**Capstone Project: Python**

**Stock Market Performance Analysis**

-Submitted by

Anamika Sinha

Table of Content:

* Introduction……………………………………………………..03
* Data Collection…………………………………………………04
* Data Pre-processing………………………………………….05
* Exploratory Data Analysis (EDA)……………………….06
* Technical Analysis……………………………………………..09
* Time Series Forecasting…………………………………….11
* Portfolio Optimization……………………………………….12
* Trading Strategy………………………………………………..13
* Conclusion………………………………………………………...14
* Challenges & Future Work…………………………………14
* Appendix……………………………………………………………15

**Introduction:**

Stock market performance analysis can be used to inform investment decisions and help investors make informed decisions about buying or selling stocks. Suppose you work as a data science professional in a company that provides services based on investment decisions. As a data science professional, you can help your business by analysing the historical performance of different companies, identifying potential opportunities and risks in the stock market, and adjusting your clients’ investment strategies accordingly.

As a data science professional, I will go through a structured process of stock market performance analysis, which involves collecting historical stock price data of different companies from trusted sources such as Yahoo Finance, visualizing data using various charts, calculating movements, averages and volatility for each company, and performing correlation analysis to analyse the relationships between different stock prices.

In the section below, I will take you through the task of Stock Market Performance Analysis using Python step by step.

**Background on stock markets:**

Stock markets play a pivotal role in the global economy. They represent a place where companies can raise capital by issuing shares to the public, and investors can buy or sell these shares. Understanding stock markets is essential for anyone interested in investment or finance.

**Objective and scope:**

In this project, I aim to harness the power of Python to analyse stock market performance. By the end, I'll understand how stock prices move, identify trends, and even develop potential investment strategies.

Let’s start the task of Stock Market Performance Analysis by importing the necessary Python libraries and the dataset. For this task, I will use the Yahoo finance API (yfinance) to collect real-time stock market data for the past three months.

It’s important to collect real-time data for this task, but still, if you are a complete beginner and want a dataset only to practice the concepts covered in this article, you can download the dataset from here. But it’s recommended to use the yfinance API to collect and work on real-time data.

You can install the yfinace API in your Python environment using the pip command mentioned below (run the command below on your command prompt or terminal):

|  |
| --- |
| for command prompt or terminal: pip install yfinance for Google Colab or Jupyter notebooks: !pip install yfinance |

**Data Collection**

Fetching stock data:

To start any stock market analysis, we need data. Python offers several libraries to fetch this data seamlessly.

**Installing necessary libraries:**

To begin, we need to install yfinance and pandas\_datareader. These libraries allow us to pull stock data easily.

|  |
| --- |
| !pip install yfinance pandas\_datareader |

**Fetching historical data:**

yfinance makes it easy to get stock data. In the example below, we're fetching data for Apple Inc. (AAPL) for the past year.

#Fetching 1 year historical data of Apple stock via yfinance

import pandas as pd

import matplotlib.pyplot as plt

import yfinance as yf

stock = "AAPL" # Using Apple's stock symbol

data = yf.Ticker(stock).history(period="1y") # Fetching data for 1 year

**Choosing stocks for analysis:**

For a comprehensive analysis, it's advisable to select stocks from various sectors. For instance, consider tech stocks (AAPL, MSFT), financials (JPM), and consumer goods (PEP). This diversification gives a broader view of the market.

**Data Pre-processing**

Before diving into analysis, we need to ensure our data is clean and in the right format.

**Handling missing values:**

It's common to encounter missing values in stock data. We need to address these to avoid errors in our analysis.

