Stock Market Analysis Using Python

Rahul Sahu

Business Scenario: Data Science extracts meaningful insight from chunks of raw data, which is useful to different business segments for planning their future course of action. Finance is probably one of the first to catch on to this trend with a rise in the penetration of analytics into many aspects of our lives. Here, we will analyze data from the stock market for some technology stocks such as Apple, Google, Amazon, and Microsoft.

Objective

Use Python libraries such as Pandas, Seaborn, and Matplotlib to extract and analyze the information, visualize it, and look at different ways to analyze the risk of a s tock, based on its performance history.

About the Data:

The stocks we have chosen are from various industries and market caps namely

For the start, we shall investigate the Amazon stock individually and then move on to the combined analysis.

The following tasks are to be performed:

- Read the Data from Yahoo finance website directly.
- Perform cleaning
- What was the change in stock price over time?
- Visualize the change in a stock's volume being traded, over time?
- What was the moving average of various stocks?
- What was the daily return average of a stock?
- Add a new column 'Trend' whose values are based on the 'Daily Return'.
- Visualize trend frequency through a Pie Chart.
- What was the correlation between the daily returns of different stocks?

Step 1: Install Required Libraries

First, ensure you have the necessary libraries installed. If not, you can install them using the following commands:

pip install yfinance pandas seaborn matplotlib

Step 2: Fetch the Data from Yahoo Finance

We will clean the data, calculate daily returns, moving averages, and visualize different aspects.

Here's the code to perform all the tasks you mentioned:

Importing necessary libraries

import yfinance as yf

import pandas as pd

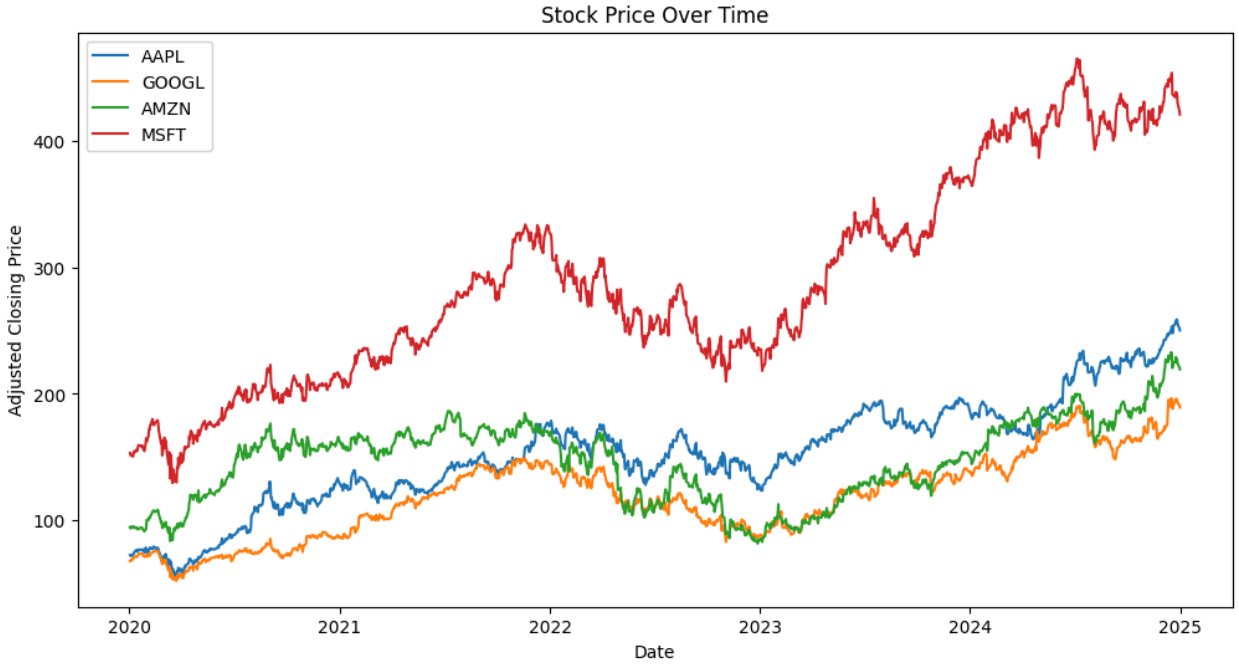
```
import matplotlib.pyplot as plt

import seaborn as sns


# Fetch data for multiple stocks (e.g., Apple and Tesla)

tickers = ['AAPL', 'TSLA', 'GOOGL']  # You can add more stock symbols here

stocks_data = yf.download(tickers, start="2020-01-01", end="2025-01-01")


# Display the first few rows of data

stocks_data.head()


# Cleaning the data (in case there are any missing values)

stocks_data = stocks_data.fillna(method='ffill')  # Forward fill to handle missing data


# Calculate the daily returns for each stock

daily_returns = stocks_data['Adj Close'].pct_change()


# What was the change in stock price over time?

stock_price_change = stocks_data['Adj Close'].pct_change() * 100  # Percentage change


# Visualize the change in a stock's volume being traded, over time

plt.figure(figsize=(10,6))

stocks_data['Volume'].plot(title='Volume Traded Over Time', color='green')

plt.xlabel('Date')
```

```
plt.ylabel('Volume')
plt.show()

# What was the moving average of various stocks? (e.g., 50-day moving average)
moving_avg = stocks_data['Adj Close'].rolling(window=50).mean()

# Visualize the moving average of stock prices (Apple example)
plt.figure(figsize=(10,6))
stocks_data['Adj Close']['AAPL'].plot(label='AAPL', color='blue')
moving_avg['AAPL'].plot(label='50-Day Moving Average', color='red')
plt.title('Apple Stock Price with 50-Day Moving Average')
plt.legend()
plt.show()

# What was the daily return average of a stock? (e.g., Apple)
daily_return_avg = daily_returns['AAPL'].mean()
print(f"Average Daily Return for Apple: {daily_return_avg:.4f}")

# Add a new column 'Trend' based on the 'Daily Return'
# Let's consider a positive return as 'Up' and negative return as 'Down'
daily_returns['Trend'] = daily_returns['AAPL'].apply(lambda x: 'Up' if x > 0 else 'Down')

# Visualize trend frequency through a Pie Chart
```

