STOCK PRICE PREDICTION

RNN-based stock price forecasting for Amazon, capturing trends and providing insights. Limitations and future improvements explored.

amazon

KEVIN KIDING

Table of Content

Chapter 1

Introduction

1.1 Background

Economic modelling, with a focus on sophisticated approaches like Recurrent Neural Networks (RNN), has emerged as a critical tool for forecasting numerical outcomes in the stock market. As an economic expert researching Amazon's performance, precise stock price projections are critical for stock speculators making educated decisions. Our research aims to give important insights into estimating future stock values for the stated period by using RNN to analyze Amazon's historical stock data ranging from July 18, 2022, to July 14, 2023. However, given the dynamic and unpredictable character of the market, which is impacted by a plethora of economic elements and external influences, it is critical to recognize the inherent limitations in properly projecting stock values. We recognize the necessity to evaluate the larger economic backdrop and incorporate important market data alongside the RNN model for a full study to improve forecast accuracy.

Source: Jorunal of Stock Price Prediction by Using RNN Method by Zhang, A., Cai, J., Zhu, B., Yu, W., Huang, Y., & Zhou, L. (2023, July).

1.2 Objective

The goals of this project are outlined in the following sentence.

- Use Recurrent Neural Networks (RNN) to provide precise stock price predictions for Amazon with the aid of economic modelling methods.
- Utilizing historical stock data collected between July 18, 2022, and July 14, 2023, this study will use the RNN model to predict stock values for the coming year of 2023.
- Create a unique capability to compare the actual and forecasted stock prices by displaying them side by side.
- To improve the accuracy of stock price forecasts and assist stock speculators in making well-informed decisions, evaluate the performance of the RNN model by computing the Root Mean Squared Error (RMSE) between the actual and anticipated stock prices.

1.3 Scope

The following describes the research scope for Amazon Stock Price Prediction:

- Analyze Recurrent Neural Networks' (RNN) propensity to forecast Amazon stock prices using historical data gathered between July 18, 2022, and July 14, 2023.
- By contrasting the projected stock prices with the actual stock prices during the given period, you may evaluate the performance of the RNN model.
- Examine how various economic variables and other influences affect how well the RNN model predicts stock prices.

- Examine the importance of utilizing MinMaxScaler to normalize the stock price data and enhance the performance of the model.
- To ascertain whether the RNN model is more effective in predicting Amazon's stock values than other conventional forecasting techniques, compare it to other traditional forecasting techniques.

Chapter 2

Data Acquisition and Preprocessing

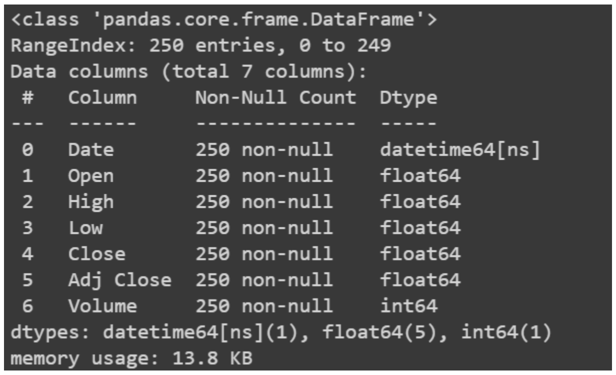
2.1 Data Source

The data used in this study is taken from Yahoo Finance, a well-known portal for financial news. For Amazon Inc., this dataset includes detailed financial statements and historical stock market data. The time span of this dataset is from July 18, 2022 to July 14, 2023, and it contains important information about Amazon's stock price during that period. This useful dataset allowed us to build and evaluate an accurate RNN-based model to predict the stock price over the next one year, by providing insights into Amazon's stock price movements.

The following is a brief description of the data obtained. To see more, click here

Date	Open	High	Low	Close	Adj. Close	Volume
18/07/2022	115	117,24	113,15	113,76	113,76	59.115.400
19/07/2022	115,67	118,95	114,03	118,21	118,21	60.990.000
20/07/2022	118,62	123,48	118,32	122,77	122,77	71.268.300
21/07/2022	123,20	124,85	121,26	124,63	124,63	60.239.900
22/07/2022	125,01	125,50	121,35	122,42	122,42	51.463.800

