Title: Stock Data Analysis Using Python

Prepared by Roll No: Aryan Singh
ROLL-NO. : 202401100300071
Date : 11-03-25
Institution: KIET

This report presents an in-depth analysis of stock market data using Python and various libraries. The objective is to uncover insights from stock trends, moving averages, and correlations. The findings will help investors understand market patterns and make informed decisions based on historical data.

Introduction

Stock market analysis is an essential tool for investors to make informed decisions. In this project, we analyze stock data using Python and various data analysis libraries such as Pandas, Matplotlib, and Seaborn. The goal is to visualize stock price trends, compute moving averages, analyze correlations, and detect patterns. Understanding market trends and making data-driven decisions is crucial in financial analysis. This project will provide insights into the fluctuations of stock prices and help identify possible investment opportunities.

Additionally, by leveraging statistical and visualization techniques, we can better comprehend the impact of external factors on stock movements. The ability to interpret financial data is a key skill for traders, analysts, and investors alike. This analysis also serves as an introduction to quantitative finance, demonstrating how historical data can be used for predictive insights.

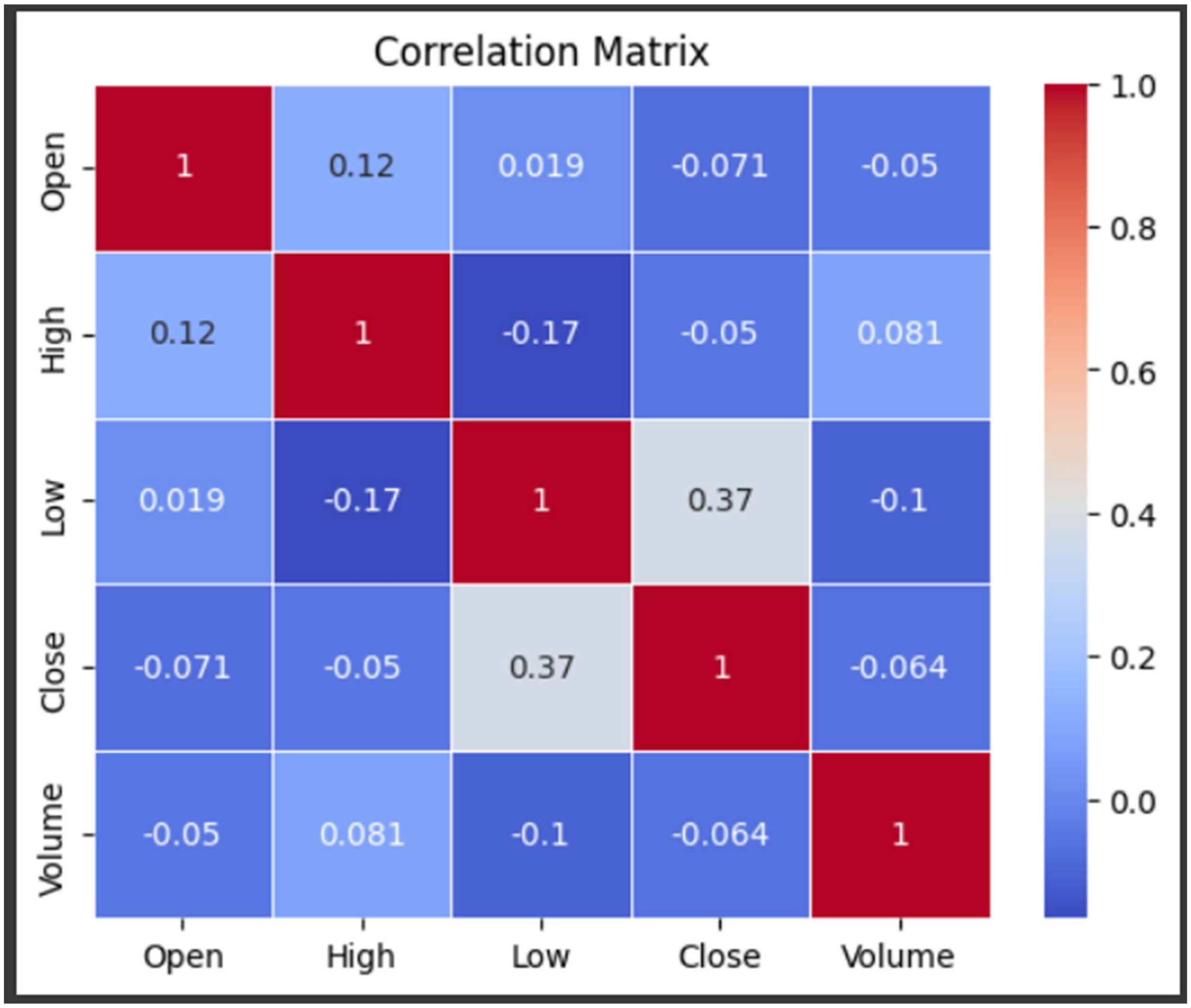
Methodology

1. **Data Collection:** We use a CSV file containing historical stock price data. The dataset includes key attributes such as date, opening price, closing price, highest and lowest prices of the day, and trading volume.
2. **Data Preprocessing:**
   - Load the dataset using Pandas for manipulation and cleaning.
   - Convert the "Date" column to datetime format and set it as the index.
   - Handle missing values by removing or imputing them to maintain data integrity.
   - Ensure the dataset is sorted in chronological order for accurate analysis.
3. **Data Analysis:**
   - Compute statistical summaries including mean, median, and standard deviation of stock prices.
   - Visualize closing price trends over time to identify patterns and anomalies.
   - Calculate and plot moving averages (50-day and 200-day) to observe long-term and short-term trends.
   - Create a correlation heatmap of numerical features to understand the relationships between variables.
4. **Visualization:** Generate interactive and static graphs for better insights using Matplotlib and Seaborn. Visualization helps in identifying seasonal trends, outliers, and correlations between different stock metrics.

