Quantitative Analysis of Stock Market using Python

AMAN KHARWAL · JANUARY 15, 2024 · 4 COMMENTS

Quantitative Analysis in the stock market involves the use of mathematical and statistical techniques to understand, predict, and make decisions about financial investments. If you want to learn how to perform Quantitative Analysis on stock market data, this article is for you. In this article, I'll take you through the task of Quantitative Analysis of the stock market using Python.

Quantitative Analysis of Stock Market: Process We Can Follow

Quantitative Analysis in the stock market is a financial methodology that utilizes mathematical and statistical techniques to analyze stocks and financial markets.

Below is the process we can follow for the task of Quantitative Analysis of the stock market:

- Clearly define the objectives and questions to be answered.
- Identify the key performance indicators (KPIs) relevant to the analysis.
- Gather historical stock market data, including prices, volumes, and other relevant financial indicators.
- Clean and preprocess the data to handle missing values, outliers, and errors.
- Conduct initial analysis to understand data distributions, patterns, and correlations.
- Implement various strategies based on quantitative analysis.

To get started with this task, we need appropriate data. I found an ideal dataset for this task. You can download the dataset from **here**.

Quantitative Analysis of Stock Market using Python

Now, let's get started with the task of Quantitative Analysis of the stock market by importing the necessary Python libraries and the **dataset**:

```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import plotly.io as pio
pio.templates.default = "plotly_white"

# Load the dataset
stocks_data = pd.read_csv("stocks.csv")

# Display the first few rows of the dataset
print(stocks_data.head())
```

```
  Ticker        Date        Open        High         Low       Close  \
0   AAPL  2023-02-07  150.639999  155.229996  150.639999  154.649994
1   AAPL  2023-02-08  153.880005  154.580002  151.169998  151.919998
2   AAPL  2023-02-09  153.779999  154.330002  150.419998  150.869995
3   AAPL  2023-02-10  149.460007  151.339996  149.220001  151.009995
4   AAPL  2023-02-13  150.949997  154.259995  150.919998  153.850006

    Adj Close    Volume
0  154.414230  83322600
1  151.688400  64120100
2  150.639999  56007100
3  151.009995  57450700
4  153.850006  62199000
```

The dataset contains the following columns for stock market data:

- Ticker: The stock ticker symbol.
- Date: The trading date.
- Open: The opening price of the stock for the day.

- High: The highest price of the stock during the day.
- Low: The lowest price of the stock during the day.
- Close: The closing price of the stock for the day.
- Adj Close: The adjusted closing price, which accounts for all corporate actions such as dividends, stock splits, etc.
- Volume: The number of shares traded during the day.

To perform a quantitative analysis, we can explore various statistical concepts like descriptive statistics, **time series analysis**, correlation analysis, and more. Here are some potential analyses we can perform:

1. **Descriptive Statistics**: Summary statistics (mean, median, standard deviation, etc.) for each stock.
2. **Time Series Analysis**: Trends and patterns over time, especially for closing prices.
3. **Volatility Analysis**: How much the stock price fluctuates over a period.
4. **Correlation Analysis**: How stock prices of different companies are related to each other.
5. **Comparative Analysis**: Comparing the performance of different stocks.
6. **Risk-Return Trade-off Analysis**: Analyzing the balance between the potential risks and rewards of different stocks, aiding in portfolio management.

Let's implement all these concepts of Quantitative Analysis of the stock market one by one.

Descriptive Statistics

Descriptive Statistics will provide summary statistics for each stock in the dataset. We'll look at measures such as mean, median, standard deviation, and more for the Close prices:

```
# Descriptive Statistics for each stock
descriptive_stats = stocks_data.groupby('Ticker')['Close']

print(descriptive_stats)
```

```
        count        mean        std         min         25%         50%  \
Ticker
AAPL     62.0  158.240645   7.360485  145.309998  152.077499  158.055000
GOOG     62.0  100.631532   6.279464   89.349998   94.702501  102.759998
MSFT     62.0  275.039839  17.676231  246.270004  258.742500  275.810013
NFLX     62.0  327.614677  18.554419  292.760010  315.672493  325.600006

                75%         max
Ticker
AAPL     165.162506  173.570007
GOOG     105.962503  109.459999
MSFT     287.217506  310.649994
NFLX     338.899994  366.829987
```

Let's break down the results for each stock:

AAPL (Apple Inc.)