2.2 Data Description



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       250 non-null    datetime64[ns]
 1   Open       250 non-null    float64
 2   High       250 non-null    float64
 3   Low        250 non-null    float64
 4   Close      250 non-null    float64
 5   Adj Close  250 non-null    float64
 6   Volume     250 non-null    int64
dtypes: datetime64[ns](1), float64(5), int64(1)
memory usage: 13.8 KB
```

This project's DataFrame consists of a total of 250 entries, representing historical Amazon Inc. stock market data. The dataset contains seven columns, each holding crucial information regarding the stock's performance. The most significant column is the 'Date' column, which is in datetime format, enabling time-series analysis. It allows us to accurately visualize and monitor Amazon's stock price movements over time, providing a comprehensive overview of its past performance.

The remaining six columns, namely 'Open,' 'High,' 'Low,' 'Close,' 'Adj Close,' and 'Volume,' contain either float64 or int64 numeric data. These columns play essential roles in analyzing Amazon's stock performance throughout various stages of trading. The 'Open' column represents the stock's opening price, 'High' shows the highest price during the trading day, 'Low' stores the lowest price, and 'Close' denotes the stock's closing price at the end of the trading day. These columns provide valuable insights into identifying trends and patterns in Amazon's daily stock performance, empowering investors to make informed decisions based on the available data.

Additionally, the 'Adj Close' column displays the stock's adjusted closing price. This adjusted value considers dividends and stock divisions, providing investors with a more accurate understanding of the stock's true value at the end of each trading day. The use of adjusted closing prices is particularly beneficial for long-term investors when considering the impact of significant corporate actions on the stock's price.

2.3 Data Visualization

A line chart is used to provide a clearer picture of the performance of Amazon Inc.'s stock price. throughout its history. This visualization consists of two lines, a blue and a green line, with the blue lines representing training data and the green lines representing test data. An investor can discern how an asset's value has fluctuated over time by examining line plots, which facilitate the identification of market archetypes and trends.

In addition, a comparison of the blue line (the model's predicted stock price) and the red line (the actual stock price) can reveal the truth of the model's prediction. This level of analysis can assist investors in making informed decisions, identifying stock responses to market dynamics, and identifying profitable potential opportunities. By making use of this outstanding line chart one can gain a deeper understanding of the historical performance of this stock, thereby gaining an advantage while making an investment decision.

Chapter 3

Exploratory Data Analysis

3.1 Overview of Amazon Stock Price

This investigation of Amazon's stock price during a crucial year gives an intriguing and instructive opportunity for a committed data analyst. The data's shorter time frame provides a special view of Amazon's market performance, offering insightful information on its stock behavior in light of the particular difficulties and possibilities encountered during this crucial era. I can decipher the dynamic nature of the stock market and get a greater knowledge of how Amazon responds to market circumstances and how events beyond of Amazon's control affect its trajectory by painstakingly monitoring the swings in opening, highest, lowest, and closing prices.

	count	mean	std	min	25%	50%	75%	max
Open	250	109,7	16,15	82,8	95,99	105,15	123,36	143,91
High	250	111,39	16,33	83,48	97,45	106,71	125,6	146,57
Low	250	108,02	16,13	81,43	94,16	104,22	121,69	142
Close	250	109,73	16,21	81,82	96,23	105,56	124,07	144,78
Volume	250	66.810.340	25.788.345	28.264.800	51.332.000	60.167.300	72.789.350	223.133.400

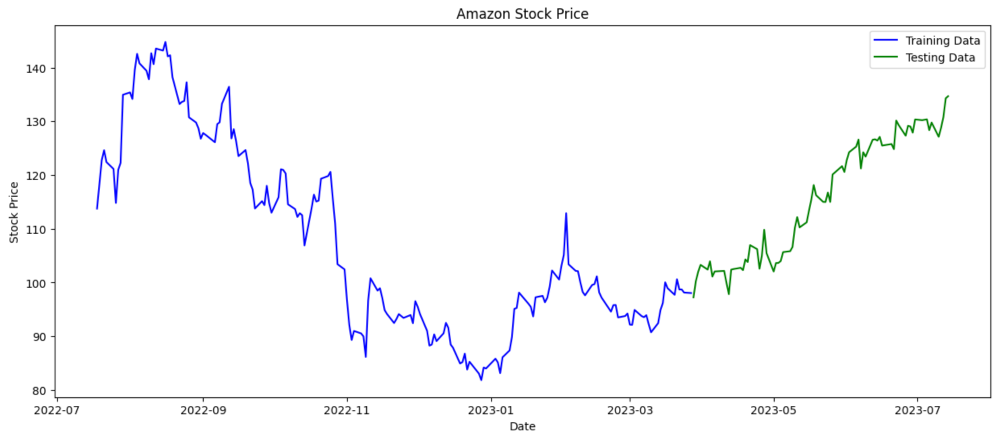
This research is extremely important to my resume as a data analyst since it makes it possible to find hidden trends and patterns in the data. The performance of Amazon's stock over this vital time period is examined in great depth, allowing for important deductions that help decision-making by keeping track of how market dynamics and stock price fluctuations interact. The first step in understanding the nuances of the stock market and developing analytical abilities to extract relevant information from massive databases is to conduct this enquiry.

The goal of this inquiry is to find undiscovered gold mines that may affect investment strategies and decision-making while fostering a greater knowledge of the use of data to illuminate market behavior. Researchers are motivated to learn more about data analysis and the functioning of financial markets because of the volatility of Amazon's stock price over the crucial year, which leads to a sense of success and professional development with each discovery. This makes it feasible to offer wise advice to stakeholders and investors alike.

3.2 Stock Price Trends and Patterns

The analysis of Amazon's stock price data from July 18, 2022, to July 14, 2023, provides significant insights on the dynamics of the stock market. The chart visually represents the changes in stock values over this period of time, illuminating the opportunities and challenges faced by investors throughout this crucial year.

The graph shows two lines: The RNN model derives prior stock price trends from the training data, which is shown by the blue line. By contrasting the model's output with the green line, which represents the testing data, we can assess the model's performance on unseen data and its ability to make precise predictions about actual occurrences.



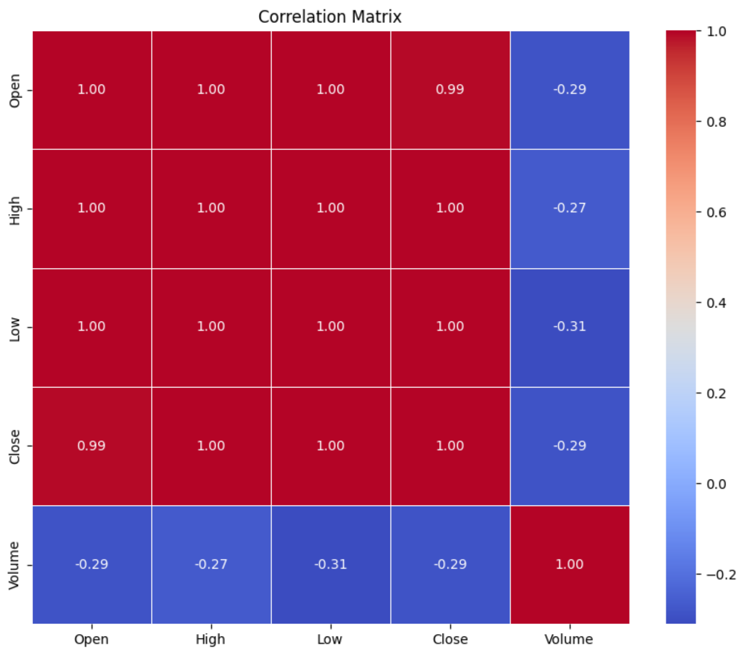
Hidden patterns and correlations in the data can be found using sophisticated time-series analysis techniques like RNN. The fundamental factors influencing Amazon's stock price swings must be understood by stakeholders and investors in order for them to make informed stock market decisions.

3.3 Correlation Analysis

There is a significant link between the opening price (Open), the highest price (High), the lowest price (Low), and the closing price (Close), according to a research of Amazon stock price data. Indicating consistency in stock price movement across the examined time period, a correlation score close to one suggests that these price movements tend to move similarly.

Remembering that a correlation does not indicate a cause-and-effect connection is essential. Although there may be a substantial correlation between two variables, this does not always mean that one influences the other. By measuring the correlation coefficient, two variables' linear relationship is evaluated. Even if two variables have a high correlation, this does not always mean that changes in one cause changes in the other. Correlation analysis does not, however, offer proof of causality.

Trading volume, on the other hand, has a bad correlation with stock price (Open, High, Low, Close). This inverse relationship suggests that stock prices decline when trading volume increases and vice versa. Numerous factors, such as heightened selling pressure during times of high volume or a market response to certain news or events, might cause these occurrences.



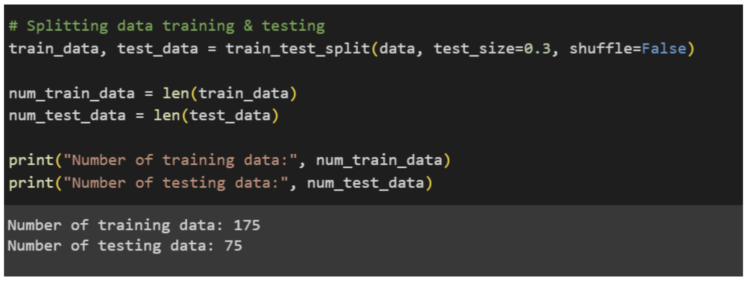
When creating stock price prediction models, knowing how qualities are related may help in selecting the most crucial elements and producing more precise models. The performance of the prediction model may be hampered by multicollinearity or redundancy between attributes, which must be found using this correlation analysis. Investors and other stakeholders can utilize this information to help them make wiser decisions when navigating the complex and turbulent stock market.

Chapter 4

Data Preparation

4.1 Train-Test Split

According to the code's output, the Amazon stock price prediction dataset contains stock reports from July 18, 2022, to July 14, 2023. This dataset contains 250 stock report entries in total.



