TITLE

Stock Price Movement Analysis

Submitted By:

Somya Mishra

B.Tech, [CSE-AI]

KIET Group of Institutions, Ghaziabad

DATE – 11 march 2025

INTRODUCTION

The stock market is a complex and dynamic system where the prices of stocks change continually based on market conditions, investor sentiment, and numerous economic and political factors. The study of stock price movement is an essential field of financial analysis. It helps investors make informed decisions about buying, selling, or holding stocks. Stock price movements are influenced by various factors such as company performance, market trends, news events, economic data, and investor behavior.

Stock price movement analysis involves examining historical data to identify patterns, trends, and predictions about future stock prices. This report focuses on analyzing stock price movements using various technical analysis techniques, statistical methods, and programming. In the age of data-driven decision-making, understanding stock price movements can provide valuable insights to financial professionals, traders, and researchers. This report will guide the process of collecting, processing, and analyzing stock price data using Python programming, and will include visualization techniques to make informed predictions about stock price trends.

Objectives of the Report:

- **To understand the fundamental principles of stock price movement analysis.**

- **To apply various technical indicators to analyze stock price movements.**
- **To demonstrate the use of Python programming for stock price analysis.**
- **To visualize stock data and forecast future stock trends.**

METHODOLOGY

In this section, we will outline the methodology used for analyzing stock price movements. The steps involved in this analysis are as follows:

Step 1: Data Collection

The first step involves collecting historical stock price data. This can be done using several free financial data sources such as:

- Yahoo Finance
- Alpha Vantage
- Quandl
- Google Finance

For this analysis, we will use Yahoo Finance API via the yfinance Python library. The historical stock data typically includes information like open, high, low, close prices, and the trading volume of stocks.

Step 2: Data Preprocessing

Once the data is collected, we need to preprocess it. This includes:

- Handling missing or null values.
- Converting date-time columns to appropriate formats.
- Resampling data (if needed) to a required frequency (daily, weekly, monthly).

- Calculating additional technical indicators like moving averages, RSI (Relative Strength Index), and MACD (Moving Average Convergence Divergence).

Step 3: Data Analysis

This step involves applying various technical analysis techniques:

- Moving Averages (MA): We will calculate Simple Moving Averages (SMA) and Exponential Moving Averages (EMA) to identify the overall trend of the stock.
- RSI: The Relative Strength Index is used to identify if the stock is overbought or oversold.
- MACD: The Moving Average Convergence Divergence is a momentum indicator used to identify changes in trend.
- Support and Resistance: Identifying key price levels that act as barriers in stock price movement.

Step 4: Visualization

Visualization is a key part of stock price analysis. Using libraries like matplotlib, seaborn, and plotly, we can create various charts such as:

- Line plots for stock prices.
- Candlestick charts for intraday stock movements.
- Scatter plots to visualize relationships between variables.

CODE TYPED

```
# Importing necessary libraries
import pandas as pd  # For data manipulation
import matplotlib.pyplot as plt  # For plotting graphs

# Load the stock data from a CSV file
file_path = "/content/stock_data.csv"  # Replace this with the path to your actual CSV file
df = pd.read_csv(file_path)  # Read the CSV file into a DataFrame

# Convert the 'Date' column to datetime format for easier manipulation
df['Date'] = pd.to_datetime(df['Date'])

# Sort the DataFrame by Date in ascending order to ensure chronological order
df = df.sort_values('Date')

# Set the 'Date' column as the index for time series analysis
df.set_index('Date', inplace=True)
```

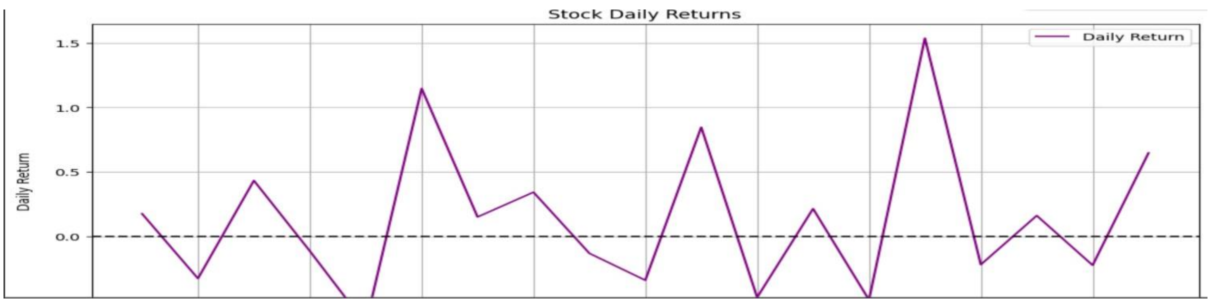
```
# Display the first few rows of the data to understand its structure
# This step helps verify the data after loading and converting the 'Date' column.
print(df.head())


# Calculate the 20-day and 50-day Simple Moving Averages (SMA)
# Moving averages help smooth out price data to identify trends over time
df['20-day MA'] = df['Close'].rolling(window=20).mean()  # 20-day moving average
df['50-day MA'] = df['Close'].rolling(window=50).mean()  # 50-day moving average


# Calculate the daily returns as the percentage change in the 'Close' price
# This helps in understanding the daily volatility and price movement.
df['Daily Return'] = df['Close'].pct_change()


# Calculate cumulative returns, which is the compounded return over time
# Cumulative return shows how an investment grows over time, considering all daily returns.
```