A screenshot of a computer

Description automatically generated

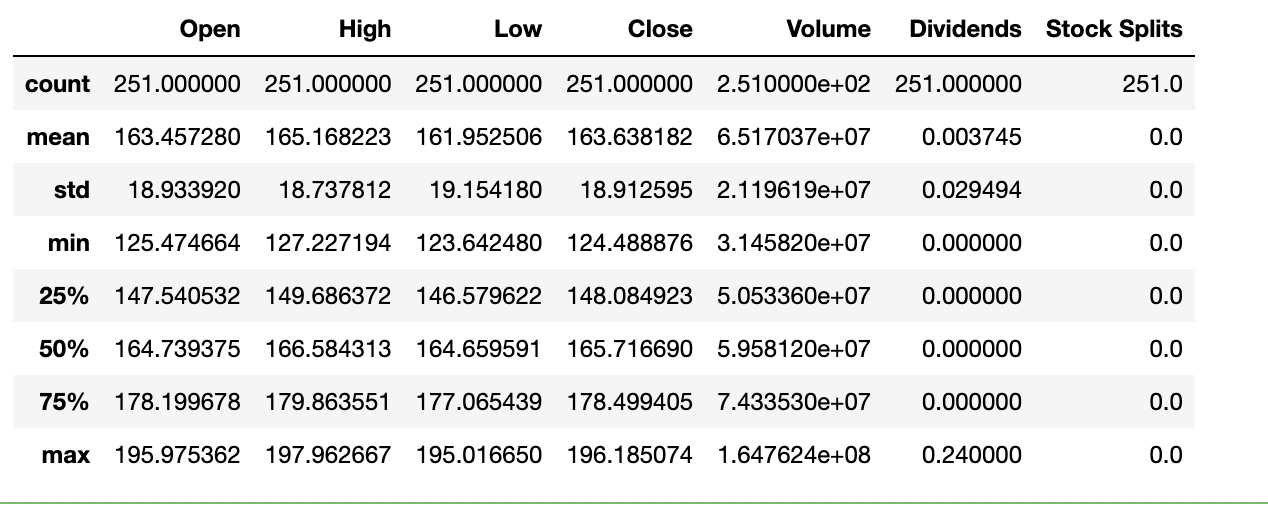
#Deep Copying data into another variable df to keep the original “data” unchanged

df = data.copy()

#Summary of the stock’s historical data

df.head()

df.describe()



#cleaning data to handle missing values

df.dropna(inplace=True) # This removes any rows with missing values

**Ensure date format is correct:**

For time series analysis, it's crucial that our date column is in the datetime format.

#converting df index to datetime format

df.index = pd.to\_datetime(df.index)

**Calculate daily returns:**

Daily returns provide insights into the stock's volatility. It's calculated as the percentage change in the stock's price from the previous day.

#adding a new column “Daily Return”which the calculated percentage change of the stock #price over previous day

df['Daily Return'] = df['Close'].pct\_change()

**Exploratory Data Analysis (EDA)**

EDA helps us understand the nature and structure of our data. It's the first step in identifying trends or anomalies.

Visualize stock price movements:

A simple line plot can show us how the stock's closing price has moved over time.

#plotting Stock Price Movement; We can observe a net growth in the stock price in the long run

df['Close'].plot(figsize=(12, 6), title="Stock Price Movement")

plt.xlabel('Date')

plt.ylabel('Stock Closing Price')

plt.show()

A graph showing a line

Description automatically generated

**Analyze volume of stocks traded:**

Volume indicates the number of shares traded in a given period. High volume can suggest significant news or events affecting the stock.

#Plotting Volume Traded over time; we can observe patterns of sudden peaks(highs) #immediately followed by sharp valleys(lows). This may be indicative of Market #manipulation schemes like “Pump and Dump”

df['Volume'].plot(figsize=(12, 6), title="Volume Traded Over Time")

plt.xlabel('Date')

plt.ylabel('Stock Traded Volume')

plt.show()

A graph showing a number of numbers

Description automatically generated

**Correlation between stocks:**

When analysing multiple stocks, it's essential to see how they move in relation to one another. A heatmap can visualize this relationship.

#importing seaborn – a python based data visualization library

import seaborn as sns

#For a comprehensive analysis, selecting stocks from various sectors. For instance, #considering tech stocks (AAPL, MSFT), financials (JPM), and consumer goods (PEP). This #diversification will give us a broader view of the market.

stock\_list = ["AAPL", "MSFT", "JPM", "PEP", "GOOGL"]

close\_prices = pd.DataFrame()

#looping through stock list to fetch 1 year historical stock data for each stock in the list

for stock in stock\_list:

close\_prices[stock] = yf.Ticker(stock).history(period="1y")['Close']

#compute correlation between the stocks to see how they move in relation to one another.