- **Count**: 62.0 (The number of observations or trading days included in the dataset for AAPL)
- **Mean**: 158.24 (The average closing price)
- **Standard Deviation**: 7.36 (Measures the amount of variation or dispersion of closing prices)
- **Minimum**: 145.31 (The lowest closing price in the dataset)

- **25th Percentile**: 152.08 (25% of the closing prices are below this value)
- **Median (50%)**: 158.06 (The middle value of the closing prices)
- **75th Percentile**: 165.16 (75% of the closing prices are below this value)
- **Maximum**: 173.57 (The highest closing price in the dataset)

GOOG (Alphabet Inc.)

Similar statistics as AAPL, but for GOOG. The mean closing price is 100.63, with a standard deviation of 6.28, indicating less variability in closing prices compared to AAPL.

MSFT (Microsoft Corporation)

The dataset includes the same number of observations for MSFT. It has a higher mean closing price of 275.04 and a higher standard deviation of 17.68, suggesting greater price variability than AAPL and GOOG.

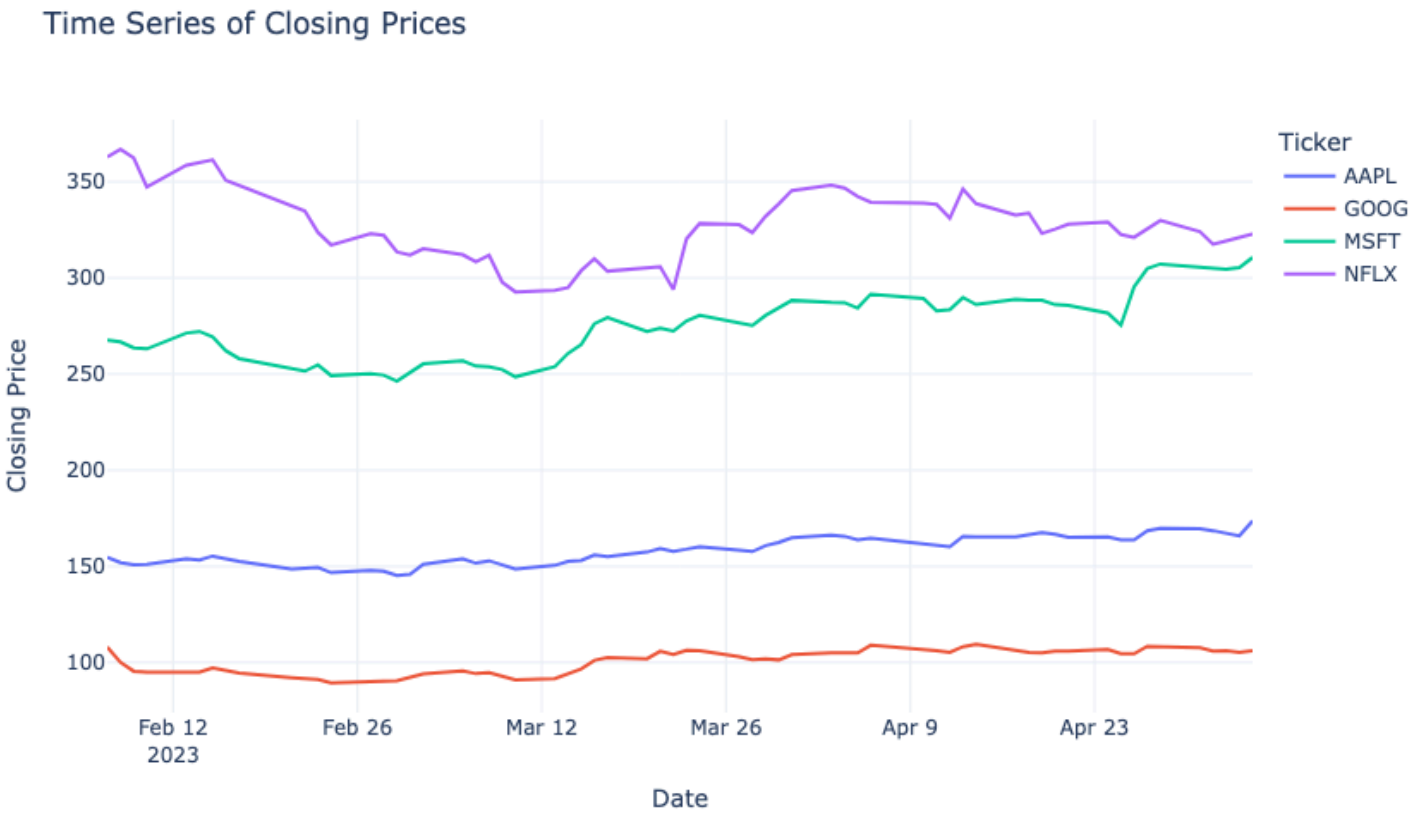
NFLX (Netflix Inc.)