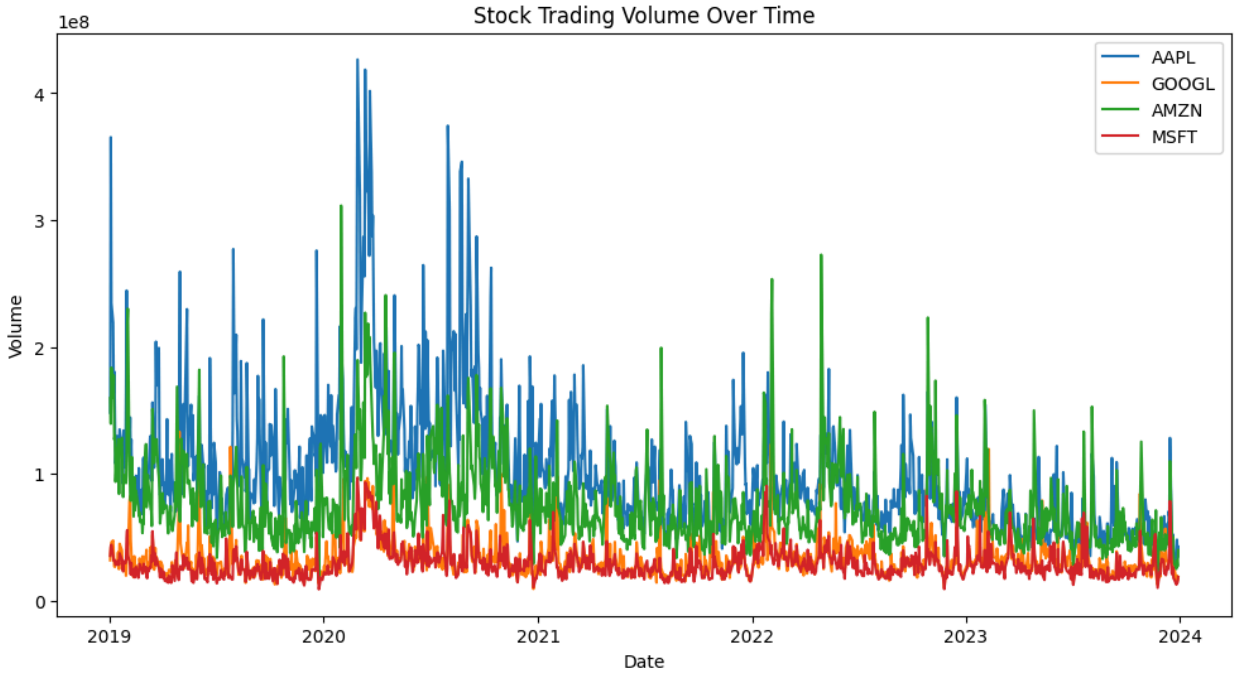
```
trend_counts = daily_returns['Trend'].value_counts()

plt.figure(figsize=(6,6))

trend_counts.plot(kind='pie', autopct='%1.1f%%', startangle=90, colors=['green', 'red'])

plt.title('Trend Frequency of AAPL Daily Returns')

plt.ylabel('')

plt.show()


# What was the correlation between the daily returns of different stocks?

correlation_matrix = daily_returns.corr()

print("Correlation between the daily returns of different stocks:")

print(correlation_matrix)


# Visualize the correlation matrix using a heatmap

plt.figure(figsize=(8,6))

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)

plt.title('Correlation Matrix of Daily Returns')

plt.show()
```