#Insight: Strong positive correlation between (GOOGLE, MSFT),

#(AAPL, MSFT) and (APPL, GOOGL),

#Weak correlation between (PEP, JPM), (PEP, APPL) and (JPM, MSFT) etc

correlation = close\_prices.pct\_change().corr()

#plotting a heatmap to visualize the relationship

sns.heatmap(correlation, annot=True, cmap='coolwarm')

plt.show()

*Note: Correlation values range between -1 and 1. A value closer to 1 implies that two stocks move in tandem, while a value closer to -1 indicates they move in opposite directions.*



**Technical Analysis**

Technical analysis involves studying past market data, primarily price and volume, to forecast future price movements. This analysis can be performed on any security with historical trading data.

Moving Averages:

A moving average smoothens price data to create a single flowing line, which makes it easier to identify the direction of the trend. The two most common types of moving averages are the Simple Moving Average (SMA) and the Exponential Moving Average (EMA).

#Simple Moving Average - smoothens price data to create a single flowing line, which makes #it easier to identify the direction of the trend. The two most common types of moving #averages are the Simple Moving Average (SMA) and the Exponential Moving Average #(EMA).

#adding Simple Moving Average Columns – 50 day & 200 day

df['SMA50'] = df['Close'].rolling(window=50).mean()

df['SMA200'] = df['Close'].rolling(window=200).mean()

#plotting SMA50, SMA200 & stock price vs time

df[['Close', 'SMA50', 'SMA200']].plot(figsize=(12,6))

plt.title('Simple Moving Average')

plt.show()

A graph with blue and orange lines

Description automatically generated

*Note: Here, SMA50 is the 50-day moving average, and SMA200 is the 200-day moving average. When SMA50 crosses above SMA200, it's often seen as a bullish sign, and vice versa.*

**Exponential Moving Average (EMA):**

#adding Exponential Moving Average Columns – 50 day & 200 day

df['EMA50'] = df['Close'].ewm(span=50, adjust=False).mean()

df['EMA200'] = df['Close'].ewm(span=200, adjust=False).mean()

#plotting EMA50, EMA200 & closing stock price vs time

#Insights: We can observe that just before March 2023, the EMA50 line crosses above #EMA200, which is often seen as bullish trend. The stock price data also seems to agree with #the prediction as can be seen in the price uptrend.

df[['Close', 'EMA50', 'EMA200']].plot(figsize=(12,6))

plt.title('Exponential Moving Average')

plt.show()

A graph of a graph showing the growth of a stock market

Description automatically generated with medium confidence

*Note: EMA gives more weight to recent prices, and therefore reacts more quickly to price changes than SMA.*

**Bollinger Bands:**

Bollinger Bands consist of a middle band being an N-period simple moving average (SMA), an upper band at K times an N-period standard deviation above the middle band, and a lower band at K times an N-period standard deviation below the middle band.

#Insights: We can observe that the stock price touches the lower band twice between Dec #2022 – Jan 2023 and the upper band post that around Feb 2023. The Volume Traded over #time plot also seems to agree with it. If a stock is overbought, it means that its price has #risen significantly in a short time period and vice versa. Also, it usually indicates a future #price reversal; however an immediate one is not guaranteed.

#Adding SMA20 column – middle band

df['SMA20'] = df['Close'].rolling(window=20).mean()

#Adding Upper column – upper band

df['Upper'] = df['SMA20'] + 2\*df['Close'].rolling(window=20).std()

#Adding Lower column – lower band

df['Lower'] = df['SMA20'] - 2\*df['Close'].rolling(window=20).std()

#plotting Bollinger bands

df[['Close', 'SMA20', 'Upper', 'Lower']].plot(figsize=(12,6))

plt.title('Bollinger Bands')

plt.show()

A graph of a line graph

Description automatically generated with medium confidence

*Note: Bollinger Bands are used to determine overbought and oversold levels; when the price reaches the upper band, it might be overbought, and when it reaches the lower band, it might be oversold.*

**Time Series Forecasting**

Time series forecasting involves predicting future values based on previously observed values.