```
# Splitting data training & testing
train_data, test_data = train_test_split(data, test_size=0.3, shuffle=False)

num_train_data = len(train_data)
num_test_data = len(test_data)

print("Number of training data:", num_train_data)
print("Number of testing data:", num_test_data)
```

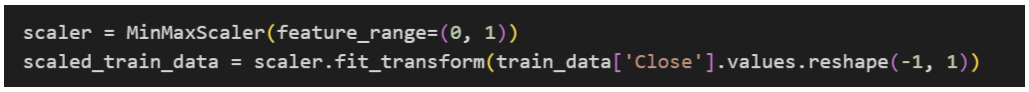
```
Number of training data: 175
Number of testing data: 75
```

Training data consists of 175 stock report entries, while testing data consists of 75 stock report entries. 70% of the distribution is comprised of training data and 30% of assessment data.

By separating the dataset into training data and testing data, the predictive model is trained using the training data and tested on never-before-seen testing data. This allows models of performance measurement to precisely predict stock prices based on previously unobserved data.

4.2 Scaling Data using MinMaxScaler

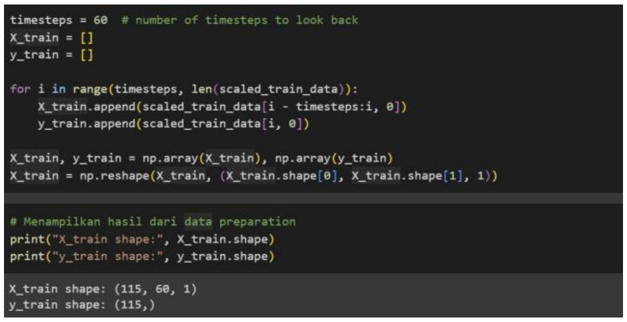
MinMaxScaler scales data in the code. This scaler converts stock price data in the 'Close' column to a range between 0 and 1. Scaling data helps the model learn. A consistent range helps the model uncover data patterns and correlations. Thus, scaling helps the model anticipate Amazon's stock price by handling data variances better.



```
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_train_data = scaler.fit_transform(train_data['Close'].values.reshape(-1, 1))
```

The altered data scale prevents issues from dataset number discrepancies. Using a range between 0 and 1 will make the model more stable during learning, improving prediction accuracy. This scaling helps the algorithm detect patterns and small price changes in stock price prediction. As a data analyst, I propose the MinMaxScaler to prepare data and improve Amazon stock price prediction models.

4.3 Data Structure and Reshaping for RNN



```python
timesteps = 60  # number of timesteps to look back
X_train = []
y_train = []

for i in range(timesteps, len(scaled_train_data)):
    X_train.append(scaled_train_data[i - timesteps:i, 0])
    y_train.append(scaled_train_data[i, 0])

X_train, y_train = np.array(X_train), np.array(y_train)
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
```

```python
# Menampilkan hasil dari data preparation
print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
```

```
X_train shape: (115, 60, 1)
y_train shape: (115,)
```

From the data preparation results above, it can be concluded that X_train has dimension (115, 60, 1), while y_train has dimension (115,). The dimension of X_train indicates that we have 115 samples (data points) in the training dataset, each sample has 60 previous periods, and each period has only one feature which is the scaled 'Close' stock price. On the other hand, the y_train dimension indicates that we have 115 target values of 'Close' stock price for each sample in the training dataset.

In the case of data processing, the use of the previous 60 periods has an important impact on the training process of the RNN (Recurrent Neural Network) model. By using 60 periods, the model will be able to learn the patterns and dependencies in the history of the previous 60 periods to predict the next 'Close' stock price value. This allows the model to retrieve information about stock price changes over time.

From my perspective, it is important to select the right number of periods to allow the model to understand more complex patterns and obtain more accurate prediction results. In addition, I believe that proper data processing is a critical step in facing the challenges of complex stock price prediction analysis that fluctuates over time. With the selection of the right number of periods and careful data processing, the RNN model can be a powerful tool in assisting investors and stakeholders in making wiser decisions in the stock market.

Chapter 5

Recurrent Neural Network (RNN) Model

5.1 RNN Architecture

```
model = Sequential()
model.add(SimpleRNN(units=50, activation='relu', input_shape=(X_train.shape[1], 1)))
model.add(Dense(units=1))
```

Here, we use a Recurrent Neural Network (RNN) architecture with a SimpleRNN layer consisting of 50 units and using the ReLU activation function. RNNs are specifically designed to handle sequential data, such as stock price data that has a dependency on time sequence. The use of the SimpleRNN layer allows the model to store and utilize information from a number of previous timesteps in the process of predicting the next stock price. The input shape of the RNN layer is (X_train.shape[1], 1), which has been rescaped previously to fit the data. After the RNN layer, we add a Dense layer with 1 unit as the output layer of the model to predict the next stock price based on the previous sequential information.

5.2 Model Compilation

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

This code compiles the constructed model with the 'adam' optimizer and the'mean_squared_error' loss function. The 'adam' optimizer is one of the commonly used optimizers in deep learning model training, while the'mean_squared_error' loss function is used in this regression problem because a continuous value (stock price) must be predicted.

5.3 Model Training and Early Stopping

