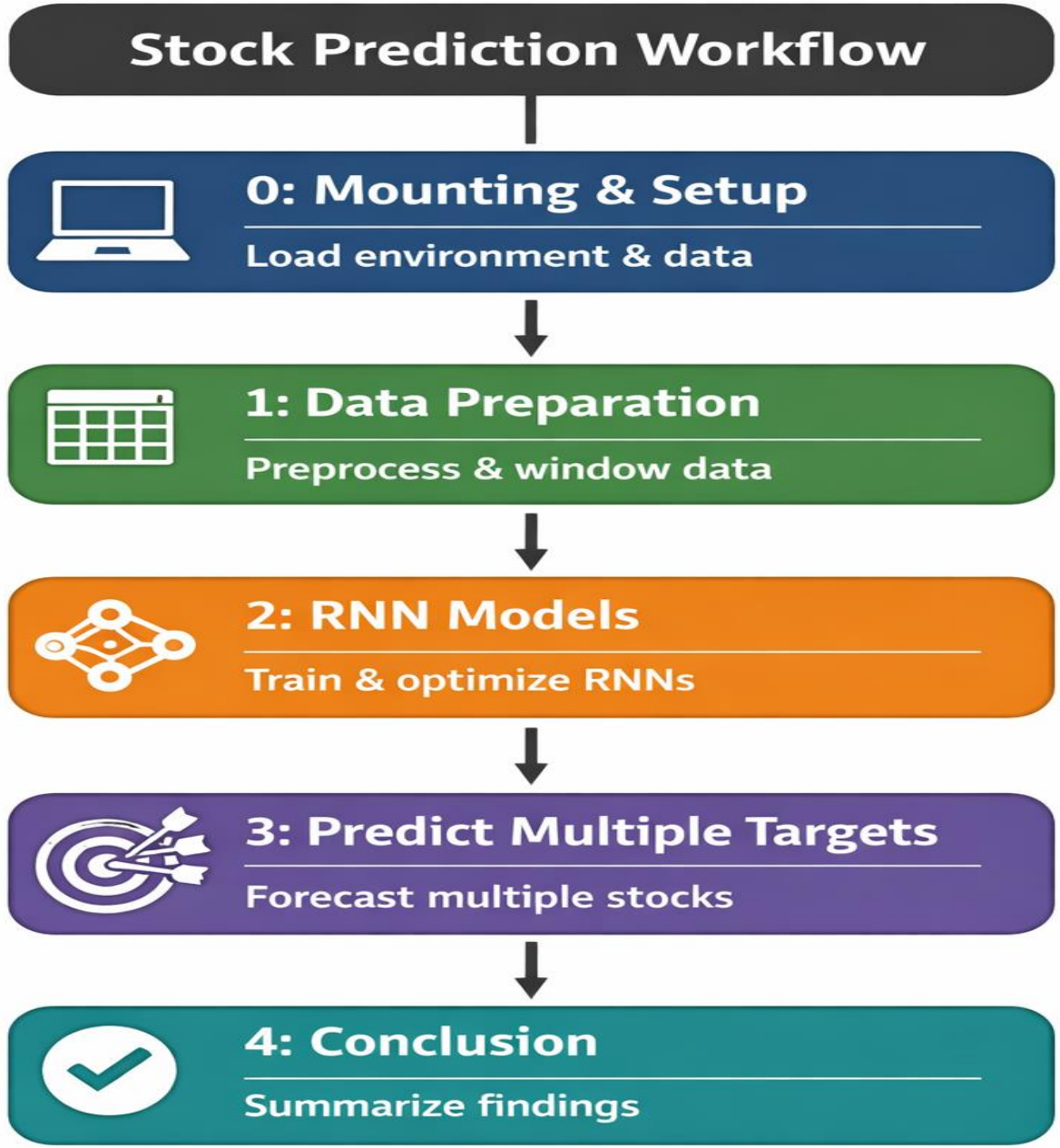
Stock Prices prediction

Introduction

- The objective of this assignment is to try and predict the stock prices using historical data from four companies IBM (IBM), Google (GOOGL), Amazon (AMZN), and Microsoft (MSFT).