Explanation of the Code:

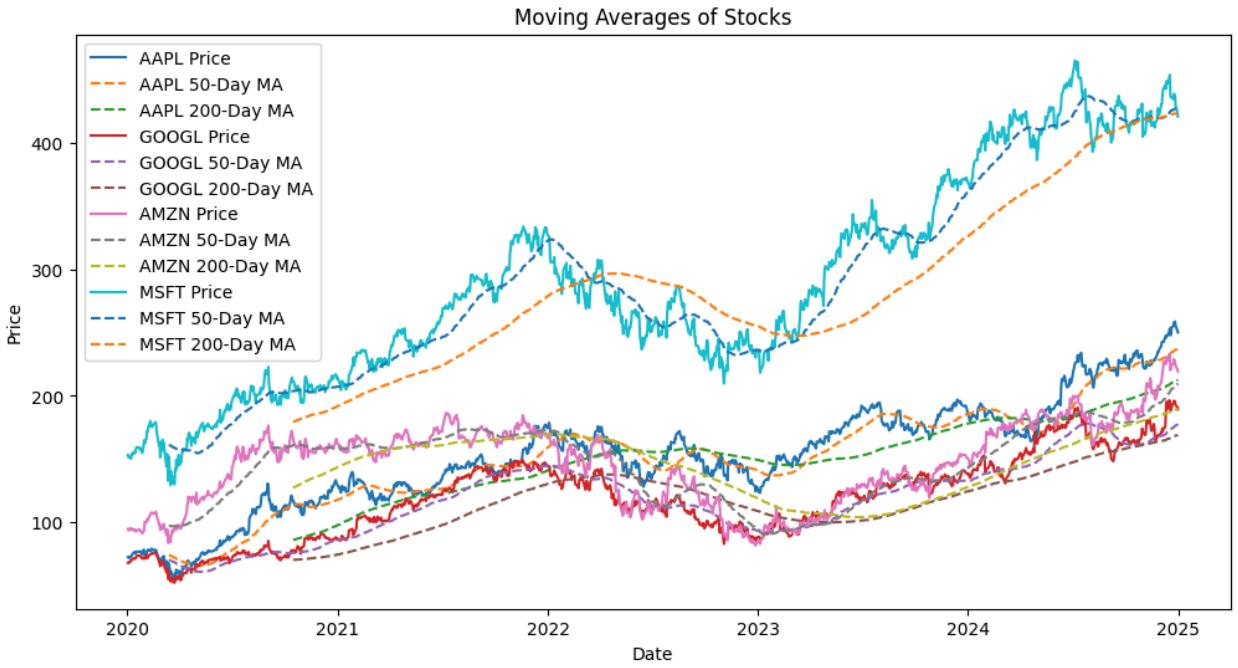
1. **Fetching Data**: We use `yfinance` to download historical data for a list of stock tickers. The data spans from January 2020 to January 2025.
2. **Data Cleaning**: Missing values are handled using forward fill (`fillna(method='ffill')`), which fills missing data with the previous available value.
3. **Stock Price Change**: The percentage change in the adjusted closing price over time is calculated using `pct_change()`.