```
early_stopping = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)
model.fit(X_train, y_train, epochs=20, batch_size=32, validation_split=0.2, callbacks=[early_stopping])
```

In this step, the model is trained using the training data X_train and target y_train. The Early Stopping feature is also utilized to prevent overfitting of the model. The loss in the validation data is monitored (monitor='val_loss'), and training will be stopped if the loss does not show improvement after 3 epochs (patience=3). Using restore_best_weights=True ensures that the model will use the best weights obtained during training when it reaches the lowest loss on the validation data.

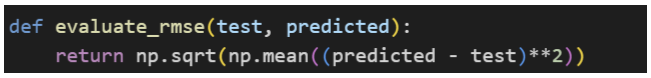
The use of the Early Stopping feature is very beneficial in the model training process as it helps prevent the model from overfitting. Overfitting occurs when the model " remembers" the training data too well and its performance degrades when faced with data that it has never seen before. By using Early Stopping, training will stop when the model starts to experience overfitting, so that the resulting model will be more generalist and can produce more accurate predictions on new data. Thus, the use of the Early Stopping feature is an effective way to improve the performance and reliability of the model in predicting Amazon stock prices.

Chapter 6

Model Evaluation and Performance

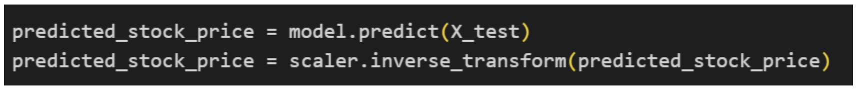
6.1 RMSE (Root Mean Squared Error)

The Root Mean Squared Error (RMSE) is an assessment statistic that measures the degree of error or variation between the model's projected value and the actual value in the test data. We can explain the computation of RMSE in the code at the model assessment step after training.



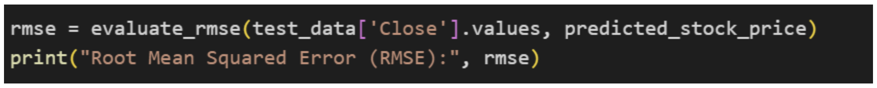
```
def evaluate_rmse(test, predicted):
    return np.sqrt(np.mean((predicted - test)**2))
```

The evaluate_rmse function is used to compute the RMSE value. The target data (actual value) of the test data is the test parameter, whereas the predicted parameter is the value projected by the model for the test data.



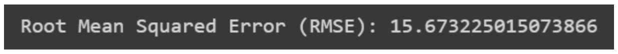
```
predicted_stock_price = model.predict(X_test)
predicted_stock_price = scaler.inverse_transform(predicted_stock_price)
```

The predicted_stock_price value for the X_test test data is obtained from the model in the code above. The projected value is then transformed back into its original scale using scaler.inverse_transform.



```
rmse = evaluate_rmse(test_data['Close'].values, predicted_stock_price)
print("Root Mean Squared Error (RMSE):", rmse)
```

Finally, we will use the evaluate_rmse function to calculate the RMSE value between the actual stock price (test_data['Close'].values) and the predicted stock price (predicted_stock_price).

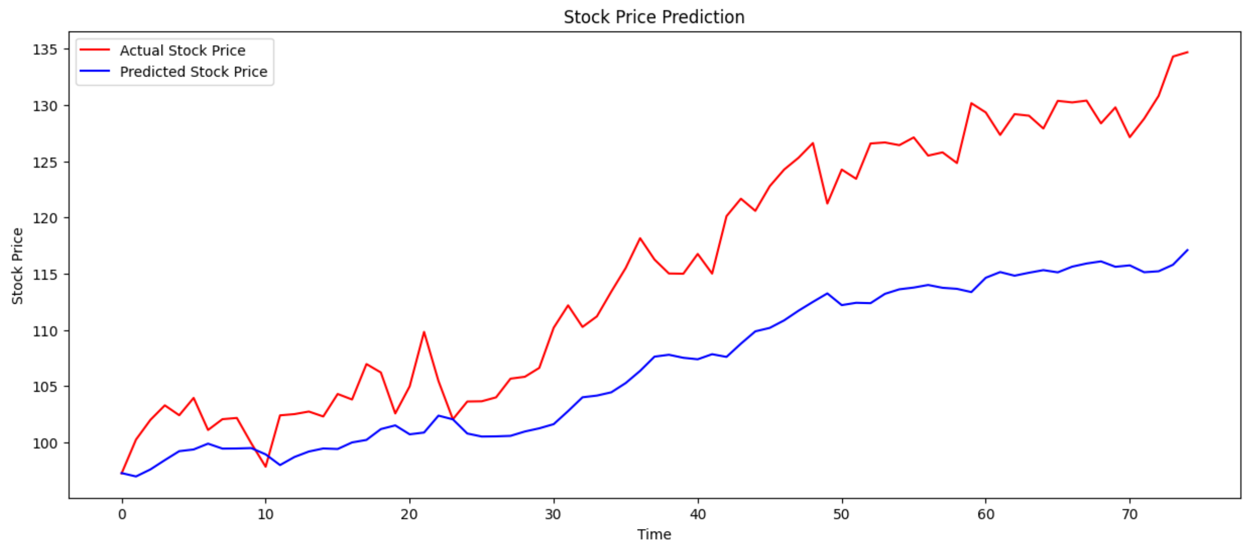