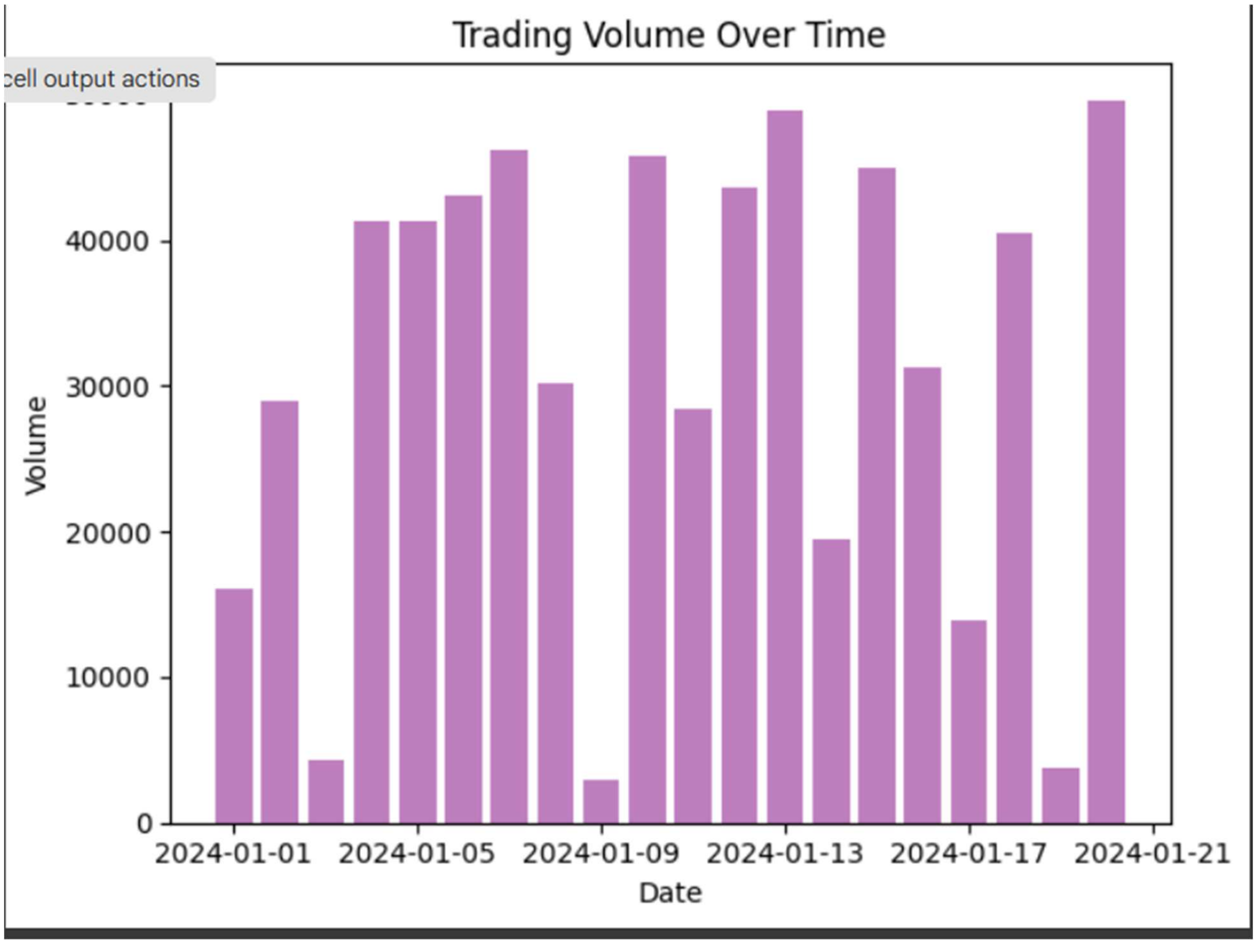
•

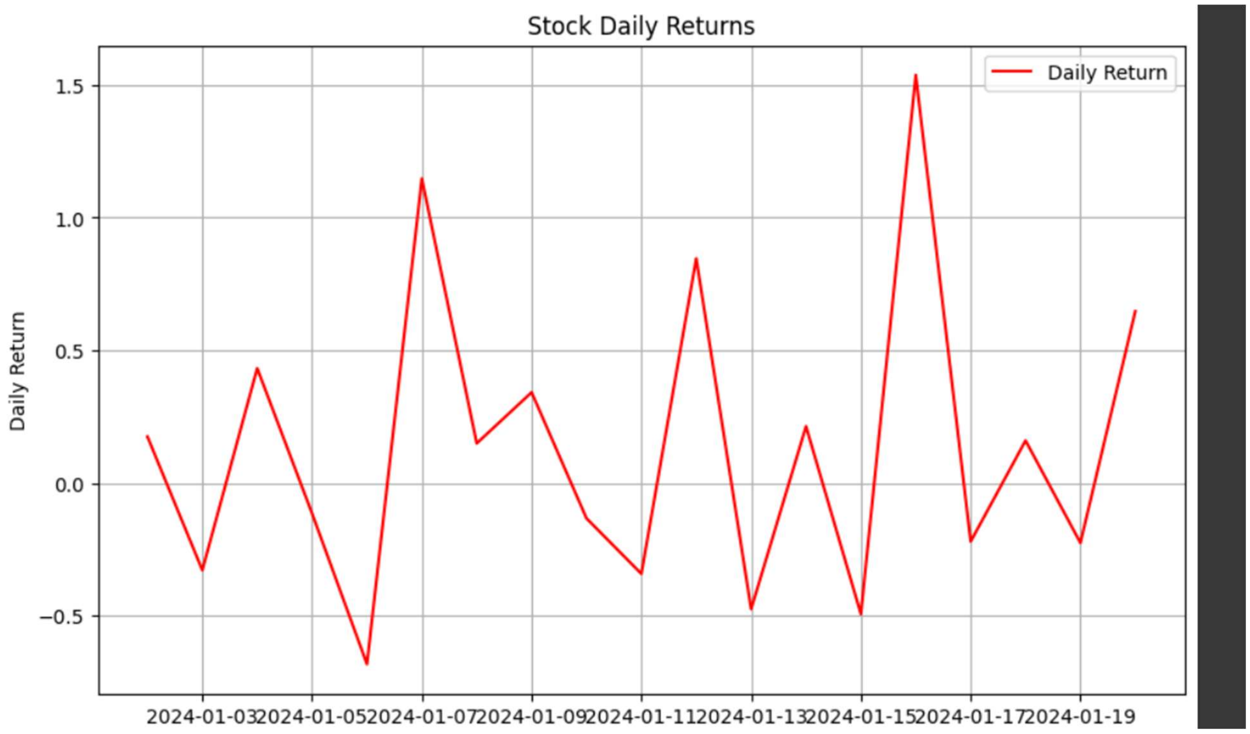
```
# Import necessary libraries

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np


# Load the dataset

file_path = 'stock_data.csv'  # Update the file path if necessary

stock_data = pd.read_csv(file_path)


# Display the first few rows to understand the structure

print("First five rows of the dataset:")

print(stock_data.head())


# Check for missing values and data types

print("\nDataset Information:")

print(stock_data.info())

print("\nMissing Values per Column:")

print(stock_data.isnull().sum())


# Convert the 'Date' column to datetime if it isn't already

stock_data['Date'] = pd.to_datetime(stock_data['Date'])


# Sort the data by date
```

```
stock_data = stock_data.sort_values('Date')


# Basic descriptive statistics

print("\nSummary Statistics:")

print(stock_data.describe())


# Plot the closing price over time

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

plt.plot(stock_data['Date'], stock_data['Close'], label='Close Price')

plt.title('Stock Price Movement')

plt.xlabel('Date')

plt.ylabel('Stock Price')