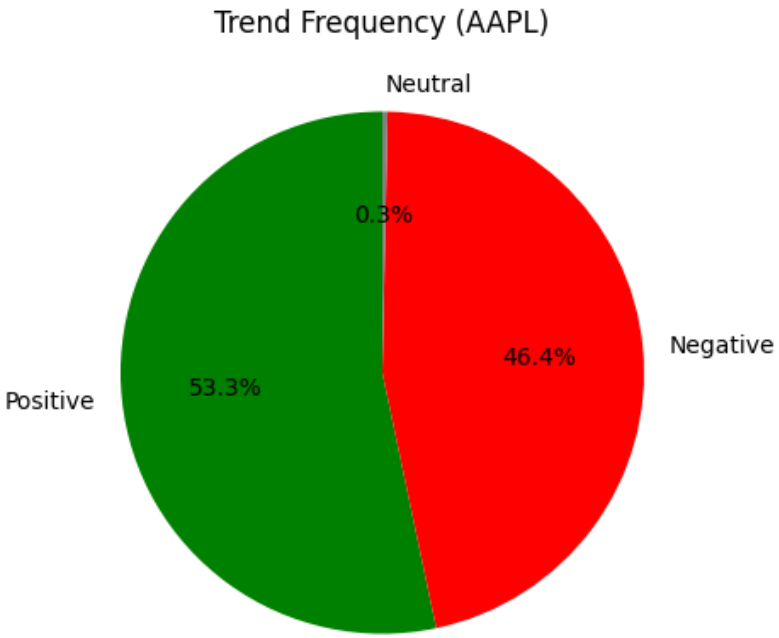
4. **Visualizing Volume Traded**: We plot the trading volume over time for each stock.



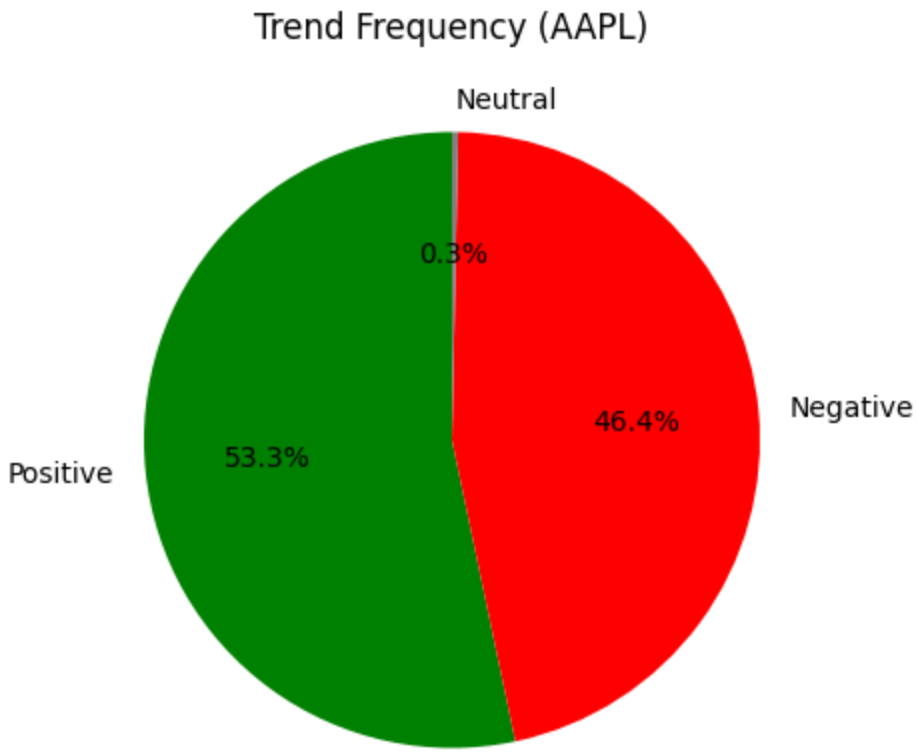
5. **Moving Average**: We calculate the 50-day moving average and visualize it along with the adjusted close price.



