FINANCIAL ANALYTICS

Course Project

Submitted To

Dr. Vivek Vijay

Associate Professor

Indian Institute of Technology Jodhpur

Submitted By

Sai Swaroop Kakarla (G24AI1026)

Purushothaman S (G24AI1042)

Mugundan B (G24AI1051)

Sri Charan Reddy (G24AI1095)

Aakanksha Nalamati (G24AI1111)

Pranav Srivastav Tenali Gnana (G24AI1114)

Indian Institute of Technology Jodhpur

Date: 25th June 2025

Index

Introduction

This project performs a comprehensive financial analysis of 10 selected assets from the stock market using Python. The analysis covers three years of historical price data, computes descriptive statistics, visualizes price trends and returns, performs correlation analysis, and constructs an optimal portfolio based on Modern Portfolio Theory (Markowitz Efficient Frontier). The objective is to understand the risk-return characteristics of the chosen assets and develop a diversified investment index that maximizes return for a given level of risk.

Portfolio Analysis

Portfolio analysis is the process of evaluating and assessing a collection of investments (a *portfolio*) to understand its overall performance, risk profile, and potential returns. In essence, it involves examining how the different assets in a portfolio (such as stocks, bonds, real estate, etc.) are combined and how that mix affects the portfolio's return and volatility. The goal is often to determine whether the portfolio is aligned with the investor's objectives and risk tolerance, and to identify ways to improve its risk-adjusted performance. By analysing a portfolio's composition and historical performance, investors can gauge whether they are on track to meet their financial goals or if adjustments (like rebalancing asset allocations or replacing underperforming assets) are needed

Practical Uses in Personal Investing

For individual investors, portfolio analysis translates theory into the practical management of one's savings and investments. One major application is **diversification for risk management**. The classic adage "Don't put all your eggs in one basket" captures this idea. By analysing their portfolio, personal investors ensure they hold a variety of asset classes (stocks, bonds, cash, perhaps real estate, or others) so that no single investment can make or break the entire portfolio. Diversification helps *reduce the volatility and potential losses* in a portfolio more than it necessarily boosts returns. For example, instead of trying to pick one winning stock (and risking picking a loser), an investor might hold dozens of stocks or use broad index funds. History is full of stories of people striking it rich on a single stock pick, but for every big winner there are many investments that flop – thus spreading money across many securities is crucial to avoid catastrophic loss if one investment fails. Through portfolio analysis, an individual can assess whether their asset mix matches their risk tolerance and goals: e.g. a young investor might analyse and find they can take more equity exposure for growth, whereas a retiree's analysis might prompt shift to more bonds for stability. Regular analysis also informs **rebalancing** (adjusting the weights of assets back to target levels) – for instance, if stock markets have a big run-up and a portfolio becomes stock-heavy relative to the investor's plan, portfolio analysis would flag this increased risk, indicating it may be time to sell some stocks and buy other assets to restore the intended risk level. Overall, in personal investing, portfolio analysis is used to achieve a well-balanced portfolio that harnesses diversification to protect against undue risk while aiming for reasonable returns aligned with one's financial objectives.

Practical Uses in Institutional Finance (Pension Funds and Endowments)

In the realm of **corporate and institutional finance**, portfolio analysis is equally – if not more indispensable. Large institutional investors such as pension funds, insurance companies, and university endowments manage billions of dollars with the dual mandate of growth and safety, and they rely on rigorous portfolio analysis to fulfil their obligations. A pension fund, for example, must ensure that its investment portfolio can generate sufficient returns to pay retirees their benefits in the future. This leads to sophisticated asset-liability management and portfolio analysis: institutions will set an asset allocation policy (spread of stocks, bonds, real assets, alternatives, etc.) and continually analyse performance, risks, and changing market conditions. **Diversification** and risk management are paramount for institutions – they often diversify not only across traditional assets but also into alternatives like private equity, hedge funds, real estate, or infrastructure. In recent years, many pension plans have actually **expanded into alternative investments to boost returns and reduce overall risk**, especially in the face of low interest rates and volatile equity markets The 2008–2009 financial crisis and the ensuing low-yield environment, for instance, prompted numerous funds to rethink the classic heavy reliance on stocks and bonds By conducting detailed portfolio analyses (including stress tests and scenario analyses), institutional investors can find an optimal balance that aims to meet long-term return targets while controlling risk to an acceptable level.

Real-world examples illustrate these practices well. Historically, many institutions followed a simple "60/40" portfolio model (60% equities, 40% bonds) as a balanced strategy. However, some forward-thinking endowments and funds used analysis to pursue better diversification. **Yale University's endowment**, under the leadership of David Swensen, famously applied MPT principles to revolutionize its portfolio: instead of sticking to the 60/40 mix, Yale's portfolio was diversified across domestic and international stocks, bonds, plus a substantial allocation to alternative assets like hedge funds, private equity, real estate, and even timber. By taking advantage of Yale's long-time horizon and emphasizing broad diversification (in what became known as the "Yale Model"), Swensen was able to greatly improve the endowment's risk-adjusted returns. In fact, Yale's endowment performance outpaced the traditional 60/40 portfolio by about 4% per year under his tenure, demonstrating the tangible benefits of robust portfolio analysis and diversification. Likewise, large pension funds (e.g. CalPERS in the U.S. or major Canadian plans) regularly perform portfolio analysis to adjust their allocations in response to economic changes – for example, increasing allocations to infrastructure or private credit when analysis shows these can improve returns or provide better hedges against inflation. In sum, institutional portfolio analysis is about ensuring that a fund's investment strategy can meet its specific goals (such as paying future liabilities or funding a university's budget) with an acceptable level of risk. This involves continuous monitoring and tweaking of the portfolio using quantitative models, risk metrics, and performance benchmarks to guide decisions like hiring asset managers, shifting allocations, or hedging certain risks.

Significance and Benefits of Portfolio Analysis

Whether for an individual investor or a large institution, conducting portfolio analysis is highly beneficial for informed decision-making. First and foremost, it imposes a **disciplined, strategic approach** to investing rather than ad-hoc choices. By systematically reviewing how a portfolio is performing relative to its goals and risk parameters, investors can *identify imbalances or problems early* and take corrective action. For example, analysis might reveal that a portfolio's strong recent returns came with an unacceptably high concentration in a single sector – prompting diversification before a downturn hits that sector. Portfolio analysis also emphasizes **risk-adjusted performance** – not just looking at returns in a vacuum, but considering the amount of risk taken to achieve those returns (through measures like the Sharpe ratio or alpha). This helps in comparing a portfolio against benchmarks and understanding if the returns are adequate for the risk incurred. Another key benefit is aligning investments with objectives: over time an investor's goals or circumstances may change (say, approaching retirement or, for a foundation, needing more cash flows), and ongoing analysis ensures the portfolio's asset mix is realigned accordingly. Broadly, portfolio analysis aids in **optimizing resource allocation, improving risk management, and guiding strategic investment decisions**. It brings to light the trade-offs between different investment choices (for instance, how adding a new asset class might lower overall volatility) and thus supports more rational, evidence-based decisions rather than emotional reactions to market swings. Ultimately, the significance of portfolio analysis lies in how it enables investors to make *informed adjustments* to their portfolios – enhancing the likelihood of achieving long-term financial goals while avoiding unnecessary risks. In a world of uncertain markets, having this analytical insight is invaluable: it provides clarity on where you stand and confidence in how to proceed, making portfolio analysis an indispensable part of prudent investment management.