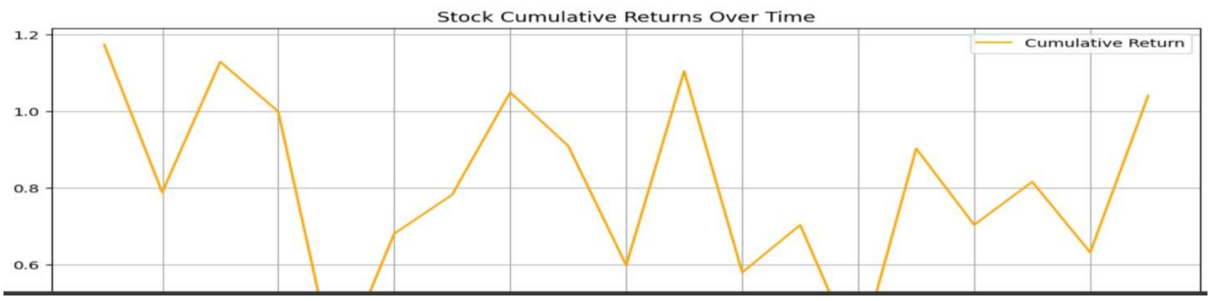
```python
df['Cumulative Return'] = (1 + df['Daily Return']).cumprod()


# Plotting the stock's closing price along with the moving averages
plt.figure(figsize=(12, 6))  # Set figure size for better readability
plt.plot(df.index, df['Close'], label="Closing Price", color='blue')  # Plot closing price in blue
plt.plot(df.index, df['20-day MA'], label="20-day MA", color='red', linestyle='dashed')  # Plot 20-day moving average in red
plt.plot(df.index, df['50-day MA'], label="50-day MA", color='green', linestyle='dashed')  # Plot 50-day moving average in green


# Adding labels and title to the plot
plt.xlabel("Date")  # X-axis label as 'Date'
plt.ylabel("Stock Price")  # Y-axis label as 'Stock Price'
plt.title("Stock Price Movement with Moving Averages")  # Title of the plot
plt.legend()  # Show legend to distinguish between lines
plt.grid(True)  # Show grid for better readability
plt.xticks(rotation=45)  # Rotate x-axis labels for better visibility
plt.show()  # Display the plot
```

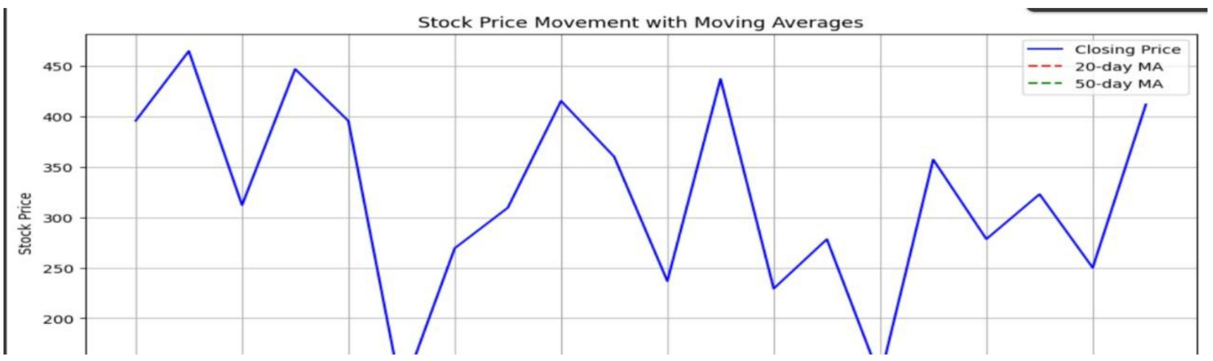
```
# Plotting the daily returns
plt.figure(figsize=(12, 6))  # Set figure size
plt.plot(df.index, df['Daily Return'], label="Daily Return", color='purple')  # Plot daily return in purple
plt.axhline(y=0, color='black', linestyle='dashed')  # Add horizontal line at y=0 for reference

# Adding labels and title to the plot
plt.xlabel("Date")  # X-axis label as 'Date'
plt.ylabel("Daily Return")  # Y-axis label as 'Daily Return'
plt.title("Stock Daily Returns")  # Title of the plot
plt.legend()  # Show legend to distinguish the line
plt.grid(True)  # Show grid for better readability
plt.xticks(rotation=45)  # Rotate x-axis labels for better visibility
plt.show()  # Display the plot

# Plotting the cumulative returns to visualize the overall growth of the investment over time
plt.figure(figsize=(12, 6))  # Set figure size
plt.plot(df.index, df['Cumulative Return'], label="Cumulative Return", color='orange')  # Plot cumulative return in orange
```

```
# Adding labels and title to the plot
plt.xlabel("Date")  # X-axis label as 'Date'
plt.ylabel("Cumulative Return")  # Y-axis label as 'Cumulative Return'
plt.title("Stock Cumulative Returns Over Time")  # Title of the plot
plt.legend()  # Show legend to distinguish the line
plt.grid(True)  # Show grid for better readability
plt.xticks(rotation=45)  # Rotate x-axis labels for better visibility
plt.show()  # Display the plot
```

SCREENSHOT OUTPUT


Stock Daily Returns
Daily Return
Daily Return


Stock Cumulative Returns Over Time
Cumulative Return


Stock Price Movement with Moving Averages
Closing Price
20-day MA
50-day MA
Stock Price