Technology used

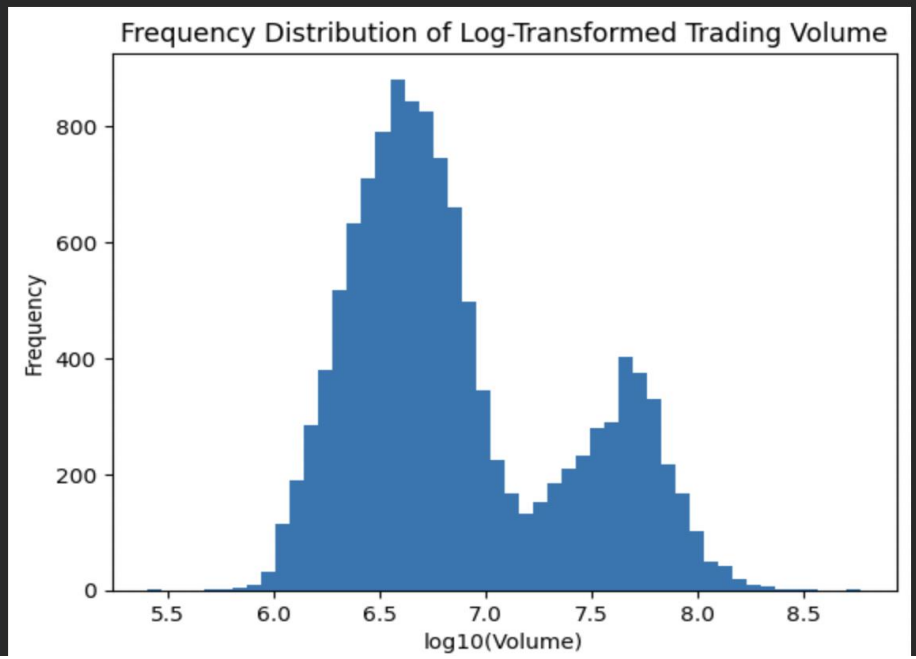
- Programming Language: Python 3.x
- Libraries & Frameworks:
  - `pandas` → data loading & manipulation
  - `numpy` → numerical computations
  - `matplotlib` & `seaborn` → data visualization
  - `scikit-learn` → data scaling and preprocessing
  - `tensorflow.keras` → building and training RNN, GRU, LSTM models
- Environment: Google Colab
- Data Sources: Historical stock data CSVs (AMZN, GOOGL, IBM, MSFT)
- Machine Learning Concepts:
  - Recurrent Neural Networks (Simple RNN, GRU, LSTM)
  - Windowing & sequence modeling for time series
  - Hyperparameter tuning & early stopping


Stock Prediction Workflow
0: Mounting & Setup
Load environment & data
1: Data Preparation
Preprocess & window data
2: RNN Models
Train & optimize RNNs
3: Predict Multiple Targets
Forecast multiple stocks
4: Conclusion
Summarize findings

