Exploring and Analyzing Tech Stock Performance: A Data Visualization Approach

Rohit Gupta

Kalinga Institute of Industrial Technology (KIIT), Bhubaneswar, Odisha

1. Abstract:

This project aims to leverage the power of data visualization and manipulation libraries, including Matplotlib, Seaborn, Plotly, Pandas, and NumPy, to explore and analyze the stock performance of leading technology companies. The primary objective is to visualize and analyze the stock price movements of prominent technology companies, namely Apple Inc., Microsoft Corporation, Amazon.com Inc., and Google., over a defined time period from March 1, 2023 to April 26, 2024. The project employs a multi-faceted approach that combines data visualization, data manipulation, and comparative analysis techniques to uncover meaningful insights and trends within the stock data. While the project primarily deals with static data, it sets the stage for future enhancements involving dynamic data sources and predictive modeling, particularly employing advanced modeling technique ARIMA model that uses time series data to better understand the data set or to predict future trends. The findings from this project lay the foundation for informed decision-making and future predictive modeling endeavors in stock price forecasting.

This project investigates the stock performance of leading technology companies - Apple Inc., Microsoft Corporation, Amazon.com Inc., and Google- using data visualization and manipulation libraries like Matplotlib, Seaborn, Plotly, Pandas, and NumPy. The analysis focuses on price movements from March 1, 2023 to April 26, 2024.

2. Key Findings:

Specific Insights: Upon comparing the stock prices, a notable observation emerged: Microsoft consistently exhibited higher opening and closing prices compared to the other companies. Additionally, Microsoft's stock consistently achieved higher highs throughout the analyzed period.

Maximum Volume Traded: Analysis of the volume traded revealed distinct peaks for each company. The maximum volume traded for Apple occurred on February 29, 2024, while Amazon experienced its peak on August 4, 2023. Both Microsoft and Google recorded their highest trading volumes on December 15, 2023.

Daily Return Analysis: Examining the daily return averages provided further insights. Notably, Apple recorded its maximum daily return on April 11, 2024, while Amazon's peak occurred on October 6, 2023. Microsoft experienced its highest daily return on July 18, 2023, and Google's peak was observed on April 1, 2024.

These findings shed light on distinct trends and patterns within the stock market, offering valuable insights for investors and analysts alike.

3. Future Directions:

The project lays the groundwork for further exploration using dynamic data sources and ARIMA models for time series forecasting.

This initial analysis can be a springboard for developing more sophisticated predictive models to understand market dynamics and potentially forecast future trends.

4. Keywords: Stock Performance, Data Visualization, Matplotlib, Seaborn, Plotly, Pandas, NumPy, Time Series Forecasting, ARIMA Models, Market Dynamics

5. Introduction:

The ability to predict stock prices has long been sought after by market and financial analysts. This project uses the power of data visualization and functional libraries such as Matplotlib, Seaborn, Plotly, Pandas and NumPy to explore and analyze business technology products of companies in the market. Additionally, the project created a framework for future predictions by integrating dynamic data and using advanced modeling techniques such as the ARIMA (Autoregressive Integrated Moving Average) model specifically designed to analyze time records.

5.1 Scope

The scope of this project encompasses the visualization and analysis of stock data for four major technology companies: Apple Inc., Microsoft Corporation, Amazon.com Inc., and Google. The analysis will focus on the stock prices from March 1, 2023 to April 26, 2024, providing a comprehensive snapshot of each company's performance during this period. While the project primarily deals with static data, it paves the way for future enhancements involving dynamic data sources and predictive modeling.

6. Data Description and preprocessing :

The dataset contains historical stock prices for four major technology companies: Apple Inc. (AAPL), Microsoft Corporation (MSFT), Amazon.com Inc. (AMZN), and Google (GOOGL). The data covers the time period from March 1, 2023 to April 26, 2024 and includes the following variables:

1. Date: The date of the trading day.
2. Open Price: The price of the stock at the beginning of the trading day.
3. Closing Price: The price of the stock at the end of the trading day.
4. Adjusted Closing Price: The closing price adjusted for any corporate actions such as stock splits or dividends.
5. High Price: The highest price the stock reached during the trading day.
6. Low Price: The lowest price the stock reached during the trading day.
7. Volume: The total number of shares traded during the trading day.

The dataset provides a comprehensive overview of the daily trading activity for each company, including the opening and closing prices, adjusted prices, daily high and low prices, and trading volume. This information is valuable for analyzing stock performance, identifying trends, and making informed investment decisions.