Script execution and Description Explanation Cell wise

Cell 1: Title and Description

# Portfolio Analysis

Added a title and short description to the notebook

Cell 2: Install Libraries

```
!pip install yfinance pandas numpy matplotlib seaborn scipy PyPortfolioOpt
```

Installs all required libraries for data fetching, analysis, visualization, and portfolio optimization.

The second cell ensures that all necessary Python packages are installed. These include libraries for financial data fetching (yfinance), numerical computations (NumPy and pandas), plotting (matplotlib and seaborn), and portfolio optimization (PyPortfolioOpt). This step is essential for running the notebook in a fresh environment using Google Colab.

Cell 3: Import Libraries

```
import yfinance as yf

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

from pypfopt import EfficientFrontier, risk_models, expected_returns, plotting

import statsmodels.api as sm
```

The third cell imports all the essential libraries required for the analysis. These include standard data manipulation and visualization packages, as well as **EfficientFrontier**, **expected_returns**, and **risk_models** from **PyPortfolioOpt**, which are used for constructing and **analyzing** the efficient frontier and optimizing the portfolio

Cell 4: Define Asset List

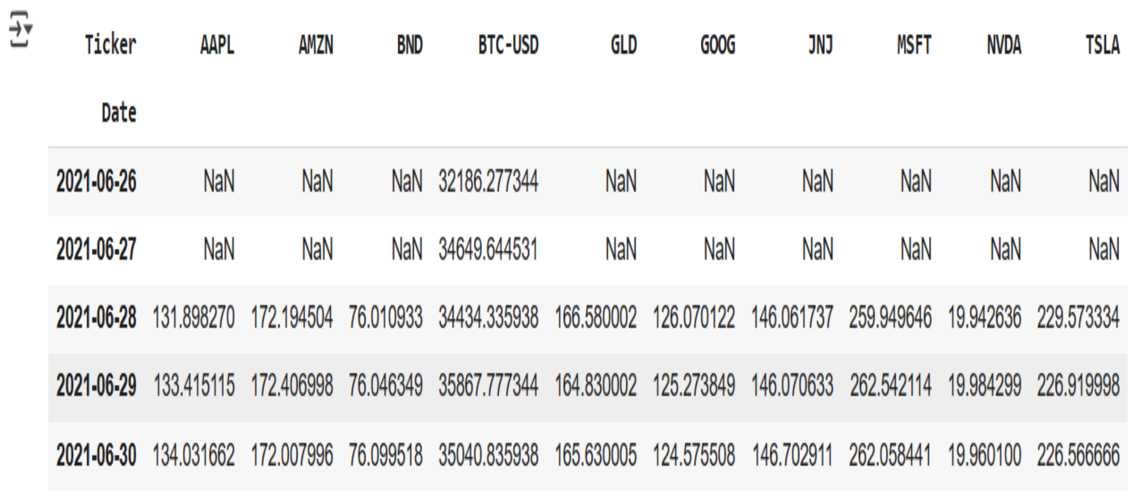
```
tickers = ['AAPL', 'MSFT', 'TSLA', 'AMZN', 'GOOG', 'NVDA', 'JNJ', 'BND', 'GLD', 'BTC-USD']
```

In the fourth cell, the user defines the list of 10 assets to analyse. These tickers are large-cap U.S. equities (e.g., AAPL, MSFT, GOOGL), but can be customized as per user preference. The cell also calculates the start and end dates for data collection, spanning the most recent three years using datetime and time delta from Python's standard library.

Download Price Data

```
data = yf.download(tickers, start='2021-06-26', end='2024-06-26',auto_adjust=False)['Adj Close']

data.to_csv("closing_prices.csv")

data.head()
```

The fifth cell fetches the historical price data for the selected assets using yfinance. download. Here, auto_adjust=False ensures that the "Adjusted Close" column is retained, which is important for calculating accurate historical returns accounting for dividends and splits. The result is a data frame of adjusted closing prices which is previewed using. head() to confirm the structure of the data.



Ticker	AAPL	AMZN	BND	BTC-USD	GLD	GOOG	JNJ	MSFT	NVDA	TSLA
Date										
2021-06-26	NaN	NaN	NaN	32186.277344	NaN	NaN	NaN	NaN	NaN	NaN
2021-06-27	NaN	NaN	NaN	34649.644531	NaN	NaN	NaN	NaN	NaN	NaN
2021-06-28	131.898270	172.194504	76.010933	34434.335938	166.580002	126.070122	146.061737	259.949646	19.942636	229.573334
2021-06-29	133.415115	172.406998	76.046349	35867.777344	164.830002	125.273849	146.070633	262.542114	19.984299	226.919998
2021-06-30	134.031662	172.007996	76.099518	35040.835938	165.630005	124.575508	146.702911	262.058441	19.960100	226.566666

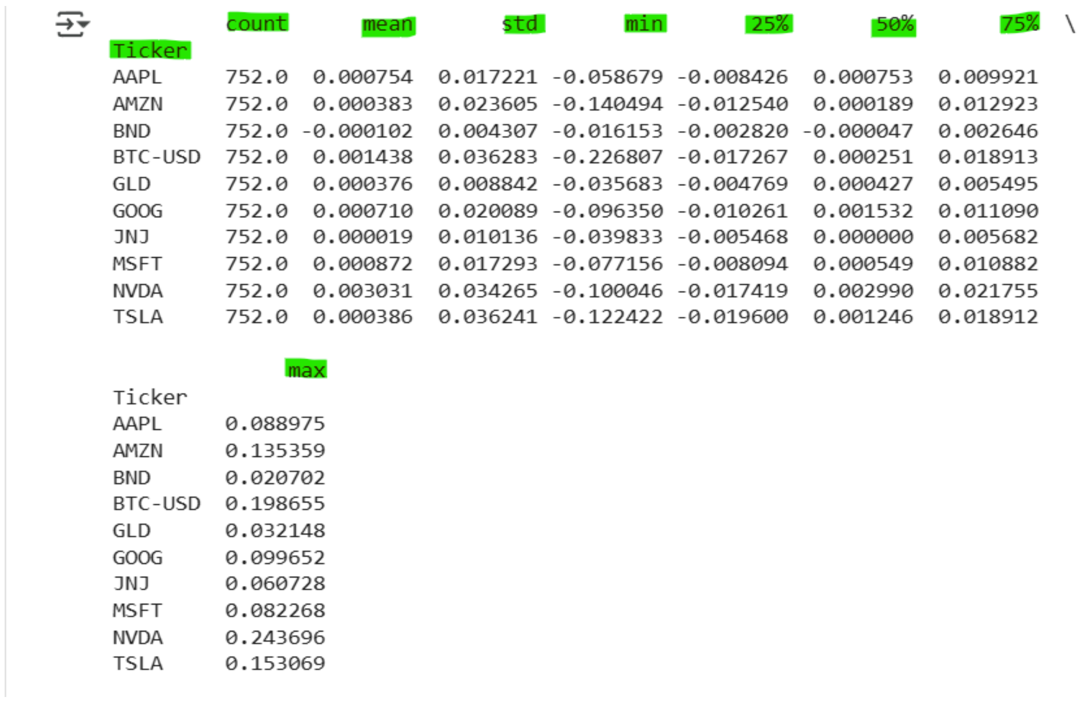
Cell 5: **data = data.dropna() - Drop rows with any NaNs (like weekends or missing stocks)**