NFLX shows the highest mean closing price (327.61) among these stocks and the highest standard deviation (18.55), indicating the most significant price fluctuation.

Time Series Analysis

Next, we'll proceed with the Time Series Analysis to examine trends and patterns over time, focusing on the closing prices:

```
# Time Series Analysis
stocks_data['Date'] = pd.to_datetime(stocks_data['Date'])
pivot_data = stocks_data.pivot(index='Date', columns='Tick

# Create a subplot
fig = make_subplots(rows=1, cols=1)

# Add traces for each stock ticker
for column in pivot_data.columns:
    fig.add_trace(
        go.Scatter(x=pivot_data.index, y=pivot_data[column
        row=1, col=1
    )

# Update layout
fig.update_layout(
    title_text='Time Series of Closing Prices',
    xaxis_title='Date',
    yaxis_title='Closing Price',
    legend_title='Ticker',
    showlegend=True
)

# Show the plot
fig.show()
```



The above plot displays the time series of the closing prices for each stock (AAPL, GOOG, MSFT, NFLX) over the observed period. Here are some key observations:

1. **Trend**: Each stock shows its unique trend over time. For instance, AAPL and MSFT exhibit a general upward trend in this period.
2. **Volatility**: There is noticeable volatility in the stock prices. For example, NFLX shows more pronounced fluctuations compared to others.
3. **Comparative Performance**: When comparing the stocks, MSFT and NFLX generally trade at higher price levels than AAPL and GOOG in this dataset.

Volatility Analysis

Next, let's focus on Volatility Analysis. We'll calculate and compare the volatility (standard deviation) of the closing prices for each stock. It will give us an insight into how much the stock prices fluctuated over the period:

```
# Volatility Analysis
volatility = pivot_data.std().sort_values(ascending=False)

fig = px.bar(volatility,
             x=volatility.index,
             y=volatility.values,
             labels={'y': 'Standard Deviation', 'x': 'Tick
             title='Volatility of Closing Prices (Standard

# Show the figure
fig.show()
```

The bar chart and the accompanying data show the volatility (measured as standard deviation) of the closing prices for each stock. Here's how they rank in terms of volatility:

1. **NFLX**: Highest volatility with a standard deviation of approximately 18.55.
2. **MSFT**: Next highest, with a standard deviation of around 17.68.
3. **AAPL**: Lower volatility compared to NFLX and MSFT, with a standard deviation of about 7.36.
4. **GOOG**: The least volatile in this set, with a standard deviation of approximately 6.28.