The data set was carefully preprocessed to ensure the suitability and reliability of the analysis. Steps are taken to check and correct any inaccuracies or inconsistencies to ensure the accuracy and reliability of the results. The date line has been converted to date format to identify the time. Additionally, the calculation of the adjusted closing price also reflects the factors that affect the closing price. These preliminary steps provide a solid foundation for critical evaluation and understanding of stock price data.

7. Data Visualization:

Data visualization is essential to understand complex data and gain insight. In this project, I created various visualizations that provide detailed information about Apple Inc., Microsoft Corporation, Amazon.com Inc.,and Google's products. From March 1, 2023 to April 26, 2024. Visualizations include charts showing high, low, open, and close price changes over time, as well as charts for adjustment closing costs that include transactions such as distributions or dividends. Using data visualization libraries such as Matplotlib, Seaborn and Plotly, I present the data in a clear and understandable way that facilitates in-depth analysis and interpretation. These recommendations are useful tools for anyone who wants to understand what is happening in the market and make informed decisions.

Fig. 1 : Open Price

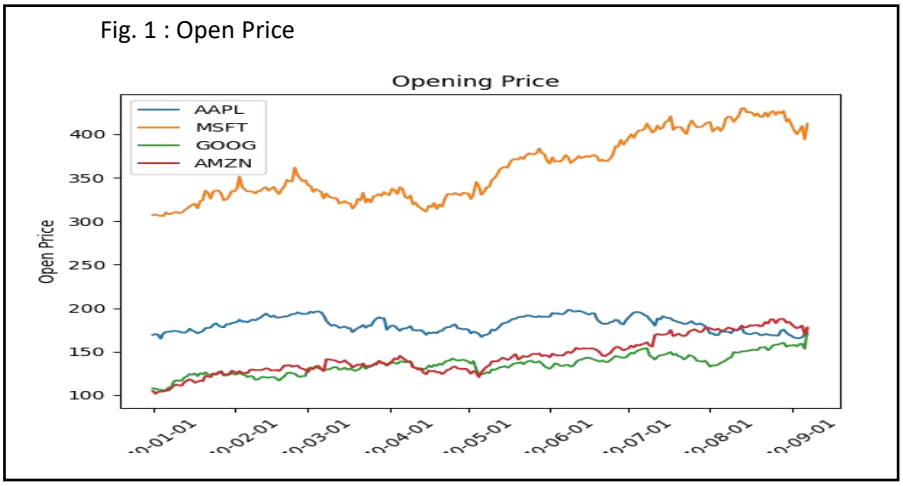


Fig. 2 : Closing Price

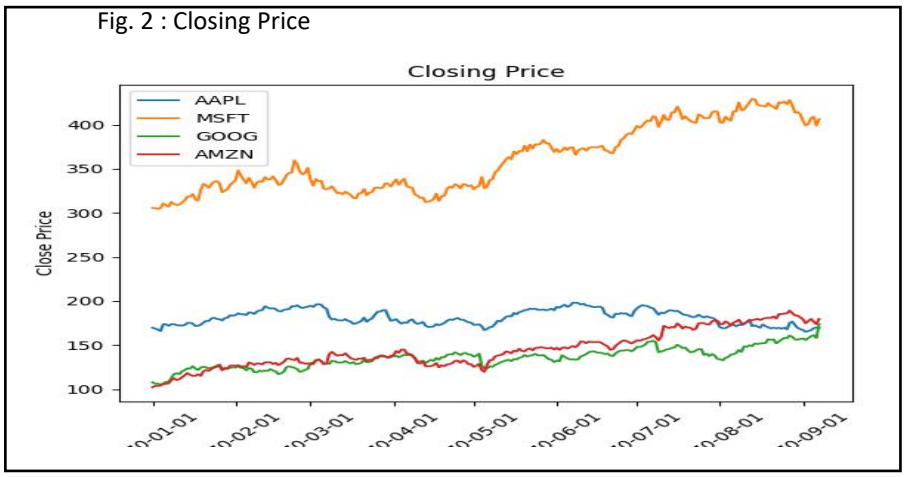


Fig. 3 : Low Price

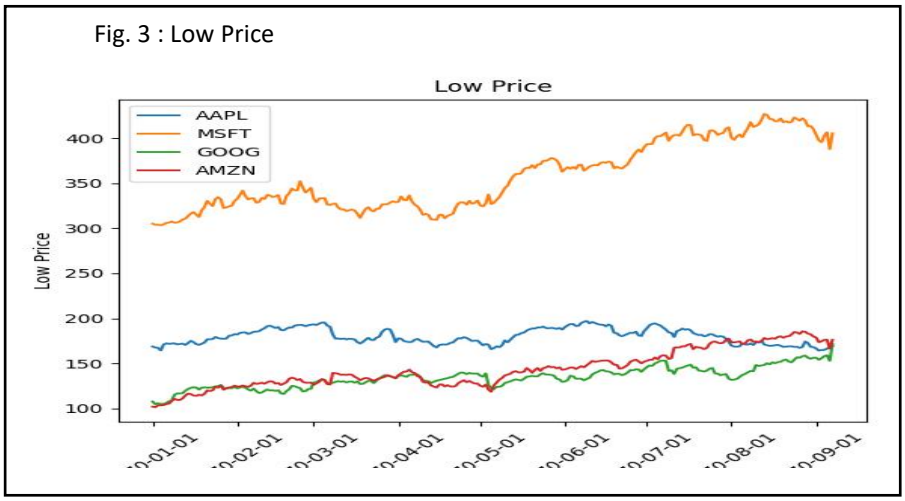


Fig. 4 : High Price

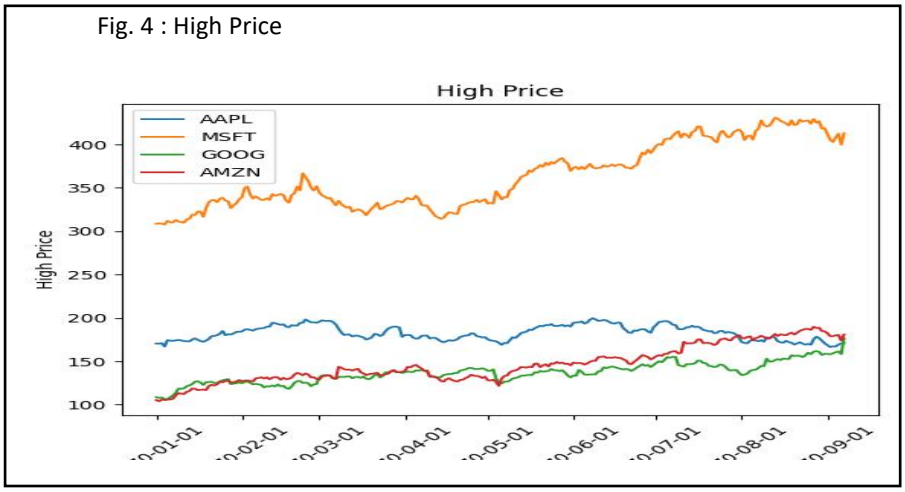
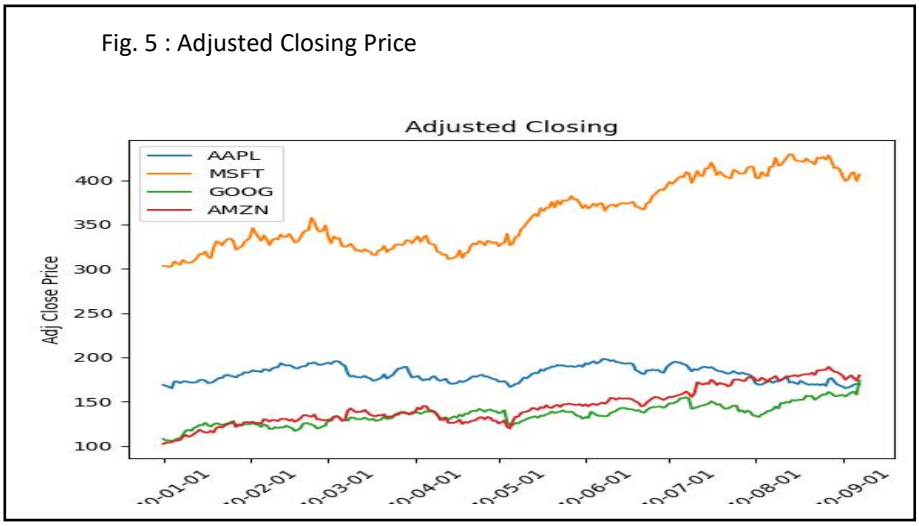


Fig. 5 : Adjusted Closing Price



8. <u>Data Manipulation and Visualization:</u>

First of all I loaded a database containing historical stock prices of Apple Inc., Microsoft Corporation, Amazon.com Inc., Google from March 1, 2023 to April 26, 2024. I have completed the process of cleaning the data, removing null values, and fixing any inconsistencies. I continued to examine it using various visual materials with the cleaning materials I had. Charts are used to show the high, low, opening and closing price differences of each company over time, thus providing information about stock price movements. I also created a closed price adjustment table that shows the adjusted price including market conditions such as distributions or dividends. This information helps you gain a deeper understanding of product performance and uncover key patterns and trends.