Data Loading and Preparation

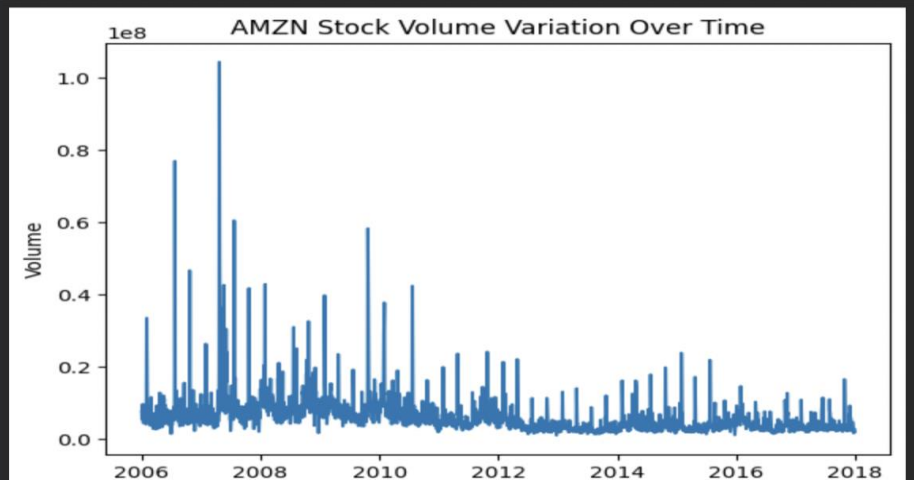
- The dataset was stored on Google Drive and mounted into the Google Colab environment to ensure consistent file paths and avoid runtime file access errors.
- Standard Python libraries such as NumPy and Pandas were used for data handling, while TensorFlow Keras was used to implement an LSTM-based recurrent neural network. why LSTM and not Vanilla RNN: to avoid vanishing gradients.
- We created a helper function to standardize loading and aggregation across multiple companies.
- A custom function was implemented to load and aggregate stock data from multiple companies, enabling a unified representation while preserving company-specific information.
- The aggregated dataset was inspected for structure, data types, and missing values to ensure consistency before preprocessing
- The data was sorted by company and date to preserve the temporal order required for recurrent neural networks.

```
plt.figure()
plt.hist(np.log10(stock_df['Volume']), bins=50)
plt.title("Frequency Distribution of Log-Transformed Trading Volume")
plt.xlabel("log10(Volume)")
plt.ylabel("Frequency")
plt.show()
```

