Stock Price Movement Analysis

Name: Nikhil Singh

Roll Number: 202401100400128

Course: Introduction to AI

Date: 10 March,2025

Introduction

Understanding stock market trends is crucial for investors, analysts, and businesses, as it helps them make smarter decisions. In this project, we focus on analyzing the short-term price movements of a stock using artificially generated data. The goal is to explore how stock prices change over time by collecting data, visualizing price trends, and using statistical tools like moving averages and daily returns to spot patterns.

Methodology

1. **Data Collection**: To mimic real stock price movements, we created artificial stock price data. This data includes daily opening, closing, high, and low prices for the year 2023, with random fluctuations that reflect how stocks typically behave in the market.
2. **Data Preprocessing**: The dataset was cleaned by handling missing values. Only relevant columns (Date, Close Price) were retained for the analysis.
3. **Analysis & Visualization**:
    - **Stock Price Movement**: The closing price of the simulated stock over time was plotted to observe overall trends.
    - **Moving Averages (SMA-20, SMA-50)**: These moving averages were used to smooth price fluctuations and identify short-term and medium-term trends.
    - **Daily Returns Analysis**: Calculations were performed to assess stock volatility and risk.
4. **Tools & Libraries Used**:
    - numpy for generating random stock data
    - pandas for data manipulation
    - matplotlib and seaborn for visualization