Here I've provided the code snippet of the Opening Price and Closing Price :

Fig. 6 : Code snippet for Opening Price

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

# Plot the data
plt.plot(df_AAPL['Date'], df_AAPL['Open'], label='AAPL')
plt.plot(df_AAPL['Date'], df_MSFT['Open'], label='MSFT')
plt.plot(df_AAPL['Date'], df_GOOG['Open'], label='GOOG')
plt.plot(df_AAPL['Date'], df_AMZN['Open'], label='AMZN')

# Set title and labels
plt.title("Opening Price ")
plt.xlabel('Date')
plt.ylabel('Open Price')

# Format x-axis dates
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))

# Set the frequency of ticks on the x-axis
plt.gca().xaxis.set_major_locator(mdates.MonthLocator())

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Display the legend
plt.legend()

plt.savefig('Opening Price.png')
plt.show()
```

Fig.7 : Code snippet for Closing Price

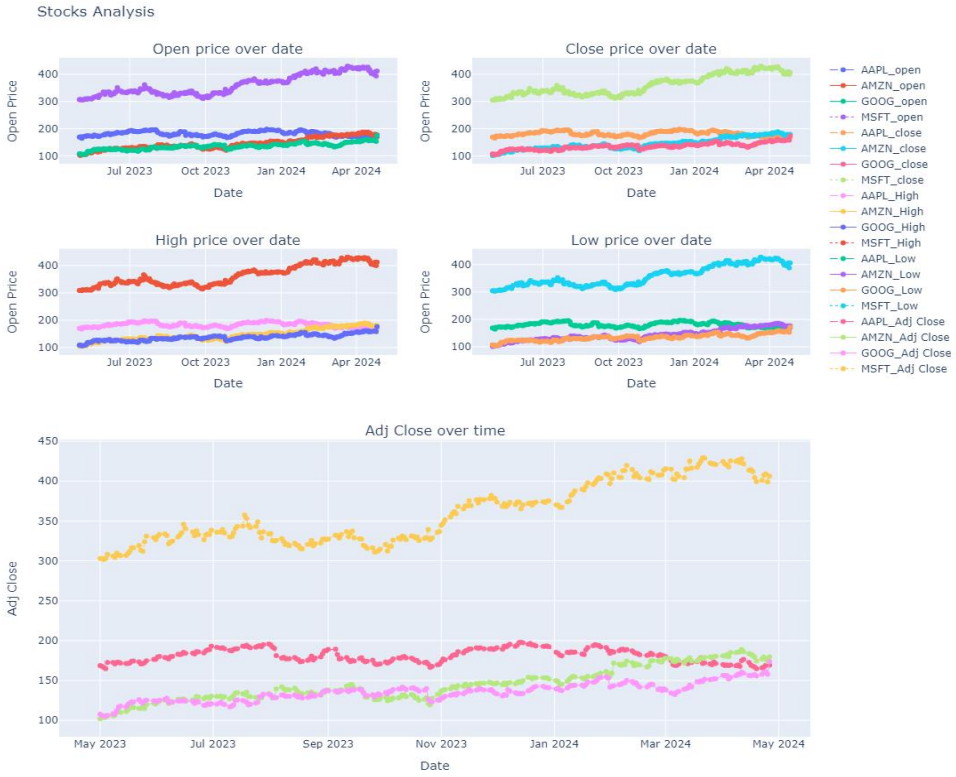
```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

# Plot the data
plt.plot(df_AAPL['Date'], df_AAPL['Close'], label='AAPL')
plt.plot(df_AAPL['Date'], df_MSFT['Close'], label='MSFT')
plt.plot(df_AAPL['Date'], df_GOOG['Close'], label='GOOG')
plt.plot(df_AAPL['Date'], df_AMZN['Close'], label='AMZN')