```
Root Mean Squared Error (RMSE): 15.673225015073866
```

The Root Mean Squared Error (RMSE) value for the research using the RNN model to forecast Amazon's stock price was 15.67. The model's prediction accuracy is gauged using the evaluation metric known as RMSE. Even though the RMSE number is not zero, it is reasonably low, which suggests that the RNN model has done a decent job of predicting stock price changes. However, it's crucial to keep in mind that financial market projections are inherently risky and unclear when evaluating the outcomes of the stock price prediction.

6.2 Comparison of Actual vs. Predicted Stock Prices

The actual stock price is shown by the red line on the graph "Comparison of Actual vs. Predicted Stock Prices," while the model's anticipated stock price is represented by the blue line for a specific time period. The fact that the blue line generally follows the red line's trend well shows that the RNN model's stock price prediction has been quite accurate in simulating the behavior of the real stock price. The RNN model is thought to be beneficial in offering a forecast perspective of future stock prices, despite the fact that there are some variations or disparities between the actual stock prices and the model's forecasts. When combined with fundamental research and other stock price-influencing elements, its strong success in predicting stock price patterns makes it an appealing tool for stock analysis. Making investing decisions, nevertheless, requires caution because predicting stock prices is a difficult undertaking that considers many different factors. In the face of market changes, continuous examination of the model findings and their conjunction with other data may assist boost confidence and support wiser investment choices.



According to the data in the graph "Comparison of Actual vs. Predicted Stock Prices," the RNN model has produced quite accurate predictions in recreating the actual stock price movements. The variations and discrepancies between the actual stock prices and the model's forecasts serve as a reminder that the stock market is still complicated and influenced by a variety of outside variables that are hard to properly anticipate. To deal with the constantly shifting market conditions, careful study and recent modifications to the model are still required. Investment decisions should still be taken with prudence. Exciting potential exists in using RNN models as a stock analysis tool to comprehend market activity. Investors should not, however, only depend on model projections and should continue to be cognizant of the dangers associated with stock investing. Making better educated investment selections that consider different analyses and pieces of information can help investors succeed on the stock market.

6.3 Interpretation of Results

The research's conclusions show that the RNN-based model used to forecast the stock price of Amazon Inc. produced promising outcomes. The model's remarkably small Root Mean Squared Error (RMSE) score of 15.67 demonstrated its ability to faithfully reproduce historical stock price movements. This shows how well the model captures the fundamental trends in Amazon's stock prices.

The RNN model's accuracy in replicating the general patterns in stock price is further confirmed by a visual comparison of the real stock prices (red line) and the anticipated stock prices (blue line). The two lines' near-perfect alignment indicates the model's capacity to offer insightful forecast insights for upcoming stock prices.

It is important to recognize that predicting stock prices is still a difficult undertaking that is impacted by a variety of uncontrollable outside factors. The observed changes and variations between the actual and expected stock prices highlight the stock market's inherent uncertainty. Therefore, for investors and stakeholders, careful decision-making and thorough study beyond the model's forecasts are essential.

In conclusion, the RNN-based model has demonstrated promising skills in faithfully recreating the behavior of Amazon's stock price. When used in conjunction with fundamental analysis and taking other market-influencing factors into account, the model is effective at forecasting stock price patterns and may thus be a useful tool for stock analysis. However, given the complex and constantly changing environment of the stock market, investors should proceed with care and use the model's forecasts as part of a comprehensive strategy, considering various data sources and considerations.

Chapter 7
Conclusion

7.1 Summary of Findings

The study that used Recurrent Neural Networks (RNN) to anticipate Amazon's stock price generated substantial results. The RNN model, which was trained on historical stock data from July 18, 2022 to July 14, 2023, proved the capacity to anticipate stock price fluctuations effectively. The model's Root Mean Squared Error (RMSE) score of 15.67 indicates its ability to capture underlying patterns in Amazon's stock movements.

The comparison of real and forecasted stock prices validated the model's performance even further. The near alignment of the two lines demonstrated the model's capacity to deliver useful insights and accurate projections for future stock prices. The research effectively contributed to the knowledge of stock price behavior and provided important information for investors and stakeholders by employing RNN and evaluating the dynamics of Amazon's stock market.

The findings highlight the need of using advanced methodologies like RNN in economic modeling and data analysis to make accurate stock price forecasts. The study adds to the realm of data-driven decision-making in the financial sector by giving useful insights for projecting future stock prices and bolstering well-informed investment plans.

7.2 Limitations and Future Work

Although the study's findings on the use of Recurrent Neural Networks (RNN) for stock price forecasting are encouraging, there are a few considerations and opportunities for future development. The following are the constraints encountered and recommendations for future research:

1. Level of Uncertainty: There is always a degree of uncertainty surrounding stock price forecasts. To enhance prediction accuracy, it is suggested to incorporate more real-time data, such as current events or economic indicators, so that the model can respond more effectively to market changes.
2. Generalizability: The research focused on Amazon.com's stock price. For the model to perform well in a variety of market scenarios, it is suggested that the research be expanded to include information from other companies or industries.
3. Comparison to other Prediction Methods: Despite the effectiveness of the RNN model, it is essential to compare it to other prediction methods. By contrasting various models, it is possible to comprehend the benefits and drawbacks of each strategy and select the most appropriate one for a given circumstance.
4. Ease and Interpretability: It is essential to ensure that the model is readily understood by non-experts to increase the prediction's credibility and applicability. The results of predictions should be communicated to a broader audience with more precise language.

5. Dealing with Uncertainty: The irrational characteristics of the stock market make it difficult for the model to predict certain abrupt changes. Future research aims to develop more accurate models that account for uncertainty and provide a clearer picture of potential outcomes.
6. Real-Time Application: Implementing the model in a real-time environment will improve decision-making and provide traders and investors with timely information.
7. Use of User-Friendly Platforms: It is recommended to establish a user-friendly platform or utility that generates predictions with the model so that more people can access it and make informed investment decisions.
8. Ethical Considerations: As a data analyst, it is essential to consider ethical issues, such as bias and data privacy, when conducting this research. Future research should place a greater emphasis on ethical conduct and careful data utilization.

Chapter 8

References

Zhang, A., Cai, J., Zhu, B., Yu, W., Huang, Y., & Zhou, L. (2023, July). Stock Price Prediction by Using RNN Method. In 2023 2nd International Conference on Social Sciences and Humanities and Arts (SSHA 2023) (pp. 923-929). Atlantis Press.

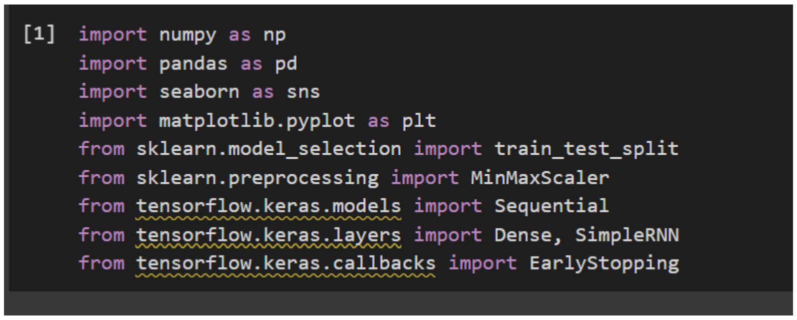
Yahoo Finance. (2023). Amazon Inc. Historical Data. Retrieved July 16, 2023, from https://finance.yahoo.com/quote/AMZN/history?fr=sycsrp_catchall

Chapter 9

Appendices

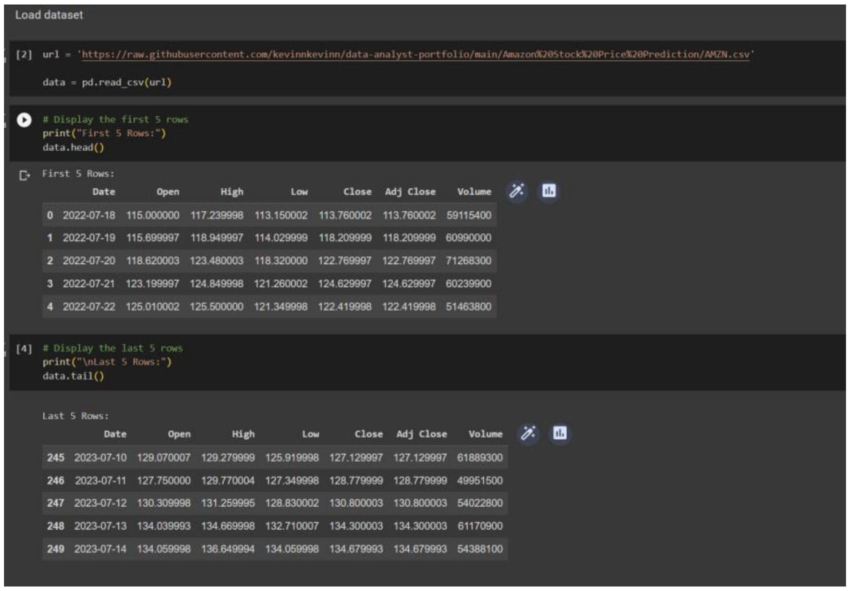
9.1 Code Implementation (Python)

- Import Library