Cell 6: **returns = data.pct_change().dropna() - Declaration of Return**

Cell 7: Calculate Descriptive Statistics

```
returns = data.pct_change().dropna()

# Histogram of returns

returns.hist(bins=50, figsize=(15, 10))

plt.suptitle("Histogram of Daily Returns")

plt.show()

# Descriptive Statistics

desc_stats = returns.describe().T

print(desc_stats)
```

Next, in the seventh cell, we performed an analysis of daily percentage returns of financial data stored in a DataFrame called data. First, it calculates the daily percentage change for each column (typically representing different financial instruments or assets) using data.pct_change() and removes any resulting NaN values with .dropna(). The resulting returns DataFrame contains the daily return values. A histogram is then plotted for each asset's daily returns using returns.hist(), with 50 bins and a large figure size to ensure clarity. The plt.suptitle() function adds a title above all subplots to indicate that the chart represents histograms of daily returns. Finally, descriptive statistics such as count, mean, standard deviation, min, max, and quartiles are computed for each asset using returns.describe().T (which transposes the output for easier readability), and the result is printed to the console. This provides a summary view of the return distributions for further analysis.

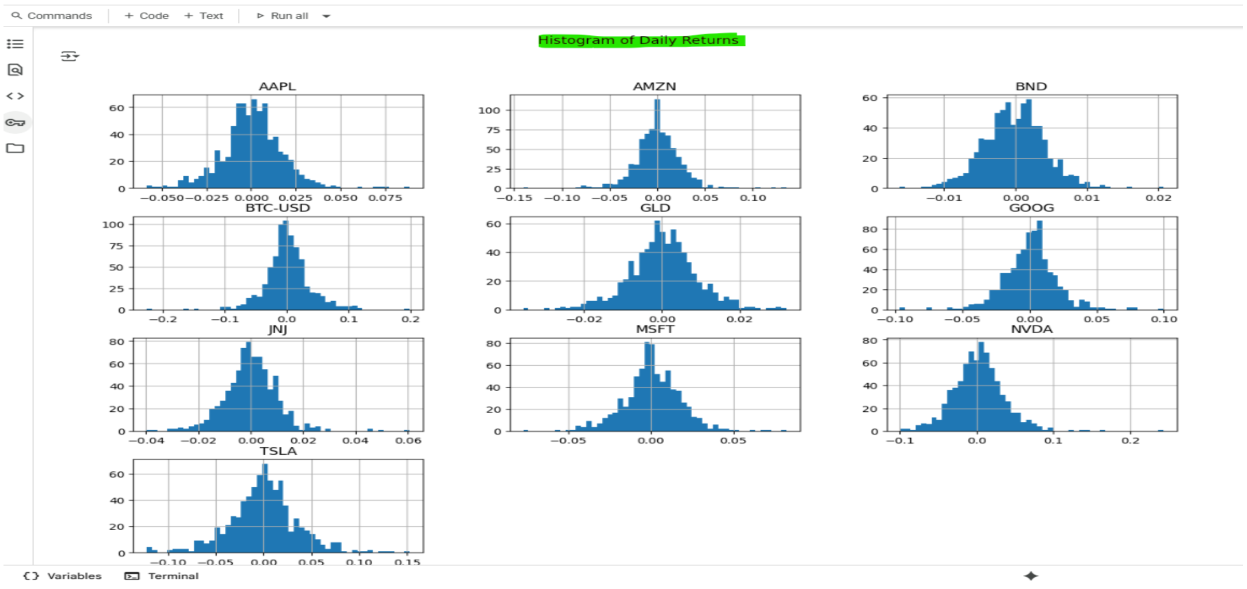


```
         count      mean       std       min       25%       50%       75%  \
Ticker
AAPL     752.0  0.000754  0.017221 -0.058679 -0.008426  0.000753  0.009921
AMZN     752.0  0.000383  0.023605 -0.140494 -0.012540  0.000189  0.012923
BND      752.0 -0.000102  0.004307 -0.016153 -0.002820 -0.000047  0.002646
BTC-USD  752.0  0.001438  0.036283 -0.226807 -0.017267  0.000251  0.018913
GLD      752.0  0.000376  0.008842 -0.035683 -0.004769  0.000427  0.005495
GOOG     752.0  0.000710  0.020089 -0.096350 -0.010261  0.001532  0.011090
JNJ      752.0  0.000019  0.010136 -0.039833 -0.005468  0.000000  0.005682
MSFT     752.0  0.000872  0.017293 -0.077156 -0.008094  0.000549  0.010882
NVDA     752.0  0.003031  0.034265 -0.100046 -0.017419  0.002990  0.021755
TSLA     752.0  0.000386  0.036241 -0.122422 -0.019600  0.001246  0.018912

              max
Ticker
AAPL     0.088975
AMZN     0.135359
BND      0.020702
BTC-USD  0.198655
GLD      0.032148
GOOG     0.099652
JNJ      0.060728
MSFT     0.082268
NVDA     0.243696
TSLA     0.153069
```

Cell 8: Data Visualization through Histograms

```
returns.hist(bins=50, figsize=(15, 10))

plt.suptitle("Histogram of Daily Returns")

plt.show()
```