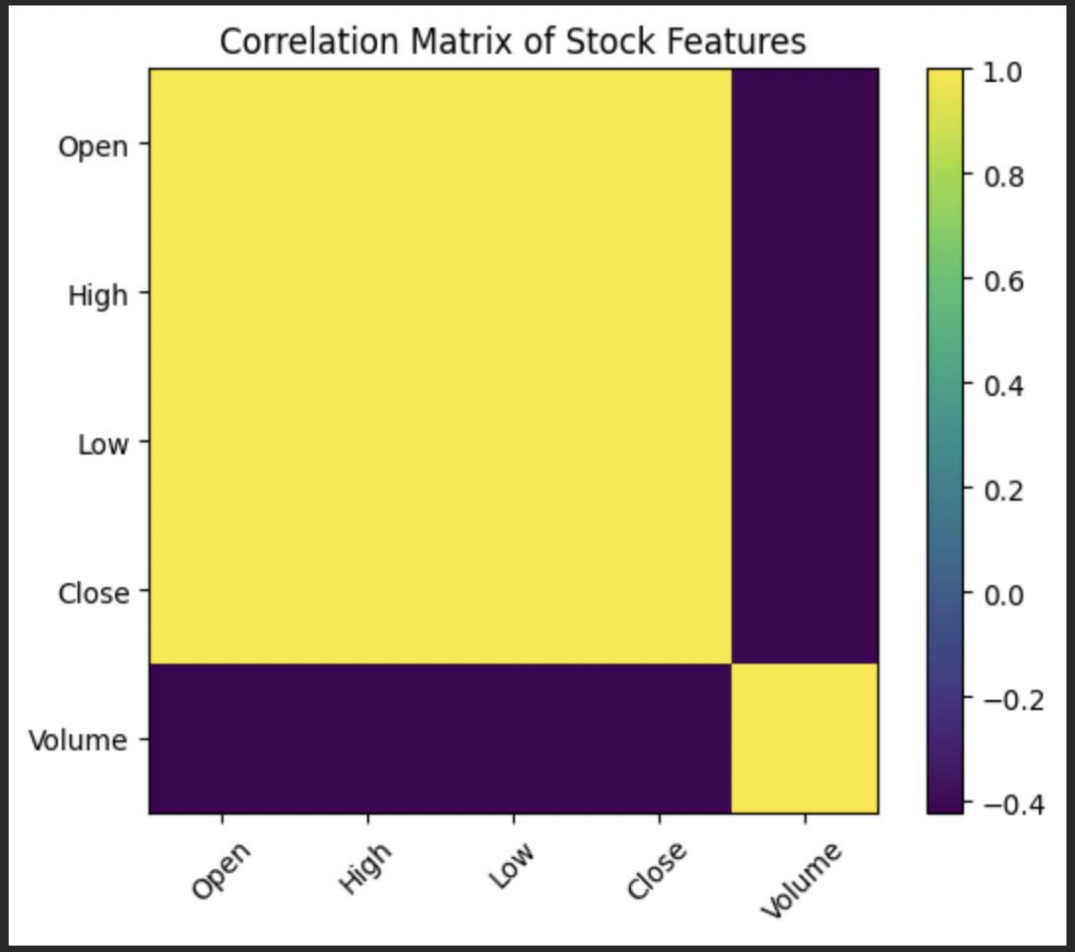


This observation motivates the use of normalization techniques before training the RNN.

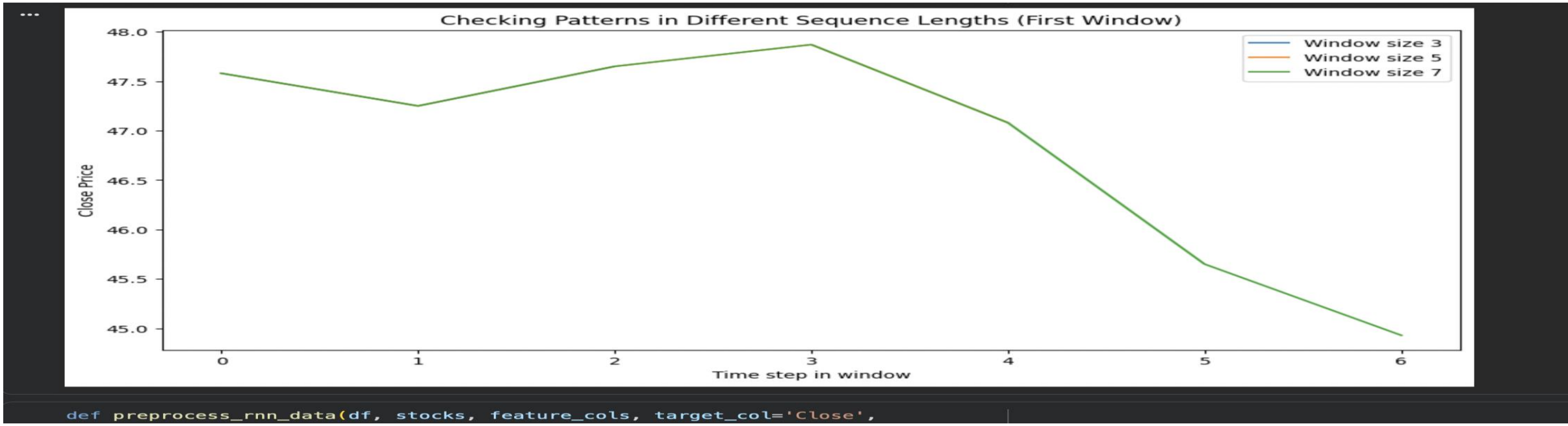
```python
# Stock volume variation over time

amzn_df = stock_df[stock_df['Company'] == 'AMZN']

plt.figure()
plt.plot(amzn_df['Date'], amzn_df['Volume'])
plt.title("AMZN Stock Volume Variation Over Time")
plt.xlabel("Date")
plt.ylabel("Volume")
plt.show()
```