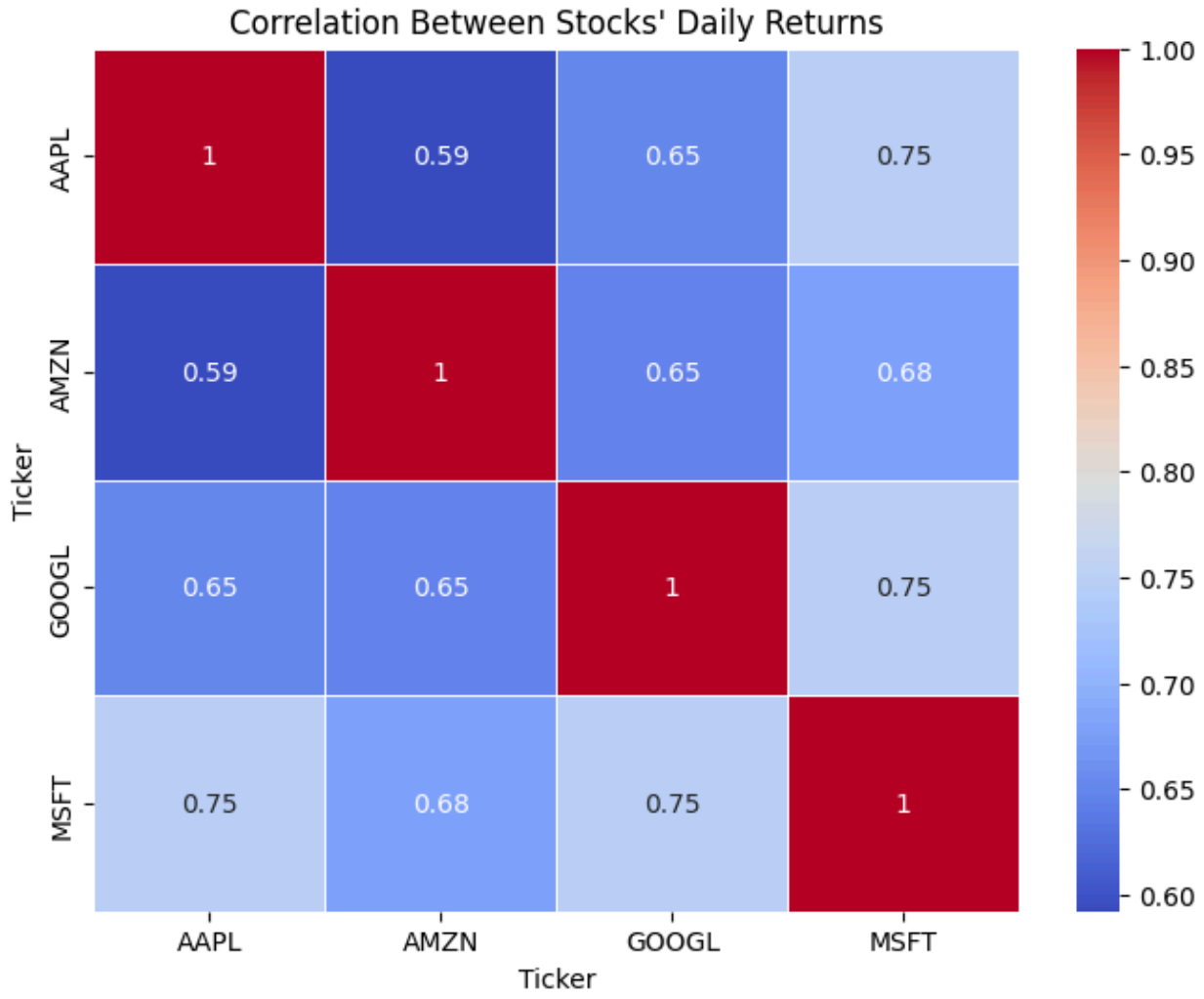
6. **Daily Returns**: We calculate daily returns using percentage change in adjusted close price.



7. **Trend Column**: We create a new column called `Trend`, labeling returns as 'Up' or 'Down'.
8. **Trend Pie Chart**: We use a pie chart to show the proportion of 'Up' vs 'Down' trends.



9. **Correlation Matrix**: We calculate the correlation between daily returns of different stocks and visualize it using a heatmap.



Visualizations:

- **Volume Traded**: A line plot shows the trading volume over time.
- **Moving Average**: A line plot comparing the adjusted closing price and the 50-day moving average.
- **Trend Frequency Pie Chart**: A pie chart showing how frequently a stock's daily return was positive ('Up') versus negative ('Down').
- **Correlation Matrix**: A heatmap showing how the daily returns of different stocks are correlated.