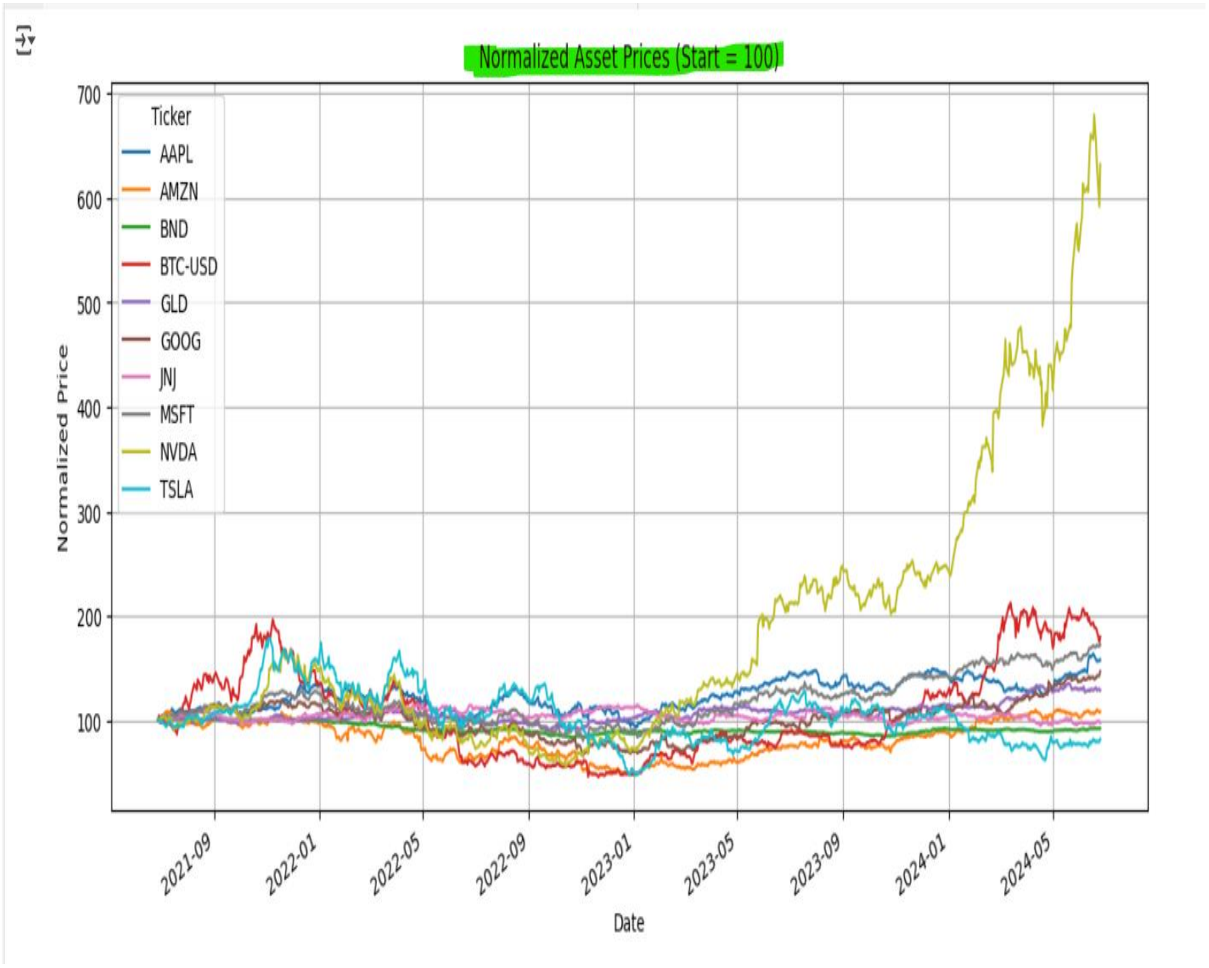
The Eight cell produces two important visualizations. The first is a time series plot of the adjusted closing prices for each asset, which helps visualize their trends and performance over time. The second is a set of histograms showing the distribution of returns for each asset. These histograms allow us to visually assess the spread and skewness of returns, helping identify which assets have more stable or volatile return distributions.



Cell 9: Normalization Comparison of Asset Price Performance Over Time

```
normalized = data / data.iloc[0] * 100

normalized.plot(figsize=(14,6), title="Normalized Asset Prices (Start = 100)")

plt.ylabel("Normalized Price")

plt.grid(True)

plt.show()
```

In the nineth cell, normalizes asset price data to enable a clear and direct comparison of their relative performance over time. By dividing each asset's price by its value on the first date (data.iloc[0]) and multiplying by 100, it sets all assets to start at a common baseline value of 100. This transformation shows how much each asset has grown or declined in percentage terms relative to its starting point, regardless of its original price scale. The normalized data is then plotted on a line chart using a specified figure size, with an appropriate title and labeled y-axis ("Normalized Price"). A grid is added for improved readability. This visualization is particularly useful in financial analysis to evaluate and compare the growth trends of different assets over the same period, highlighting which ones have outperformed or underperformed relative to others.



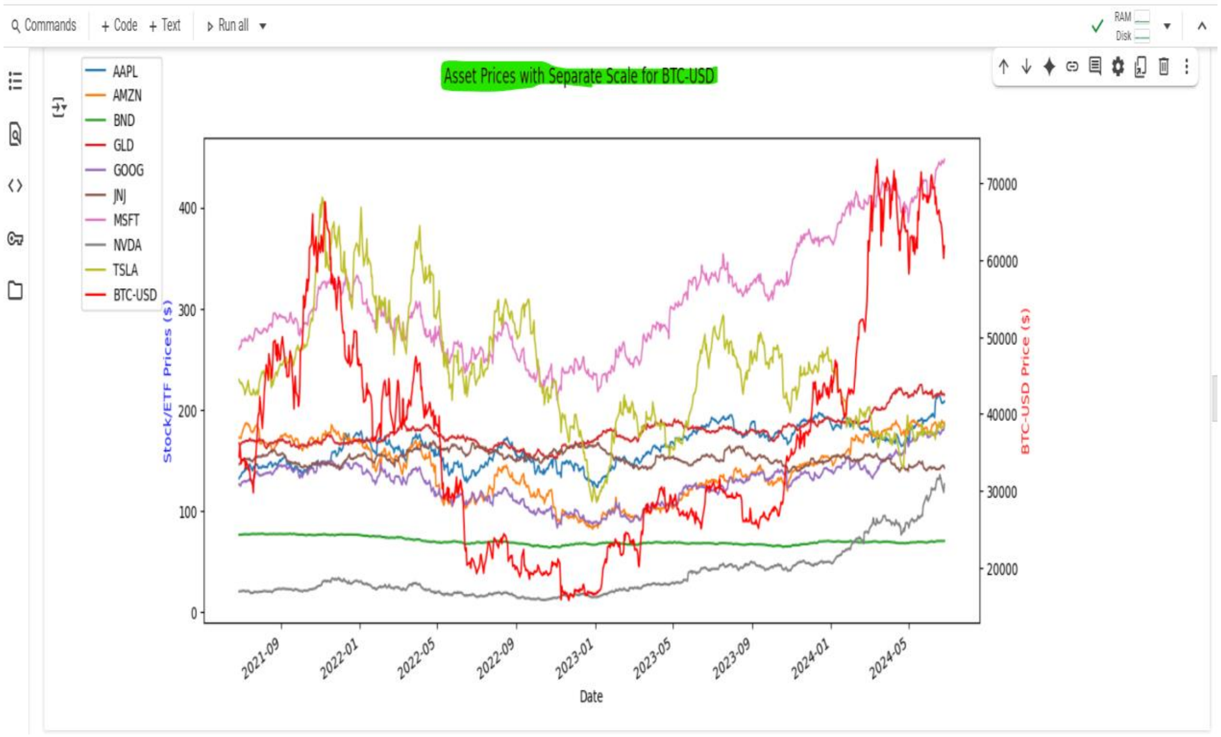
Cell 10: Dual-Axis Plot of Asset Prices with Separate Scale for BTC-USD

Script

```
# Dual-Axis Plot of Asset Prices with Separate Scale for BTC-USD
fig, ax1 = plt.subplots(figsize=(14,6))

# Plot non-BTC assets
non_btc = data.drop(columns='BTC-USD')
non_btc.plot(ax=ax1, legend=False)
ax1.set_ylabel("Stock/ETF Prices ($)", color='blue')

# Plot BTC on secondary y-axis
ax2 = ax1.twinx()
data['BTC-USD'].plot(ax=ax2, color='red', label='BTC-USD')
ax2.set_ylabel("BTC-USD Price ($)", color='red')

fig.suptitle("Asset Prices with Separate Scale for BTC-USD")
fig.legend(loc='upper left')
plt.show()
```

In the tenth cell we are creating a **dual-axis line plot** to visually compare Bitcoin's price (BTC-USD) with other assets such as stocks or ETFs, which typically have much lower price scales. It starts by initializing a plot with two y-axes (ax1 and ax2) on the same x-axis (time). The first axis (ax1) is used to plot all assets **excluding BTC-USD**, with the y-axis labeled in blue to distinguish it. The second axis (ax2) is created using twinx() to share the x-axis but allow for a **different y-scale**, and it is used to plot **only the BTC-USD** price in red.