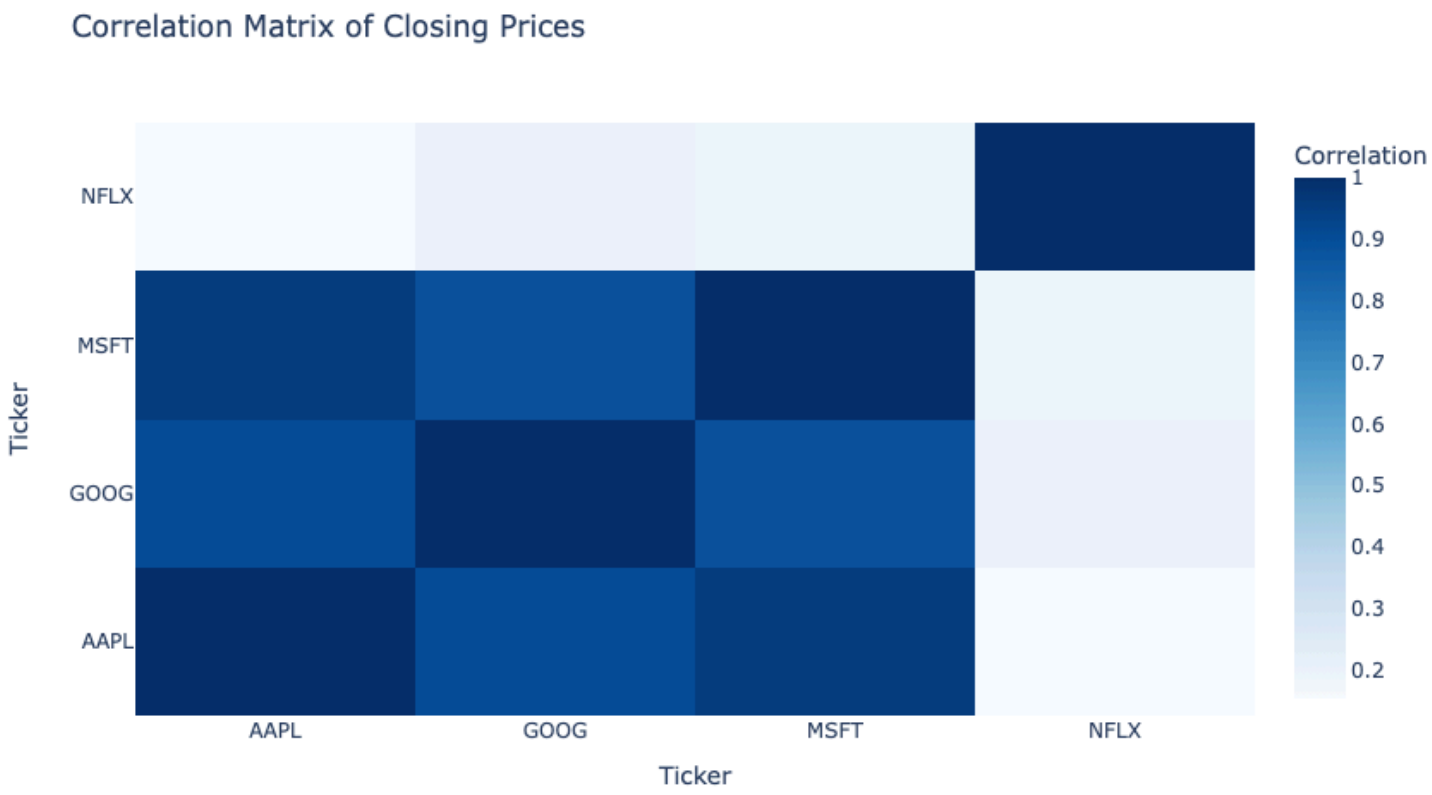
It indicates that NFLX and MSFT stocks were more prone to price fluctuations during this period compared to AAPL and GOOG.

Correlation Analysis

Next, we'll perform a Correlation Analysis to understand how the stock prices of these companies are related to each other:

```
# Correlation Analysis
correlation_matrix = pivot_data.corr()

fig = go.Figure(data=go.Heatmap(
                    z=correlation_matrix,
                    x=correlation_matrix.columns,
                    y=correlation_matrix.columns,
                    colorscale='blues',
                    colorbar=dict(title='Correlation'),
                    ))
```

```
11
12 # Update layout
13 fig.update_layout(
14     title='Correlation Matrix of Closing Prices',
15     xaxis_title='Ticker',
16     yaxis_title='Ticker'
17 )
18
19 # Show the figure
20 fig.show()
```



The heatmap above displays the correlation matrix of the closing prices of the four stocks (AAPL, GOOG, MSFT, NFLX). Here's what the correlation coefficients suggest:

- Values close to +1 indicate a strong positive correlation, meaning that as one stock's price increases, the other tends to increase as well.
- Values close to -1 indicate a strong negative correlation, where one stock's price increase corresponds to a decrease in the other.
- Values around 0 indicate a lack of correlation.