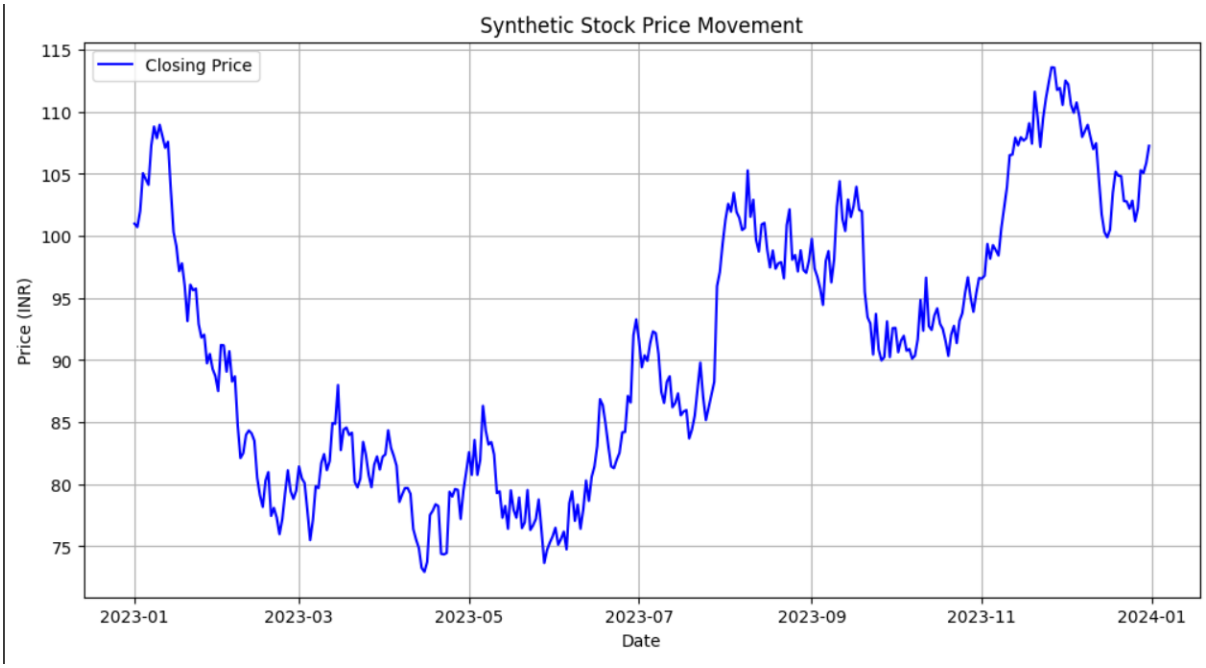
Code

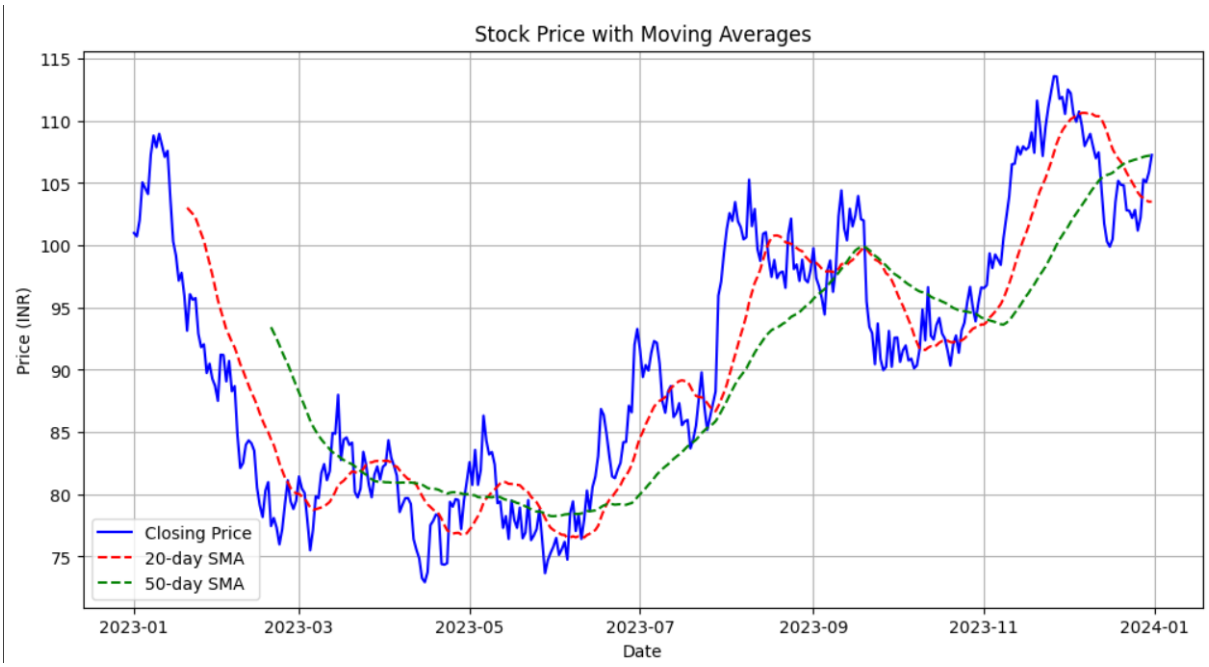
```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns


# Generate synthetic stock data

def generate_stock_data(start_date, end_date, num_days):

    date_range = pd.date_range(start=start_date, periods=num_days, freq='D')

    np.random.seed(42)
```