This separation is necessary because Bitcoin's price often differs by orders of magnitude from traditional assets, and plotting everything on the same y-axis would make the smaller-scale assets unreadable. A title and legend are added to clearly describe the chart. This type of visualization is especially useful in financial data analysis when comparing the **trends** of assets with drastically different price ranges on the **same time scale**.



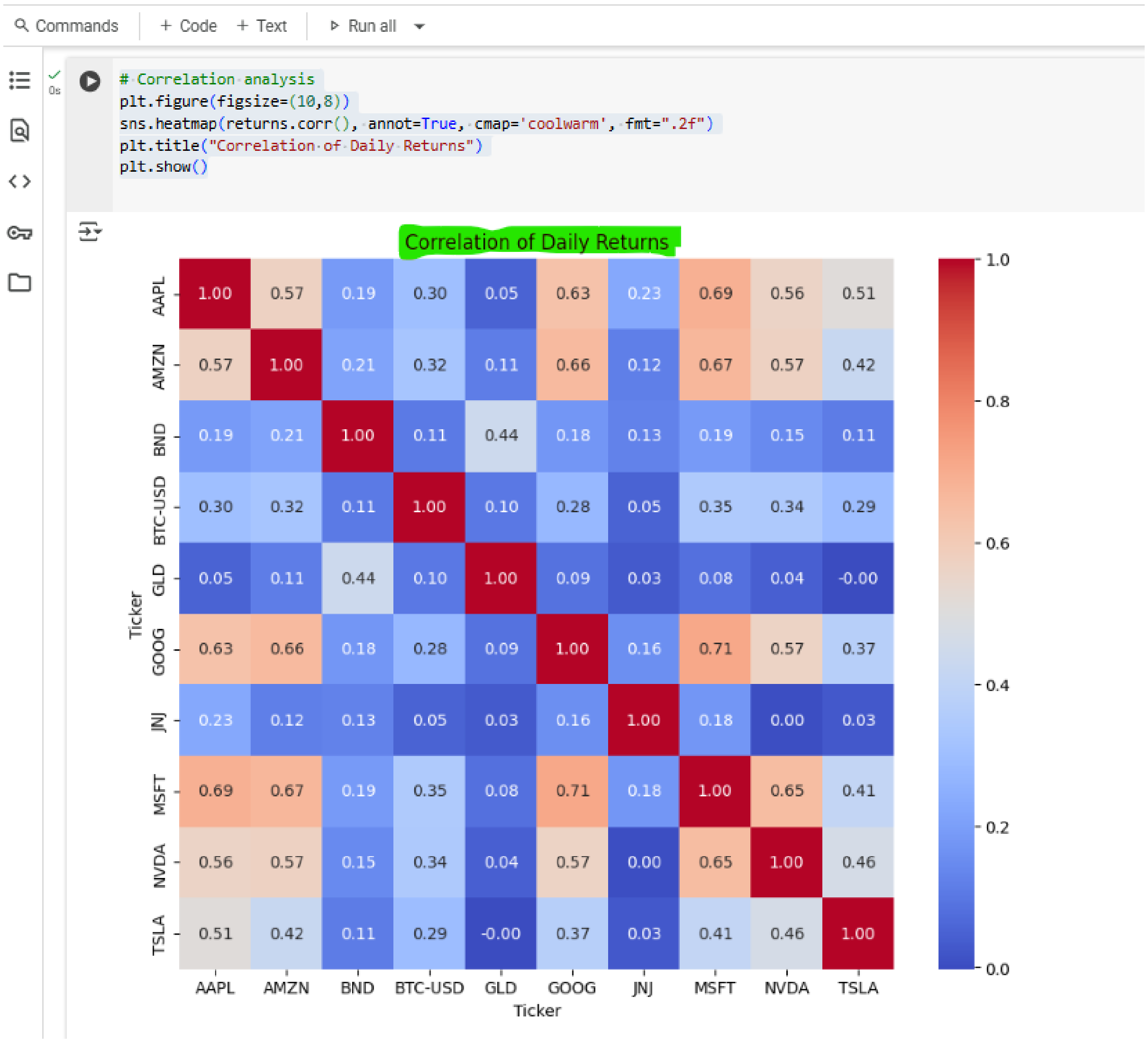
Cell 11: Correlation analysis

Script

```
# Correlation analysis
plt.figure(figsize=(10,8))
sns.heatmap(returns.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation of Daily Returns")
plt.show()
```

In the eleventh cell we are performing a correlation analysis of the daily returns of multiple financial assets and visualizes the results using a heatmap. The function returns.corr() computes the Pearson correlation coefficient between each pair of assets, indicating how similarly they move relative to each other. A value close to 1 means strong positive correlation (they move together), while a value close to -1 indicates a strong negative correlation (they move in opposite directions).

The sns.heatmap() function from the Seaborn library is used to create a colored matrix of these correlation values. It displays the coefficients as numbers (with two decimal places) and colors each cell using the 'coolwarm' color map—where warmer colors typically indicate higher correlation. The figure is sized for clarity, titled "Correlation of Daily Returns," and finally shown with plt.show(). This visual summary helps quickly identify which assets have similar return patterns and can inform diversification and risk management strategies in portfolio construction



```
# Correlation analysis
plt.figure(figsize=(10,8))
sns.heatmap(returns.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation of Daily Returns")
plt.show()
```

Correlation of Daily Returns

Ticker	AAPL	AMZN	BND	BTC-USD	GLD	GOOG	JNJ	MSFT	NVDA	TSLA
AAPL	1.00	0.57	0.19	0.30	0.05	0.63	0.23	0.69	0.56	0.51
AMZN	0.57	1.00	0.21	0.32	0.11	0.66	0.12	0.67	0.57	0.42
BND	0.19	0.21	1.00	0.11	0.44	0.18	0.13	0.19	0.15	0.11
BTC-USD	0.30	0.32	0.11	1.00	0.10	0.28	0.05	0.35	0.34	0.29
GLD	0.05	0.11	0.44	0.10	1.00	0.09	0.03	0.08	0.04	-0.00
GOOG	0.63	0.66	0.18	0.28	0.09	1.00	0.16	0.71	0.57	0.37
JNJ	0.23	0.12	0.13	0.05	0.03	0.16	1.00	0.18	0.00	0.03
MSFT	0.69	0.67	0.19	0.35	0.08	0.71	0.18	1.00	0.65	0.41
NVDA	0.56	0.57	0.15	0.34	0.04	0.57	0.00	0.65	1.00	0.46
TSLA	0.51	0.42	0.11	0.29	-0.00	0.37	0.03	0.41	0.46	1.00