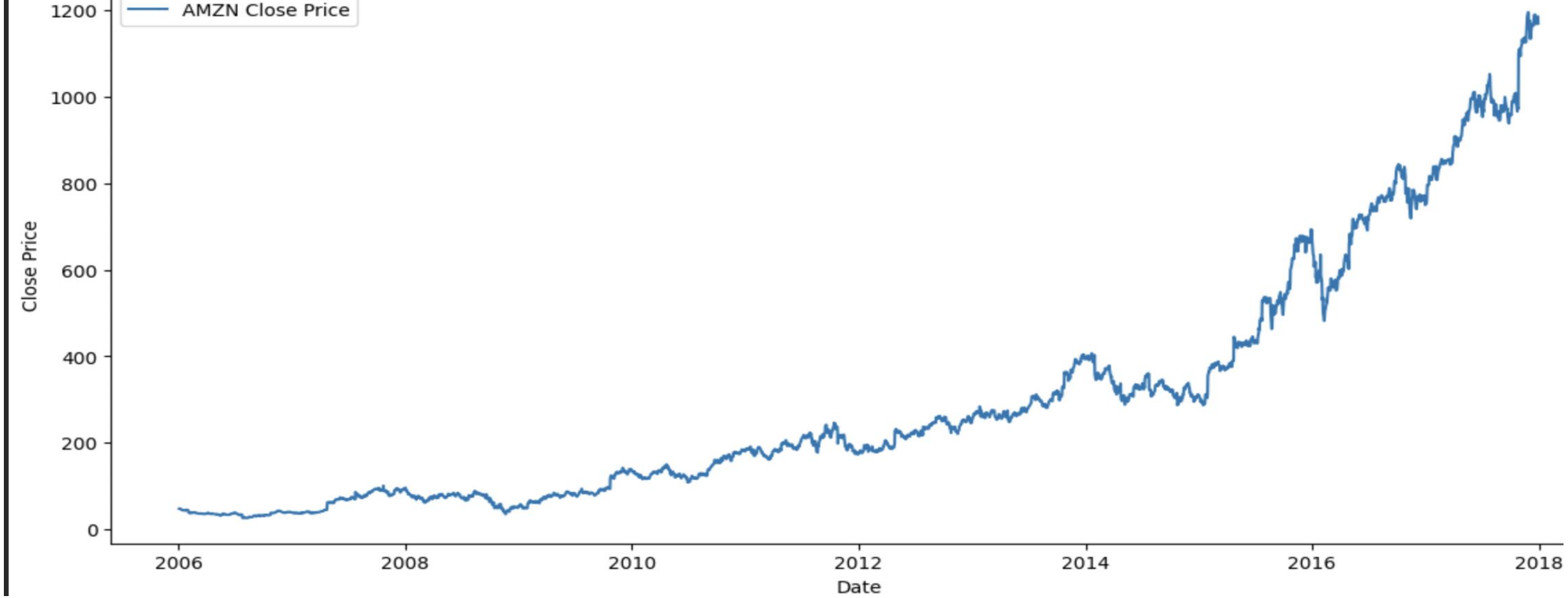
The variation of trading volume over time shows significant fluctuations, with sharp spikes corresponding to periods of increased market activity. This further highlights the sequential and time-dependent nature of stock market data



The correlation analysis shows strong relationships among price-related features such as Open, High, Low, and Close, while Volume exhibits weaker correlation. Despite this, all features were retained to allow the RNN to learn complex temporal dependencies.



- Different sequence lengths (window sizes) were analyzed to examine the temporal patterns in stock prices. This helped inform the selection of an appropriate window size for the RNN input sequences



Different sequence lengths were analyzed using sliding windows to examine how the Close price varies over time. This informed the choice of an appropriate window size for the RNN input sequences.