```
[1]  import numpy as np
     import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import MinMaxScaler
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense, SimpleRNN
     from tensorflow.keras.callbacks import EarlyStopping
```

- Load Dataset



Load dataset

```
[2]  url = 'https://raw.githubusercontent.com/kevinnkevinn/data-analyst-portfolio/main/Amazon%20Stock%20Price%20Prediction/AMZN.csv'

     data = pd.read_csv(url)
```

```
# Display the first 5 rows
print("First 5 Rows:")
data.head()
```

```
First 5 Rows:
```

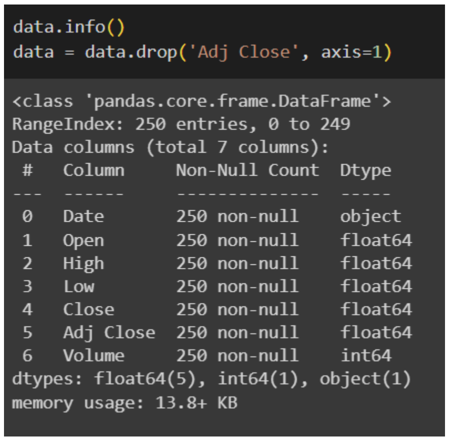
	Date	Open	High	Low	Close	Adj Close	Volume
0	2022-07-18	115.000000	117.239998	113.150002	113.760002	113.760002	59115400
1	2022-07-19	115.699997	118.949997	114.029999	118.209999	118.209999	60990000
2	2022-07-20	118.620003	123.480003	118.320000	122.769997	122.769997	71268300
3	2022-07-21	123.199997	124.849998	121.260002	124.629997	124.629997	60239900
4	2022-07-22	125.010002	125.500000	121.349998	122.419998	122.419998	51463800

```
[4]  # Display the last 5 rows
     print("\nLast 5 Rows:")
     data.tail()
```

```
Last 5 Rows:
```

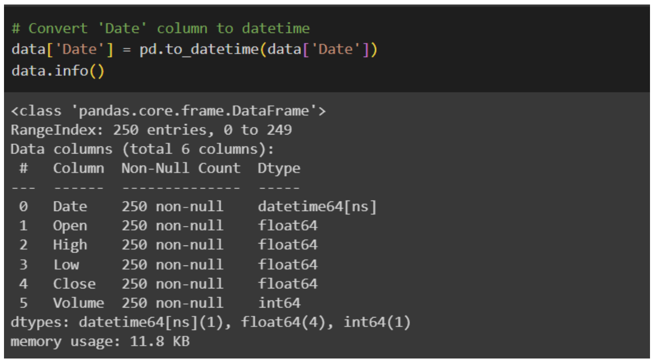
	Date	Open	High	Low	Close	Adj Close	Volume
245	2023-07-10	129.070007	129.279999	125.919998	127.129997	127.129997	61889300
246	2023-07-11	127.750000	129.770004	127.349998	128.779999	128.779999	49951500
247	2023-07-12	130.309998	131.259995	128.830002	130.800003	130.800003	54022800
248	2023-07-13	134.039993	134.669998	132.710007	134.300003	134.300003	61170900
249	2023-07-14	134.059998	136.649994	134.059998	134.679993	134.679993	54388100

- Gather data information and change data type



```
data.info()
data = data.drop('Adj Close', axis=1)
```

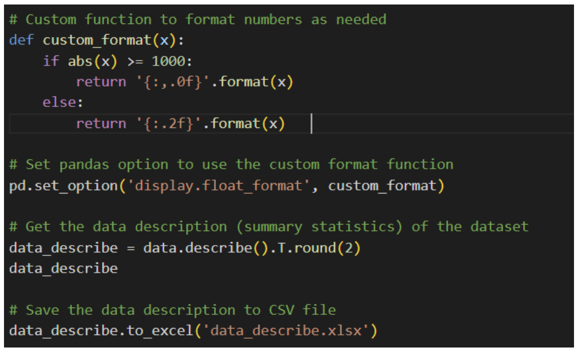
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       250 non-null    object 
 1   Open       250 non-null    float64
 2   High       250 non-null    float64
 3   Low        250 non-null    float64
 4   Close      250 non-null    float64
 5   Adj Close  250 non-null    float64
 6   Volume     250 non-null    int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 13.8+ KB
```



```
# Convert 'Date' column to datetime
data['Date'] = pd.to_datetime(data['Date'])
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    250 non-null    datetime64[ns]
 1   Open    250 non-null    float64
 2   High    250 non-null    float64
 3   Low     250 non-null    float64
 4   Close   250 non-null    float64
 5   Volume  250 non-null    int64
dtypes: datetime64[ns](1), float64(4), int64(1)
memory usage: 11.8 KB
```

- Data Describe



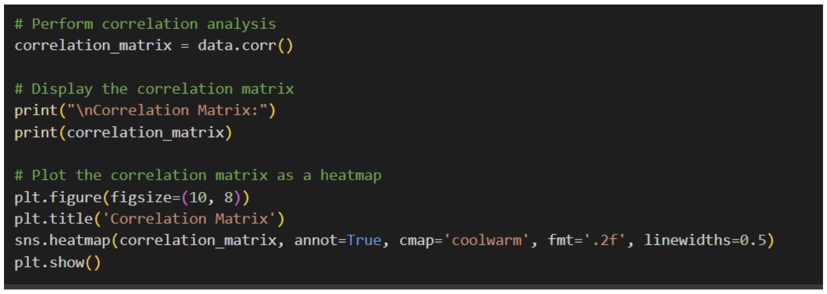
```
# Custom function to format numbers as needed
def custom_format(x):
    if abs(x) >= 1000:
        return '{:,.0f}'.format(x)
    else:
        return '{:.2f}'.format(x)

# Set pandas option to use the custom format function
pd.set_option('display.float_format', custom_format)

# Get the data description (summary statistics) of the dataset
data_describe = data.describe().T.round(2)
data_describe

# Save the data description to CSV file
data_describe.to_excel('data_describe.xlsx')
```

- Correlation Analysis

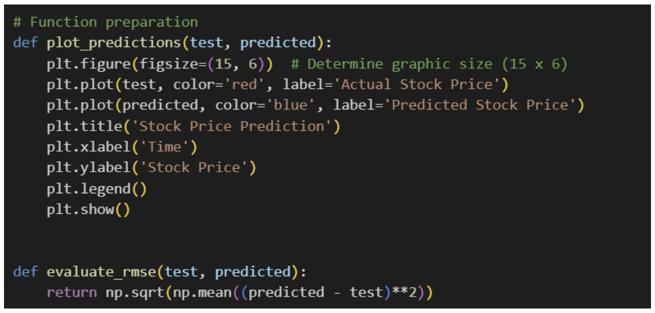


```
# Perform correlation analysis
correlation_matrix = data.corr()

# Display the correlation matrix
print("\nCorrelation Matrix:")
print(correlation_matrix)

# Plot the correlation matrix as a heatmap
plt.figure(figsize=(10, 8))
plt.title('Correlation Matrix')
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.show()
```