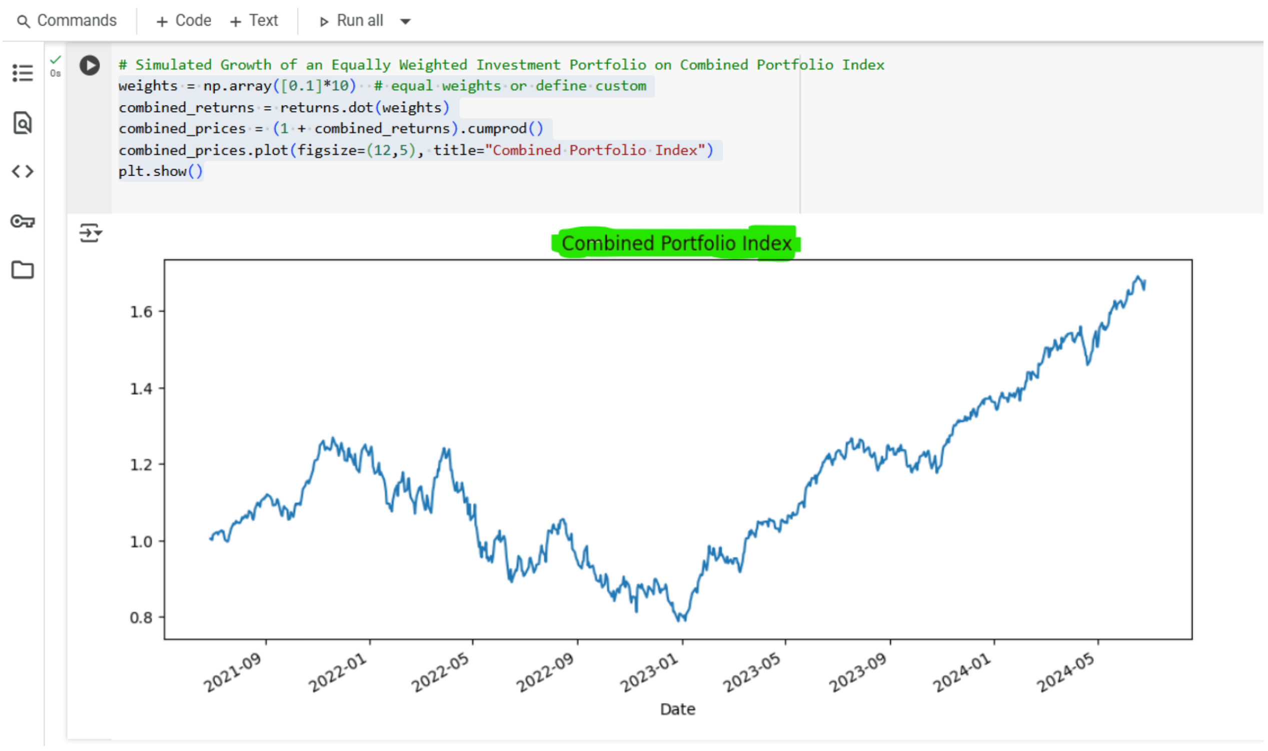
Cell 12: Simulated Growth of an Equally Weighted Investment Portfolio on Combined Portfolio Index

Script

```
weights = np.array([0.1]*10)  # equal weights or define custom
combined_returns = returns.dot(weights)
combined_prices = (1 + combined_returns).cumprod()
combined_prices.plot(figsize=(12,5), title="Combined Portfolio Index")
plt.show()
```

In the twelfth cell we are simulates and visualizes the performance of a **combined investment portfolio** composed of 10 assets, each assigned an equal weight of 10% (or custom-defined). The weights array defines how the total investment is distributed across the assets. By taking the dot product of the returns DataFrame with the weights, it calculates the **daily return of the entire portfolio** as a weighted average of individual asset returns.

The next step, (1 + combined_returns).cumprod(), computes the **cumulative return** of the portfolio over time—starting from an initial value (implicitly $1)—to simulate how the investment grows. The resulting combined_prices series reflects the hypothetical growth of the portfolio's value over the time period. Finally, this is plotted in a line chart titled **"Combined Portfolio Index"**, providing a clear visualization of the portfolio's performance trajectory. This approach is commonly used in portfolio analysis to evaluate the effect of diversification and weighting strategies



```
# Simulated Growth of an Equally Weighted Investment Portfolio on Combined Portfolio Index
weights = np.array([0.1]*10)  # equal weights or define custom
combined_returns = returns.dot(weights)
combined_prices = (1 + combined_returns).cumprod()
combined_prices.plot(figsize=(12,5), title="Combined Portfolio Index")
plt.show()
```

Cell 13: Markowitz Efficient Frontier Analysis and Visuals

Script

mu = expected_returns.mean_historical_return(data)

S = risk_models.sample_cov(data)

ef = EfficientFrontier(mu, S)

fig, ax = plt.subplots()

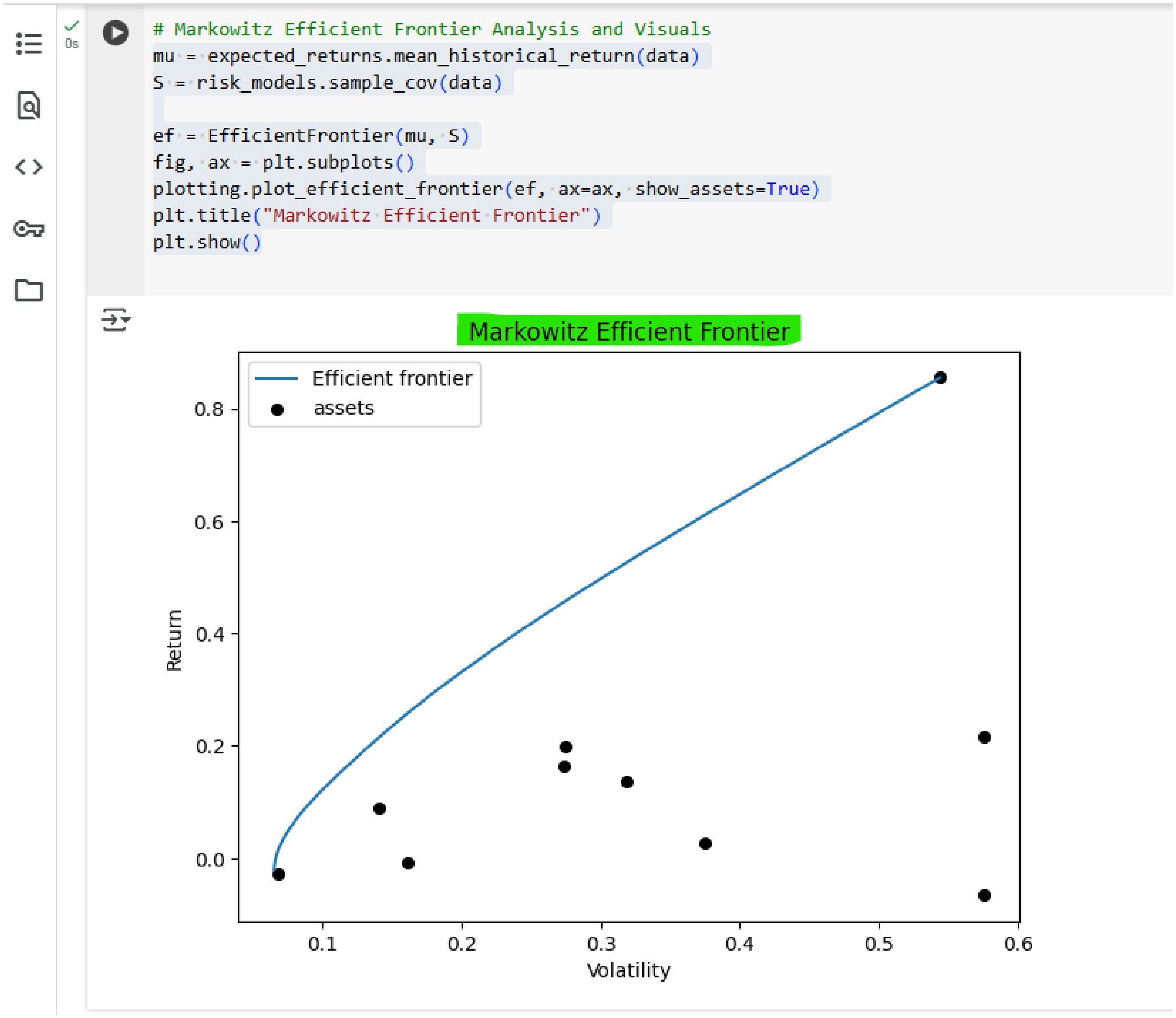
plotting.plot_efficient_frontier(ef, ax=ax, show_assets=True)