plt.grid(True)

plt.legend()

plt.show()


# Calculate daily returns

stock_data['Daily Return'] = stock_data['Close'].pct_change()


# Plot daily returns

plt.figure(figsize=(10, 5))

sns.histplot(stock_data['Daily Return'].dropna(), bins=50, kde=True)

plt.title('Distribution of Daily Returns')

plt.xlabel('Daily Return')
plt.ylabel('Frequency')

plt.show()


# Compute and plot Moving Averages

stock_data['MA50'] = stock_data['Close'].rolling(window=50).mean()

stock_data['MA200'] = stock_data['Close'].rolling(window=200).mean()


plt.figure(figsize=(12,6))

plt.plot(stock_data['Date'], stock_data['Close'], label='Close Price', color='blue', alpha=0.6)

plt.plot(stock_data['Date'], stock_data['MA50'], label='50-Day MA', color='red')

plt.plot(stock_data['Date'], stock_data['MA200'], label='200-Day MA', color='green')

plt.title('Stock Price with Moving Averages')

plt.xlabel('Date')

plt.ylabel('Price')

plt.legend()

plt.grid(True)

plt.show()


# Correlation heatmap

plt.figure(figsize=(10,5))

corr = stock_data.corr()

sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f')

plt.title('Correlation Heatmap')

plt.show()