**ARIMA Model:**

ARIMA stands for AutoRegressive Integrated Moving Average. It is a forecasting method for univariate time series data.

#forecating AAPL Stock Price for next 100 days

#importing the ARIMA python library

from statsmodels.tsa.arima.model import ARIMA

model = ARIMA(df['Close'], order=(5,1,0))

model\_fit = model.fit()

forecast\_steps = 100

forecast = model\_fit.get\_forecast(steps=forecast\_steps, dynamic=True)

# Plot the original time series data and the forecasted values

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

plt.plot(df["Close"], label='Closing Stock Price')

plt.plot(pd.date\_range(df["Close"].index[-1], periods=forecast\_steps, freq='D'), forecast.predicted\_mean, label='Forecasted Stock price')

plt.title('Time Series Data with ARIMA Forecast')

plt.legend()

plt.show()

A graph showing a line

Description automatically generated with medium confidence

**Portfolio Optimization**

When dealing with multiple stocks, it's crucial to determine how much of each stock you should hold in your portfolio.

**Efficient Frontier:**

This is a concept from Modern Portfolio Theory. The efficient frontier represents the set of optimal portfolios that offer the highest expected return for a specific level of risk.

#Portfolio Optimization - to determine how much of each stock you should hold in your portfolio, when dealing with multiple stocks

#Insights: Portfolio Optimization suggests that we should include ˜0.41 stocks of MSFT, ˜0.47 stocks of JPM and 0.12 stocks of GOOGL and not include AAPL & PEP stocks in our portfolio

#Importing minimize function from scipy python library

from scipy.optimize import minimize

# Assuming stock\_list contains the symbols of the stocks in the portfolio

# calculating stock closing price percentage change over previous trading day

returns = close\_prices.pct\_change()

mean\_returns = returns.mean()

cov\_matrix = returns.cov()

#risk appetite

risk\_free\_rate = 0.0178

# Portfolio optimization function

def portfolio\_annualised\_performance(weights, mean\_returns, cov\_matrix):

# 251 trading days

returns = np.sum(mean\_returns\*weights ) \*251

std = np.sqrt(np.dot(weights.T, np.dot(cov\_matrix, weights))) \* np.sqrt(251)

return std, returns

# Minimize negative Sharpe Ratio to get optimal portfolio

def neg\_sharpe\_ratio(weights, mean\_returns, cov\_matrix, risk\_free\_rate):

p\_var, p\_ret = portfolio\_annualised\_performance(weights, mean\_returns, cov\_matrix)

return -(p\_ret - risk\_free\_rate) / p\_var

# Constraints for optimization

constraints = ({'type': 'eq', 'fun': lambda x: np.sum(x) - 1})

#initial guess of 0.2 qty for each stock

initial = [1./len(stock\_list) for stock in stock\_list]

bounds = tuple((0, 1) for asset in range(len(stock\_list)))

# Running the optimization

result = minimize(neg\_sharpe\_ratio, initial, args=(mean\_returns, cov\_matrix, risk\_free\_rate), bounds=bounds, constraints=constraints)

# Qty of stocks of each type in optimized portfolio

print(result)

print([round(float(r), 2) for r in result.x])

A screenshot of a math problem

Description automatically generated

*Note: The above is a simple example of portfolio optimization. There are many other factors and constraints that can be considered.*

**Conclusion**

Summarize the insights gathered from the analysis:

* Stock price trends over the past year.
* Technical indicators and their implications.
* Potential investment strategies based on historical data.
* Performance of the developed trading strategy.

Remember, the stock market is influenced by a myriad of factors, including economic data, corporate earnings reports, geopolitical events, and others. While historical data provides valuable insights, it's essential to stay updated with current events and adjust strategies accordingly.

**Challenges & Future Work**

Discuss the challenges encountered during the project:

* Data inconsistencies or missing values.
* Assumptions made during the analysis and their implications.
* Limitations of the chosen models and strategies.