plt.title("Markowitz Efficient Frontier")

plt.show()

in the thirteenth cell we compute and visualizes the Markowitz Efficient Frontier, which represents the set of optimal portfolios offering the highest expected return for a given level of risk. It begins by calculating the expected returns (mu) using the historical mean return of the assets via mean_historical_return(data) and the sample covariance matrix (S) using sample_cov(data)—both from the pypfopt library.

These values are then passed to the EfficientFrontier class, which applies Modern Portfolio Theory (MPT) to determine efficient portfolios. The plot_efficient_frontier() function generates a plot of this frontier on a risk-return graph, with each point representing a different asset combination. The show_assets=True option displays individual asset positions on the graph. The plot is titled "Markowitz Efficient Frontier" and shown using plt.show(). This visualization is useful for investors and analysts seeking to construct a diversified portfolio that balances risk and return optimally

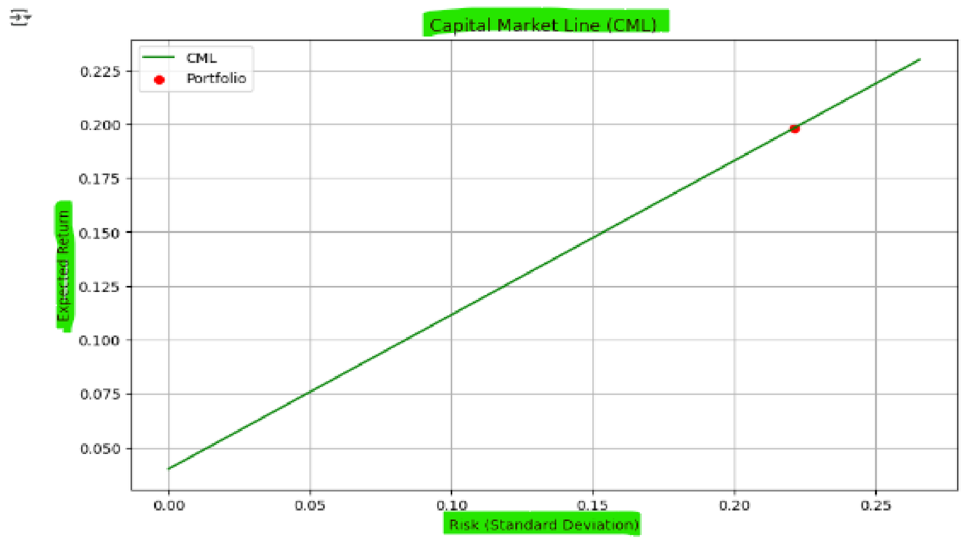


```
# Markowitz Efficient Frontier Analysis and Visuals
mu = expected_returns.mean_historical_return(data)
S = risk_models.sample_cov(data)

ef = EfficientFrontier(mu, S)
fig, ax = plt.subplots()
plotting.plot_efficient_frontier(ef, ax=ax, show_assets=True)
plt.title("Markowitz Efficient Frontier")
plt.show()
```

Cell 14: Capital Market Line Analysis and Visuals and Annualized return & risk of combined portfolio

Script

```
risk_free_rate = 0.04

portfolio_return = combined_returns.mean() * 252

portfolio_volatility = combined_returns.std() * (252 ** 0.5)

sharpe_ratio = (portfolio_return - risk_free_rate) / portfolio_volatility

import numpy as np

import matplotlib.pyplot as plt

# CML line

x = np.linspace(0, portfolio_volatility * 1.2, 100)

y = risk_free_rate + sharpe_ratio * x

#Plot

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

plt.plot(x, y, label="CML", color='green')

plt.scatter(portfolio_volatility, portfolio_return, color='red', label="Portfolio")

plt.title("Capital Market Line (CML)")

plt.xlabel("Risk (Standard Deviation)")

plt.ylabel("Expected Return")

plt.legend()

plt.grid(True)

plt.show()
```



Here we a compute Capital Market Line (CML) analysis to evaluate the performance of a combined investment portfolio relative to the optimal risk-return trade-off. It begins by setting a risk-free rate (e.g., from government bonds) at 4%, then calculates the annualized return and annualized volatility (risk) of the portfolio using daily returns (combined_returns). The Sharpe Ratio—a measure of risk-adjusted return—is computed as the excess return over the risk-free rate divided by the portfolio's volatility.

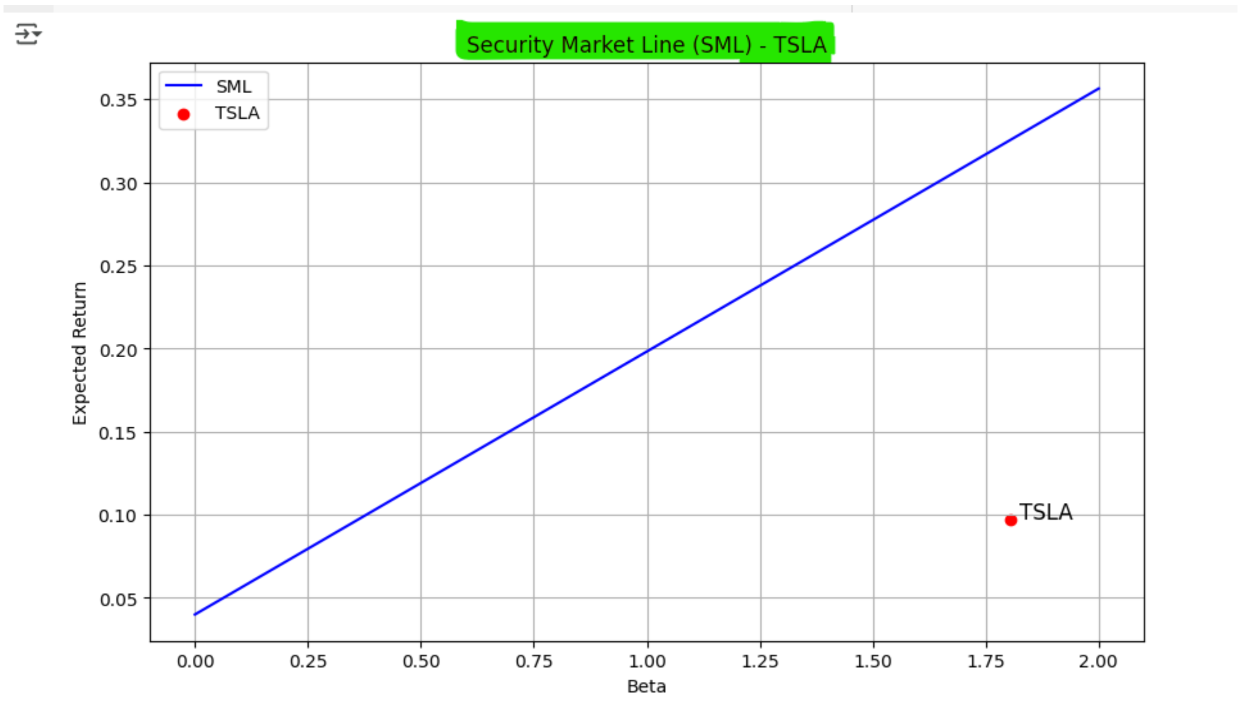
Next, the code generates the CML, which illustrates the highest expected return for a given level of risk when combining a risk-free asset with a risky portfolio. The line is plotted using the Sharpe ratio as its slope, spanning a range of risk levels (x). The portfolio's own return and risk are plotted as a red point on the graph for visual comparison. The chart, titled "Capital Market Line (CML)", clearly shows how the portfolio aligns with or deviates from the optimal trade-off between risk and return. This analysis helps investors assess whether their portfolio is efficiently positioned in terms of reward for risk taken