# Set title and labels
plt.title("Closing Price")
plt.xlabel('Date')
plt.ylabel('Close Price')

# Format x-axis dates
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))

# Set the frequency of ticks on the x-axis
plt.gca().xaxis.set_major_locator(mdates.MonthLocator())

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Display the legend
plt.legend()

plt.savefig('Closing Price.png')
plt.show()
```

These separate graphs provide a good understanding of the fluctuation of the prices. For a deeper insight, I've created subplots using Plotly. Here, I've provided a code snippet of a portion of the subplot (attaching the full code would be lengthy).

Fig. 8 : Code snippet for subplots

```
fig = go.Figure()
fig = make_subplots(rows=4, cols=2, subplot_titles=("Open price over date",
                                                    "Close price over date",
                                                    "High price over date",
                                                    "Low price over date",
                                                    "Adj Close over time"),
                    specs=[[{}, {}],
                           [{}, {}],
                           [{'rowspan': 2, 'colspan': 2}, None],
                           [None, None]],
                    print_grid=True)
# #for open price
fig.add_trace(go.Scatter(x=df.Date, y=df_AAPL.Open, mode='lines+markers', name='AAPL_open',
                         line=dict(width=1, dash='dashdot')), row=1, col=1)
fig.add_trace(go.Scatter(x=df.Date, y=df_AMZN.Open, mode='lines+markers', name='AMZN_open',
                         line=dict(width=1)), row=1, col=1)
fig.add_trace(go.Scatter(x=df.Date, y=df_GOOG.Open, mode='lines+markers', name='GOOG_open',
                         line=dict(width=1)), row=1, col=1)
fig.add_trace(go.Scatter(x=df.Date, y=df_MSFT.Open, mode='lines+markers', name='MSFT_open',
                         line=dict(width=1, dash='dot')), row=1, col=1)
```

Fig. 9 : Subplots for Stock Analysis



This is the outcome of the subplot, which gives a clear understanding of different prices at a single glance.

After manipulation and visualization of the prices I moved to visualise the volume being traded and here is the simple code snippet which explains the code.

Fig. 10 : Code snippet for Volume being traded overtime

```
fig = go.Figure()
fig = make_subplots(rows=4, cols=1,
                    specs=[[{}],
                           [{}],
                           [{}],
                           [{}]],
                    print_grid=True)
# #for open price
fig.add_trace(go.Scatter(x=df.Date, y=df_AAPL.Volume, mode='lines+markers', name='AAPL_Volume',
                         line=dict(width=1, dash='dashdot')), row=1, col=1)
fig.add_trace(go.Scatter(x=df.Date, y=df_AMZN.Volume, mode='lines+markers', name='AMZN_Volume',
                         line=dict(width=1)), row=2, col=1)
fig.add_trace(go.Scatter(x=df.Date, y=df_GOOG.Volume, mode='lines+markers', name='GOOG_Volume',
                         line=dict(width=1)), row=3, col=1)
fig.add_trace(go.Scatter(x=df.Date, y=df_MSFT.Volume, mode='lines+markers', name='MSFT_Volume',
                         line=dict(width=1, dash='dot')), row=4, col=1)


# Update xaxis properties
fig.update_xaxes(title_text="Date", row=1, col=1)
fig.update_xaxes(title_text="Date", row=2, col=1)
fig.update_xaxes(title_text="Date", row=3, col=1)
fig.update_xaxes(title_text="Date", row=4, col=1)

# update yaxis properties
fig.update_yaxes(title_text="AAPL_Volume", range=[0, 160000000], row=1, col=1)
fig.update_yaxes(title_text="AMZN_Volume", range=[0, 160000000], row=2, col=1)
fig.update_yaxes(title_text="GOOG_Volume", range=[0, 160000000], row=3, col=1)
fig.update_yaxes(title_text="MSFT_Volume", range=[0, 160000000], row=4, col=1)