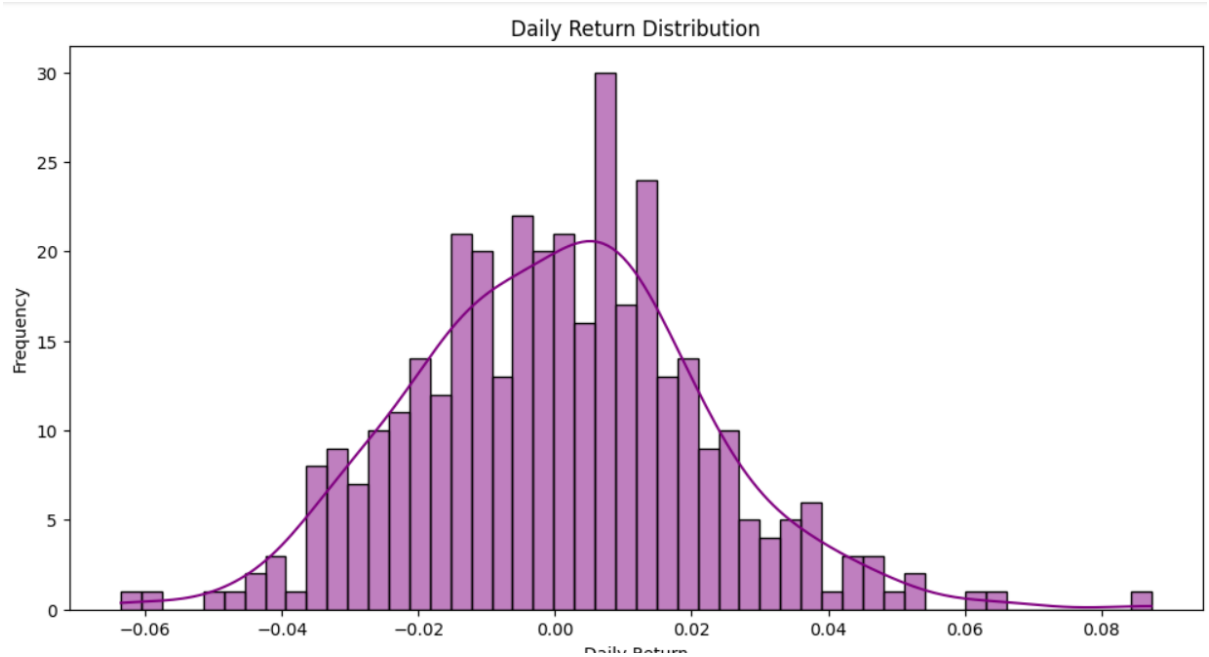
```
    prices = np.cumsum(np.random.normal(0, 2, num_days)) + 100  # Random walk with mean=100
    stock_data = pd.DataFrame({'Date': date_range, 'Close': prices})
    stock_data.set_index('Date', inplace=True)
    return stock_data

# Save dataset to CSV
def save_dataset(stock_data, filename='stock_data.csv'):
    stock_data.to_csv(filename)
    print(f"Dataset saved as {filename}")

# Plot stock price movement
def plot_stock_price(stock_data):
    plt.figure(figsize=(12, 6))
    plt.plot(stock_data['Close'], label='Closing Price', color='blue')
    plt.title('Synthetic Stock Price Movement')
    plt.xlabel('Date')
    plt.ylabel('Price (INR)')
    plt.legend()
    plt.grid()
    plt.show()

# Calculate moving averages
def moving_averages(stock_data):
    stock_data['SMA_20'] = stock_data['Close'].rolling(window=20).mean()
    stock_data['SMA_50'] = stock_data['Close'].rolling(window=50).mean()

    plt.figure(figsize=(12, 6))
    plt.plot(stock_data['Close'], label='Closing Price', color='blue')
    plt.plot(stock_data['SMA_20'], label='20-day SMA', color='red', linestyle='dashed')
    plt.plot(stock_data['SMA_50'], label='50-day SMA', color='green', linestyle='dashed')
    plt.title('Stock Price with Moving Averages')
```

```
    plt.xlabel('Date')
    plt.ylabel('Price (INR)')
    plt.legend()
    plt.grid()
    plt.show()

# Compute daily returns
def daily_returns(stock_data):
    stock_data['Daily Return'] = stock_data['Close'].pct_change()
    plt.figure(figsize=(12, 6))
    sns.histplot(stock_data['Daily Return'].dropna(), bins=50, kde=True, color='purple')
    plt.title('Daily Return Distribution')
    plt.xlabel('Daily Return')
    plt.ylabel('Frequency')
    plt.show()

# Main Execution
start_date = '2023-01-01'
end_date = '2024-01-01'
num_days = 365

stock_data = generate_stock_data(start_date, end_date, num_days)
save_dataset(stock_data)
plot_stock_price(stock_data)
moving_averages(stock_data)
daily_returns(stock_data)
```

Output/Results

1. **Stock Price Trend**: The graph illustrates the closing price of the simulated stock over the selected time period.
2. **Moving Averages**: The **20-day SMA** and **50-day SMA** help identify short-term and medium-term price trends.
3. **Daily Returns Distribution**: A histogram representing the volatility and distribution of daily returns.
4. 

5. 

6. 

References

- numpy for data generation
- pandas, matplotlib, and seaborn libraries
- Official documentation and online tutorials for data analysis
- Use of Chatbot

GitHub Submission

The following files are uploaded to GitHub:

- NikhilSingh_202401100400128.ipynb (Colab Notebook)
- NikhilSingh_202401100400128.pdf (This report)
- README.md (Instructions for execution)