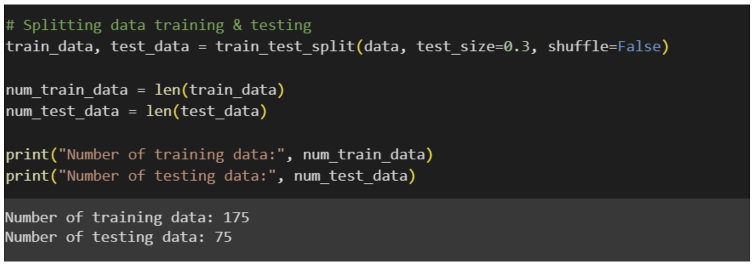
- Create Function ‘plot_prediction’ and ‘evaluate_rmse’



```
# Function preparation
def plot_predictions(test, predicted):
    plt.figure(figsize=(15, 6))  # Determine graphic size (15 x 6)
    plt.plot(test, color='red', label='Actual Stock Price')
    plt.plot(predicted, color='blue', label='Predicted Stock Price')
    plt.title('Stock Price Prediction')
    plt.xlabel('Time')
    plt.ylabel('Stock Price')
    plt.legend()
    plt.show()


def evaluate_rmse(test, predicted):
    return np.sqrt(np.mean((predicted - test)**2))
```

- Splitting data



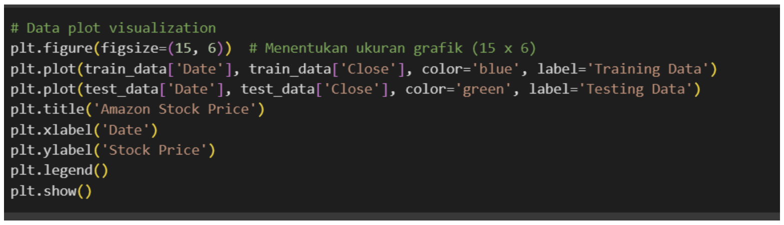
```
# Splitting data training & testing
train_data, test_data = train_test_split(data, test_size=0.3, shuffle=False)

num_train_data = len(train_data)
num_test_data = len(test_data)

print("Number of training data:", num_train_data)
print("Number of testing data:", num_test_data)
```

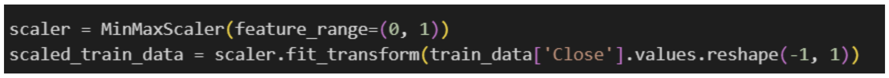
```
Number of training data: 175
Number of testing data: 75
```

- Create Data Visualization



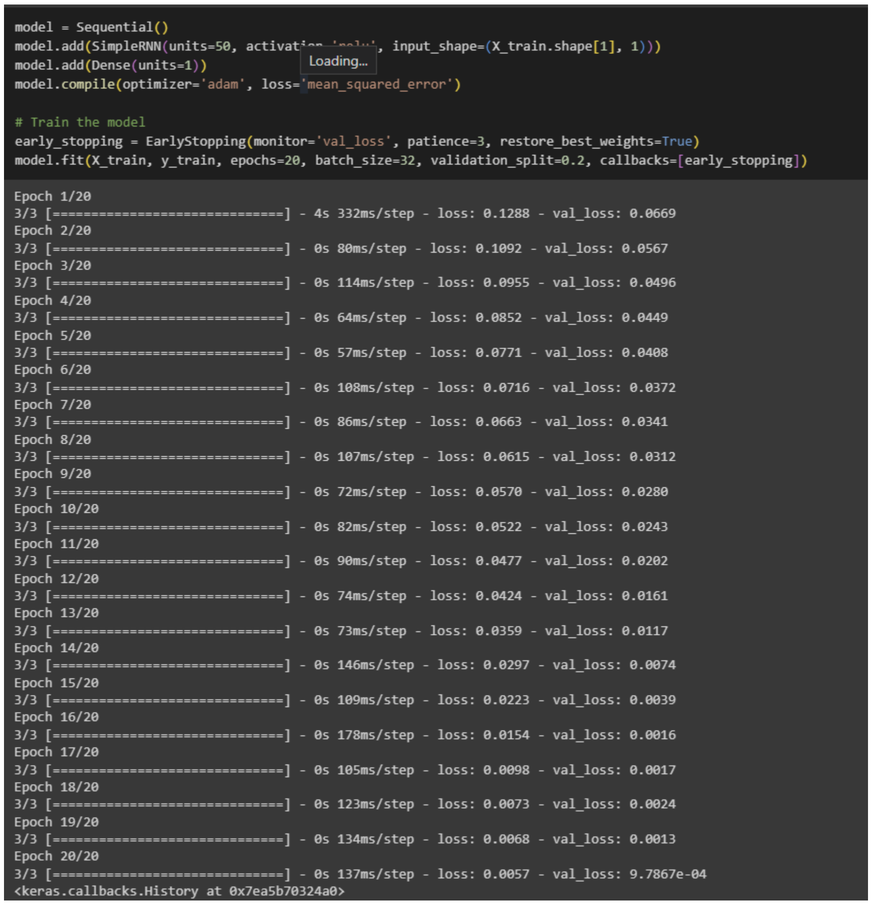
```
# Data plot visualization
plt.figure(figsize=(15, 6))  # Menentukan ukuran grafik (15 x 6)
plt.plot(train_data['Date'], train_data['Close'], color='blue', label='Training Data')
plt.plot(test_data['Date'], test_data['Close'], color='green', label='Testing Data')
plt.title('Amazon Stock Price')
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.legend()
plt.show()
```

- Scaling Data



```
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_train_data = scaler.fit_transform(train_data['Close'].values.reshape(-1, 1))
```

- Create and Train RNN model.