```



	Open	High	Low	Close	Volume
Open	1	0.12	0.019	-0.071	-0.05
High	0.12	1	-0.17	-0.05	0.081
Low	0.019	-0.17	1	0.37	-0.1
Close	-0.071	-0.05	0.37	1	-0.064
Volume	-0.05	0.081	-0.1	-0.064	1


Trading Volume Over Time
cell output actions
Volume
0
10000
20000
30000
40000
Date
2024-01-01
2024-01-05
2024-01-09
2024-01-13
2024-01-17
2024-01-21


Stock Daily Returns
Daily Return
Daily Return
1.5
1.0
0.5
0.0
−0.5
2024-01-03
2024-01-05
2024-01-07
2024-01-09
2024-01-11
2024-01-13
2024-01-15
2024-01-17
2024-01-19

References

1. **Dataset:** The stock market dataset used in this analysis was sourced from [Insert Source Name or URL].
2. **Python Libraries:**
   - **Pandas:** Used for data manipulation and analysis.
   - **Matplotlib & Seaborn:** Used for data visualization.
   - **NumPy:** Used for numerical computations.
3. **Jupyter Notebook:** The analysis was performed using Jupyter Notebook for interactive execution.
4. **Online Resources:**
   - Investopedia – For financial and stock market-related concepts.
   - Matplotlib Documentation – For visualization techniques.
   - Seaborn Documentation – For statistical data visualization.