From the heatmap, we can observe that there are varying degrees of positive correlations between the stock prices, with some pairs showing stronger correlations than others. For instance, AAPL and MSFT seem to have a relatively higher positive correlation.

Comparative Analysis

Now, let's move on to Comparative Analysis. In this step, we'll compare the performance of different stocks based on their returns over the period. We'll calculate the percentage change in closing prices from the start to the end of the period for each stock:

```
# Calculating the percentage change in closing prices
percentage_change = ((pivot_data.iloc[-1] - pivot_data.ilo

fig = px.bar(percentage_change,
             x=percentage_change.index,
             y=percentage_change.values,
             labels={'y': 'Percentage Change (%)', 'x': 'T
             title='Percentage Change in Closing Prices')

# Show the plot
fig.show()
```

The bar chart and the accompanying data show the percentage change in the closing prices of the stocks from the start to the end of the observed period:

- **MSFT**: The highest positive change of approximately 16.10%.
- **AAPL**: Exhibited a positive change of approximately 12.23%. It indicates a solid performance, though slightly lower than MSFT's.
- **GOOG**: Showed a slight negative change of about -1.69%. It indicates a minor decline in its stock price over the observed period.
- **NFLX**: Experienced the most significant negative change, at approximately -11.07%. It suggests a notable decrease in its stock price during the period.