# fig.update_layout(sub_plots)
fig.update_layout(height=1100, width=1200,
                  title_text="Volume being traded overtime")
```

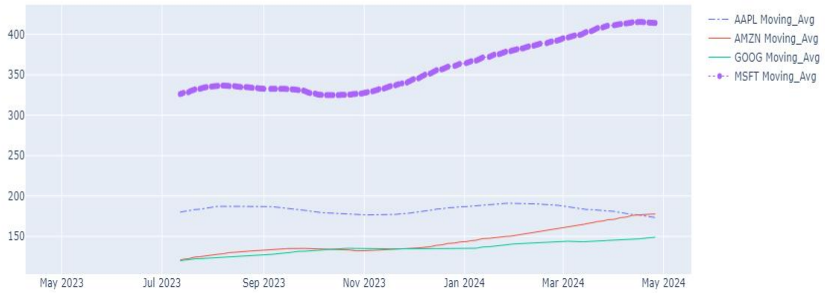
After analyzing stock price and volume data with various visualization methods, I further refine my analysis by calculating moving averages. The price data for each stock uses a moving average with a window size of 50 days. Analysts often use moving averages to identify changes in a security's strength. By keeping price movements over a period of time, moving averages provide information about the overall pattern of a stock's price movement. Analysts often use moving averages to confirm uncertainty about trends or potential changes in time, providing important signals in making business decisions.

Fig.11 : Moving Average

```
company_list = [df_AAPL,df_AMZN,df_GOOG,df_MSFT]
for list in company_list:
    window_size = 50
    closing_series = pd.Series(list['Close'])
    windows = closing_series.rolling(window_size)
    moving_average = windows.mean()
    moving_average_list = pd.DataFrame(moving_average)
    final_list = moving_average_list[window_size - 1:]
    pd.DataFrame(final_list)

    list['Moving Average'] = final_list
```

```
fig = go.Figure()

# Visualise volume being traded overtime
fig.add_trace(go.Scatter(x=df.Date, y=df_AAPL['Moving Average'], mode='lines', name='AAPL Moving_Avg',
                         line=dict(width=1, dash='dashdot')))
fig.add_trace(go.Scatter(x=df.Date, y=df_AMZN['Moving Average'], mode='lines', name='AMZN Moving_Avg',
                         line=dict(width=1)))
fig.add_trace(go.Scatter(x=df.Date, y=df_GOOG['Moving Average'], mode='lines', name='GOOG Moving_Avg',
                         line=dict(width=1)))
fig.add_trace(go.Scatter(x=df.Date, y=df_MSFT['Moving Average'], mode='lines+markers', name='MSFT Moving_Avg',
                         line=dict(width=1, dash='dot')))
```

Fig.12 : Moving Average



I have also created subplots for moving average, attaching the code and the graphs will make it lengthy.

After the moving average I've calculated the Daily return average. Daily return analysis is used to evaluate the stock from day to day. It compares one day's closing price with the previous day's closing price. Positive daily returns indicate that stock prices are rising.

Fig.13 : Daily Return Average

After analysing the daily return average of each company, I concluded that Daily return was maximum for Apple was on 11th April 2024, Amazon was on 6th October 2023, Microsoft was on 18th July 2023, and that of Google was on 1st April 2024.

9. Future Work and Enhancements:

In the section on Future Work and Development, I outline strategies to enhance the project's functionality and broaden its reach. A key objective is to incorporate dynamic data to augment the potency and adaptability of predictive models in line with market timing. This will necessitate the establishment of a pipeline for the prompt and precise receipt and processing of data. I'm also contemplating the use of forecasting methodologies such as the ARIMA (Autoregressive Integrated Moving Average) model. ARIMA models facilitate more precise future price predictions by identifying intricate patterns in price data through time series techniques. However, this endeavor presents its own set of challenges, including the complexity of the initial data, the refinement and training of the model, and the requirement to upkeep and update the model's prediction. Despite these hurdles, the advantages of merging robust data and top-tier modeling are substantial; it offers investors and alysts the chance to gain a deeper understanding of the business process and enhance decision-making.

10. Conclusion :

This project has provided valuable insights into the stock performance of leading technology companies, including Apple Inc., Microsoft Corporation, Amazon.com Inc., and Google. The analysis revealed various trends and patterns in the stock price data, highlighting the volatility and fluctuations inherent in the stock market. The effectiveness of data visualization techniques, such as line plots and subplots, was evident in providing a clear and intuitive representation of the data, enabling deeper analysis and interpretation. By visualizing trends in high, low, opening, and closing prices, as well as volume traded, I gained a comprehensive understanding of the stock market dynamics during the specified timeframe. Furthermore, the project's significance extends beyond mere analysis, laying the groundwork for future endeavors in stock prediction using dynamic data sources and advanced modeling techniques. The potential implications of this project are vast, offering investors and analysts valuable tools for making informed decisions and navigating the complexities of the stock market with greater confidence.