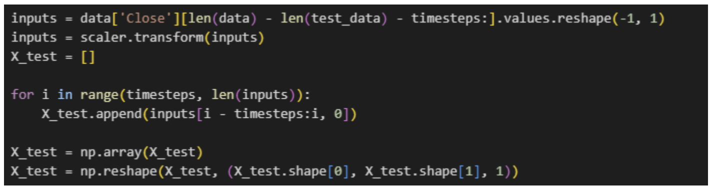


```
model = Sequential()
model.add(SimpleRNN(units=50, activat[illegible], input_shape=(X_train.shape[1], 1)))
model.add(Dense(units=1))
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
early_stopping = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)
model.fit(X_train, y_train, epochs=20, batch_size=32, validation_split=0.2, callbacks=[early_stopping])
```

```
Epoch 1/20
3/3 [==============================] - 4s 332ms/step - loss: 0.1288 - val_loss: 0.0669
Epoch 2/20
3/3 [==============================] - 0s 80ms/step - loss: 0.1092 - val_loss: 0.0567
Epoch 3/20
3/3 [==============================] - 0s 114ms/step - loss: 0.0955 - val_loss: 0.0496
Epoch 4/20
3/3 [==============================] - 0s 64ms/step - loss: 0.0852 - val_loss: 0.0449
Epoch 5/20
3/3 [==============================] - 0s 57ms/step - loss: 0.0771 - val_loss: 0.0408
Epoch 6/20
3/3 [==============================] - 0s 108ms/step - loss: 0.0716 - val_loss: 0.0372
Epoch 7/20
3/3 [==============================] - 0s 86ms/step - loss: 0.0663 - val_loss: 0.0341
Epoch 8/20
3/3 [==============================] - 0s 107ms/step - loss: 0.0615 - val_loss: 0.0312
Epoch 9/20
3/3 [==============================] - 0s 72ms/step - loss: 0.0570 - val_loss: 0.0280
Epoch 10/20
3/3 [==============================] - 0s 82ms/step - loss: 0.0522 - val_loss: 0.0243
Epoch 11/20
3/3 [==============================] - 0s 90ms/step - loss: 0.0477 - val_loss: 0.0202
Epoch 12/20
3/3 [==============================] - 0s 74ms/step - loss: 0.0424 - val_loss: 0.0161
Epoch 13/20
3/3 [==============================] - 0s 73ms/step - loss: 0.0359 - val_loss: 0.0117
Epoch 14/20
3/3 [==============================] - 0s 146ms/step - loss: 0.0297 - val_loss: 0.0074
Epoch 15/20
3/3 [==============================] - 0s 109ms/step - loss: 0.0223 - val_loss: 0.0039
Epoch 16/20
3/3 [==============================] - 0s 178ms/step - loss: 0.0154 - val_loss: 0.0016
Epoch 17/20
3/3 [==============================] - 0s 105ms/step - loss: 0.0098 - val_loss: 0.0017
Epoch 18/20
3/3 [==============================] - 0s 123ms/step - loss: 0.0073 - val_loss: 0.0024
Epoch 19/20
3/3 [==============================] - 0s 134ms/step - loss: 0.0068 - val_loss: 0.0013
Epoch 20/20
3/3 [==============================] - 0s 137ms/step - loss: 0.0057 - val_loss: 9.7867e-04
<keras.callbacks.History at 0x7ea5b70324a0>
```

- Prepare Test data for Prediction.

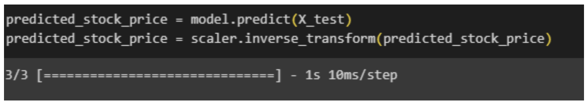


```
inputs = data['Close'][len(data) - len(test_data) - timesteps:].values.reshape(-1, 1)
inputs = scaler.transform(inputs)
X_test = []

for i in range(timesteps, len(inputs)):
    X_test.append(inputs[i - timesteps:i, 0])

X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
```

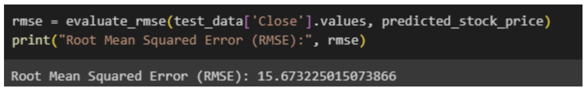
- Predict Stock Price



```
predicted_stock_price = model.predict(X_test)
predicted_stock_price = scaler.inverse_transform(predicted_stock_price)
```

```
3/3 [==============================] - 1s 10ms/step
```

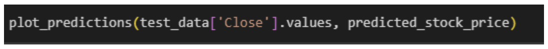
- Calculate RMSE



```
rmse = evaluate_rmse(test_data['Close'].values, predicted_stock_price)
print("Root Mean Squared Error (RMSE):", rmse)

Root Mean Squared Error (RMSE): 15.673225015073866
```

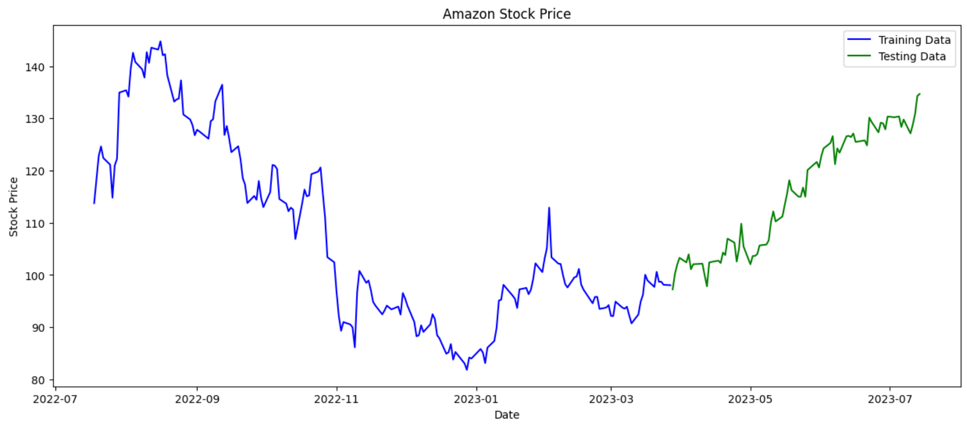
- Visualize the prediction.



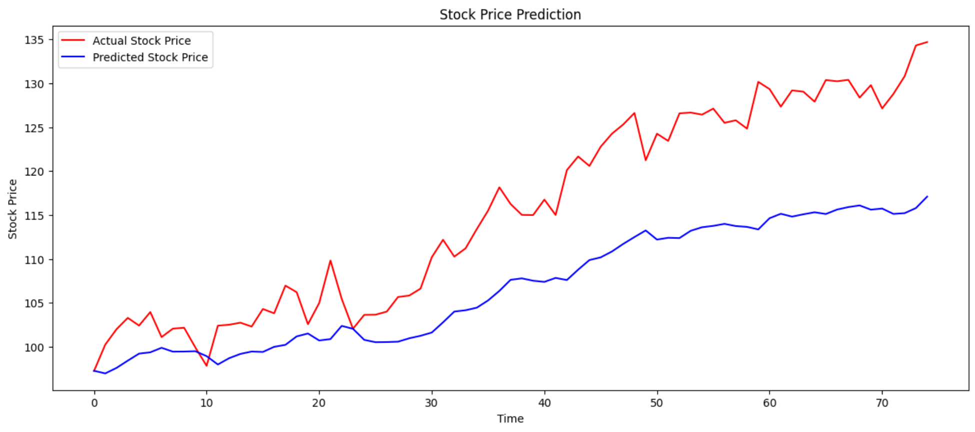
```
plot_predictions(test_data['Close'].values, predicted_stock_price)
```

9.2 Additional Plots and Visualizations

- Amazon Stock Price



- Stock Price Prediction


Stock Price Prediction
Actual Stock Price
Predicted Stock Price
Stock Price
Time