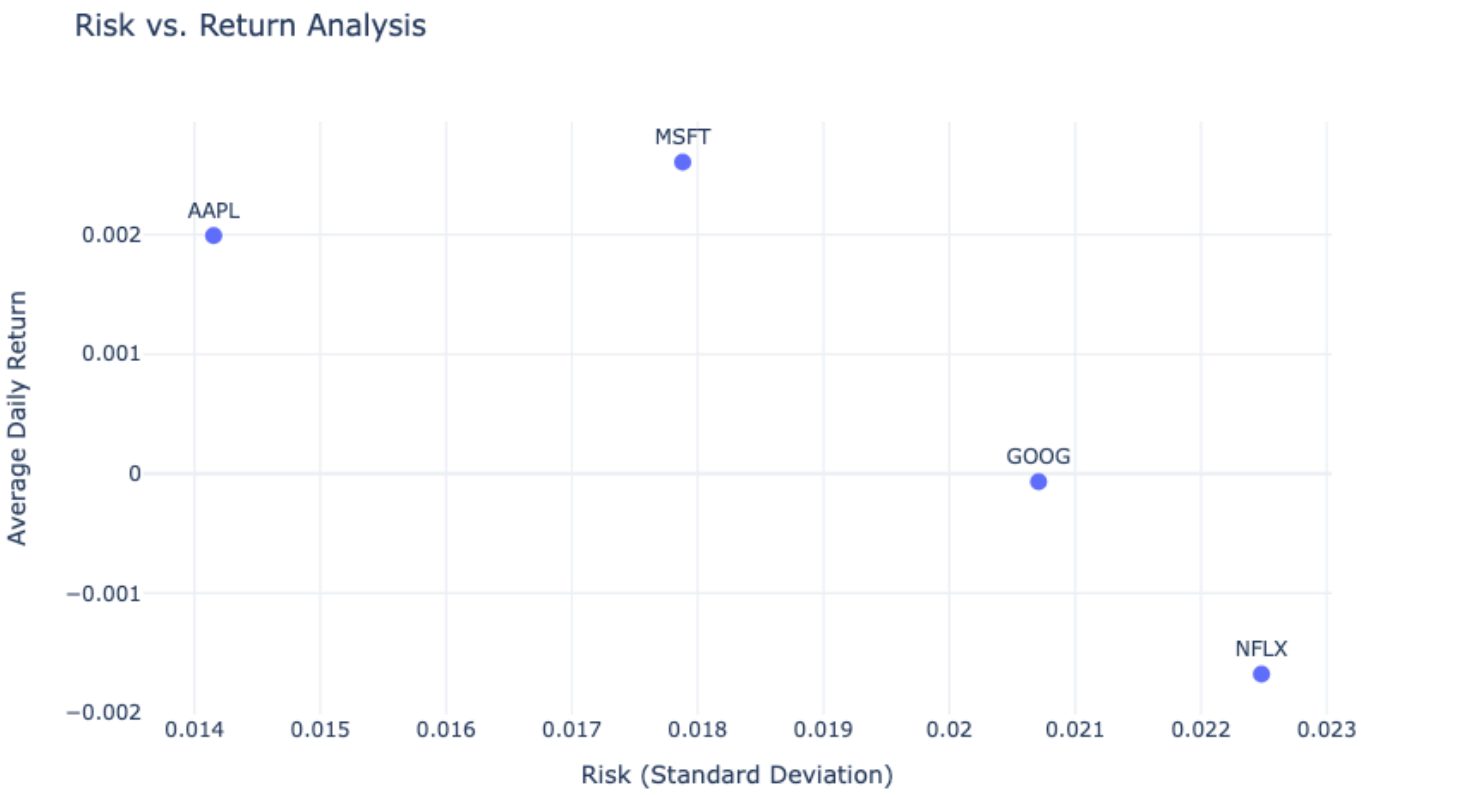
Daily Risk Vs. Return Analysis

To perform a Risk vs. Return Analysis, we will calculate the average daily return and the standard deviation of daily returns for each stock. The standard deviation will serve as a proxy for risk, while the average daily return represents the expected return.

We will then plot these values to visually assess the risk-return profile of each stock. Stocks with higher average returns and lower risk (standard deviation) are generally more desirable, but investment decisions often depend on the investor's risk tolerance:

```
daily_returns = pivot_data.pct_change().dropna()

# Recalculating average daily return and standard deviatio
avg_daily_return = daily_returns.mean()
risk = daily_returns.std()

# Creating a DataFrame for plotting
risk_return_df = pd.DataFrame({'Risk': risk, 'Average Dail

fig = go.Figure()

# Add scatter plot points
fig.add_trace(go.Scatter(
    x=risk_return_df['Risk'],
    y=risk_return_df['Average Daily Return'],
    mode='markers+text',
    text=risk_return_df.index,
    textposition="top center",
    marker=dict(size=10)
))

```

```
# Update layout
fig.update_layout(
    title='Risk vs. Return Analysis',
    xaxis_title='Risk (Standard Deviation)',
    yaxis_title='Average Daily Return',
    showlegend=False
)

# Show the plot
fig.show()
```



So, AAPL shows the lowest risk combined with a positive average daily return, suggesting a more stable investment with consistent returns. GOOG has higher volatility than AAPL and, on average, a slightly negative daily return, indicating a riskier and less rewarding investment during this period.

MSFT shows moderate risk with the highest average daily return, suggesting a potentially more rewarding investment, although with higher volatility compared to AAPL. NFLX exhibits the highest risk and a negative average daily return, indicating it was the most volatile and least rewarding investment among these stocks over the analyzed period.

Summary

So, this is how you can perform a Quantitative Analysis of the Stock Market using Python. Quantitative Analysis in the stock market is a financial methodology that utilizes mathematical and statistical techniques to analyze stocks and financial markets. I hope you liked this article on Quantitative Analysis of Stock Market using Python. Feel free to ask valuable questions in the comments section below.

Aman Kharwal

AI/ML Engineer | Published Author. My aim is to decode data science for the real world in the most simple words.

PREVIOUS POST

Best Data Science Roles Based on Education Background

NEXT POST

Probability and Statistics for Data Science

Recommended For You

A Beginner's Guide to Using APIs in Data Science

October 23, 2025

Explain Your ML Model Predictions with a Visualization

October 18, 2025

Remote Data Science Internships for October

October 17, 2025

High-Paying Non-Technical AI Jobs

October 13, 2025

4 Comments

José Vargas JANUARY 20, 2024 / 4:35 AM REPLY

Excelent!

Bhabatosh Panda MARCH 18, 2024 / 3:54 PM REPLY

Very happy to read this example, it’s very basic but certainly gives motivitation for a great start.

Harsh JUNE 24, 2024 / 12:12 PM REPLY

Can I try this project on my own with taking help from this article.And can I add this to my portfolio too?

Aman Kharwal JUNE 25, 2024 / 1:00 AM REPLY

Yes, you can try this using whatever you learned from it.

Leave a Reply

FACEBOOK INSTAGRAM MEDIUM LINKEDIN