**Suggest avenues for future exploration:**

Incorporating more sophisticated models like neural networks for stock price prediction.

Exploring algorithmic trading strategies using tools like backtrader or QuantConnect.

Integrating real-time data feeds for live trading.

**External Resources**

**Websites:**

1. [Investopedia](https://www.google.com/url?q=https://www.investopedia.com/&sa=D&source=editors&ust=1698478990061521&usg=AOvVaw34m2y-8i7NbPkivQ5V28Xj): A comprehensive resource for all things related to finance and investing. It provides detailed explanations of financial terms and concepts.
2. [Yahoo Finance](https://www.google.com/url?q=https://finance.yahoo.com/&sa=D&source=editors&ust=1698478990062045&usg=AOvVaw26vIfdkmLTxQ8z-LeVF37V): A widely-used platform for checking stock prices, financial news, and historical data. It also offers educational articles and videos.
3. [StockCharts.com](https://www.google.com/url?q=https://stockcharts.com/&sa=D&source=editors&ust=1698478990062329&usg=AOvVaw2ZSnpi3-nM-7czVmiPEETb): A resource for technical analysis enthusiasts. It offers tutorials, articles, and tools for charting and technical analysis.
4. [Modern Portfolio Theory (Investopedia)](https://www.google.com/url?q=https://www.investopedia.com/terms/m/modernportfoliotheory.asp&sa=D&source=editors&ust=1698478990062602&usg=AOvVaw37EukH6eHv5OYTNDHHMHQG): Learn more about the concept of the Efficient Frontier and Modern Portfolio Theory.

**YouTube Channels:**

1. [The Plain Bagel](https://www.google.com/url?q=https://www.youtube.com/c/ThePlainBagel&sa=D&source=editors&ust=1698478990063061&usg=AOvVaw0zK7YGxL7yQh-gkNbDURVw): Provides clear explanations of finance and investing concepts, including topics related to stock market analysis.
2. [Investopedia](https://www.google.com/url?q=https://www.youtube.com/user/investopediacom&sa=D&source=editors&ust=1698478990063358&usg=AOvVaw2ZeR57win0j6l4Z7p_5Hik): The official YouTube channel of Investopedia features video explanations of financial terms, strategies, and concepts.
3. [UKspreadbetting](https://www.google.com/url?q=https://www.youtube.com/user/ukspreadbetting&sa=D&source=editors&ust=1698478990063631&usg=AOvVaw1Dg_4J2F8jX4y_BNFtyzwu): Offers educational content on trading strategies, technical analysis, and market psychology.

**Videos:**

1. [How the Stock Market Works (Investopedia Video)](https://www.google.com/url?q=https://www.youtube.com/watch?v%3D7aJ2y7wWVdU&sa=D&source=editors&ust=1698478990064075&usg=AOvVaw2sFa2yfFNZm0GQYrtpY6YK): An introductory video explaining the basics of the stock market.
2. [What is Technical Analysis? (Investopedia Video)](https://www.google.com/url?q=https://www.youtube.com/watch?v%3DjT-sEzHTTto&sa=D&source=editors&ust=1698478990064427&usg=AOvVaw3v8dvbvjYhxc1Y1ZUAQoLS): Provides an overview of technical analysis and its key components.
3. [Introduction to ARIMA Time Series Forecasting (Data School)](https://www.google.com/url?q=https://www.youtube.com/watch?v%3De8Yw4alG16Q&sa=D&source=editors&ust=1698478990064706&usg=AOvVaw1J4oOXq1rQySKwuH9i0Vj1): A detailed video tutorial on ARIMA modelling for time series forecasting.
4. [LSTM Time Series Prediction Example with Keras in Python (Machine Learning Mastery)](https://www.google.com/url?q=https://www.youtube.com/watch?v%3DftMq5ps503w&sa=D&source=editors&ust=1698478990065121&usg=AOvVaw1Pw1j1YKRcGgwseU_z1XsK): A practical guide to implementing LSTM for time series prediction.