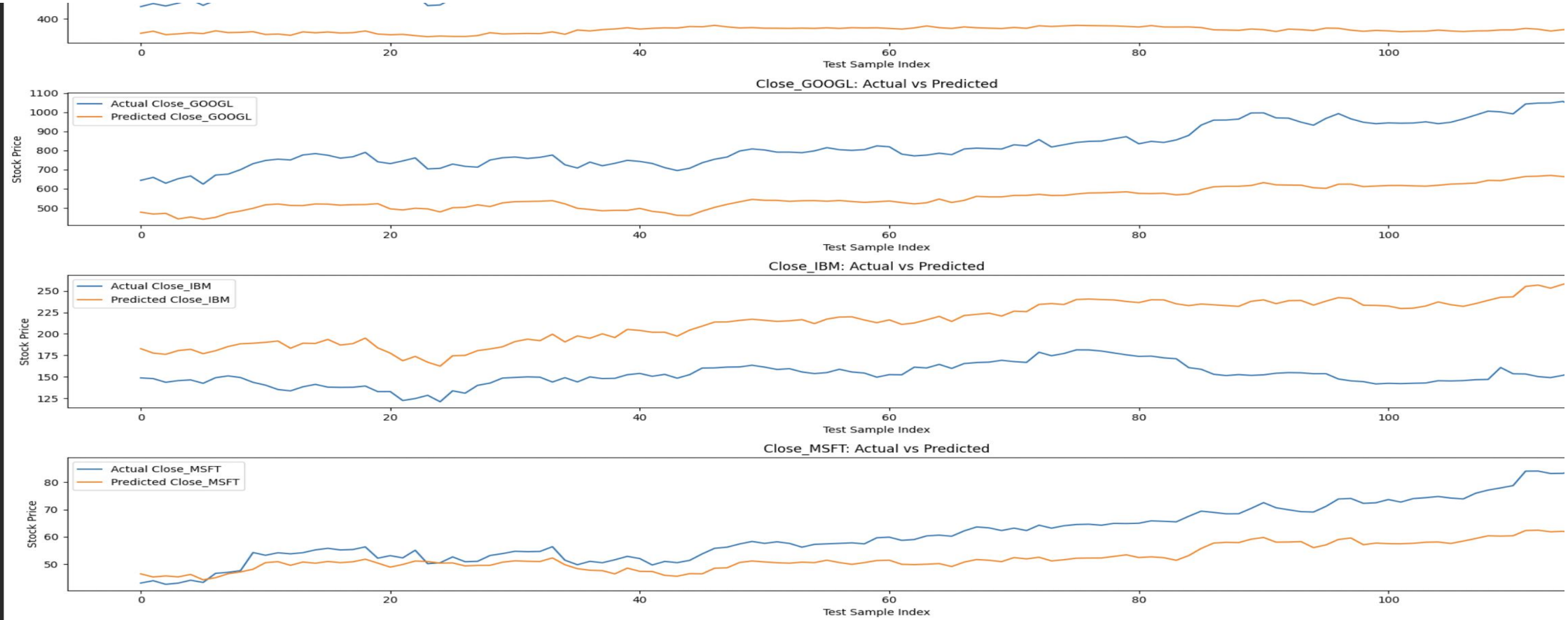
RNN Models

- Defined a function that creates a simple RNN
- Tuned the RNN for different hyperparameter values
- View the performance of the optimal model on the test data

```
76/76 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step
```

It is worth noting that every training session for a neural network is unique. So, the results may vary slightly each time you retrain the model.

3 Predicting Multiple Target Variables



Conclusion

- Sequential modeling works well for stock data: Recurrent Neural Networks (RNNs) effectively capture temporal patterns in historical stock prices.
- Multi-stock modeling captures market correlations: Using data from multiple technology companies (AMZN, GOOGL, IBM, MSFT) improves prediction by leveraging broader market trends.
- Advanced RNNs outperform simple RNNs: GRU/LSTM networks handle long-term dependencies better, producing lower validation and test errors than simple RNNs.
- Data preprocessing is critical: Proper windowing, scaling, and handling missing values prevent NaNs and ensure stable model training.
- Predictions are reasonably accurate but not perfect: While the models can track trends and approximate stock movements, stock markets remain inherently volatile and influenced by external factors beyond historical prices.