Cell 15: Security Market Line Analysis and Visuals

Script

```
# Market = your combined portfolio returns

market = combined_returns

asset = returns['TSLA']  # the asset we compare

# Linear regression: TSLA ~ market

X = sm.add_constant(market)

y = asset

model = sm.OLS(y, X).fit()

beta_tsla = model.params[0]

alpha_tsla = model.params['const']

# CAPM Expected return of TSLA

market_return = market.mean() * 252

expected_tsla_return = risk_free_rate + beta_tsla * (market_return - risk_free_rate)

actual_tsla_return = y.mean() * 252

# SML line

betas = np.linspace(0, 2, 100)

sml_line = risk_free_rate + (market_return - risk_free_rate) * betas
```

```
# Plot SML

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

plt.plot(betas, sml_line, label='SML', color='blue')

plt.scatter(beta_tsla, actual_tsla_return, color='red', label='TSLA')

plt.text(beta_tsla + 0.02, actual_tsla_return, 'TSLA', fontsize=12)

plt.xlabel("Beta")

plt.ylabel("Expected Return")

plt.title("Security Market Line (SML) - TSLA")

plt.legend()

plt.grid(True)

plt.show()
```



Here we are computing Security Market Line (SML) analysis for the asset TSLA (Tesla) (Sample stack) using the Capital Asset Pricing Model (CAPM) framework. The analysis begins by treating the previously computed combined portfolio returns as the proxy for the market returns. It then selects TSLA's daily returns from the returns DataFrame and runs an Ordinary Least Squares (OLS) linear regression to model the relationship between TSLA and the market: TSLA ~ Market.

The regression yields two key coefficients:

- Alpha (intercept): TSLA’s return independent of the market.

- Beta (slope): TSLA's sensitivity to market movements.

Using CAPM, the expected return of TSLA is calculated based on the risk-free rate and TSLA's beta:

Expected ReturnTSLA=Rf+β·(Rm−Rf)\text{Expected Return}_{\text{TSLA}} = R_f + \beta \cdot (R_m - R_f)Expected ReturnTSLA=Rf+β·(Rm−Rf)

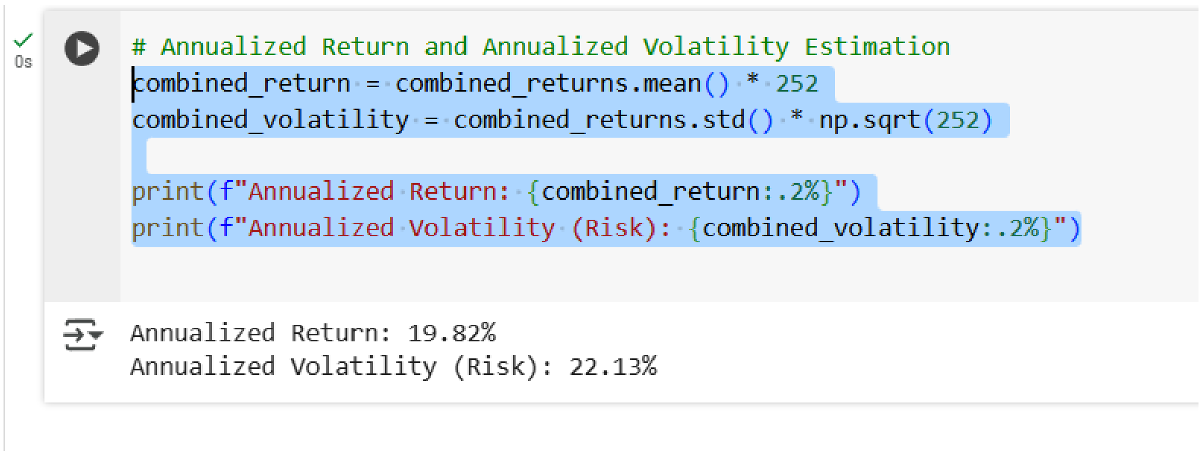
where RfR_fRf is the risk-free rate, and RmR_mRm is the market's expected return. The actual annualized return of TSLA is also computed for comparison.

Then, the Security Market Line is plotted by calculating the expected return for a range of betas from 0 to 2. The SML represents the theoretical expected return for any asset given its beta. Finally, TSLA is plotted as a red dot based on its actual beta and return, with a label. If TSLA lies above the SML, it may be undervalued (offering a higher return than expected); if below, it may be overvalued. This visualization helps evaluate whether TSLA's risk-adjusted return aligns with CAPM predictions**.**

Cell 16: Annualized Return and Annualized Volatility Estimation

Script

```
combined_return = combined_returns.mean() * 252

combined_volatility = combined_returns.std() * np.sqrt(252)

print(f"Annualized Return: {combined_return:.2%}")

print(f"Annualized Volatility (Risk): {combined_volatility:.2%}")
```



```
# Annualized Return and Annualized Volatility Estimation
combined_return = combined_returns.mean() * 252
combined_volatility = combined_returns.std() * np.sqrt(252)

print(f"Annualized Return: {combined_return:.2%}")
print(f"Annualized Volatility (Risk): {combined_volatility:.2%}")
```

```
Annualized Return: 19.82%
Annualized Volatility (Risk): 22.13%
```

here we calculate and display the annualized return and annualized volatility (risk) of a combined investment portfolio.

- The line combined_return = combined_returns.mean() * 252 computes the average daily return of the portfolio and multiplies it by 252 (the typical number of trading days in a year) to estimate the annualized return.
- The line combined_volatility = combined_returns.std() * np.sqrt(252) calculates the standard deviation of daily returns (a measure of daily risk) and scales it by the square root of 252 to obtain the annualized volatility.

Finally, the results are printed in percentage format using Python’s formatted string literals, clearly reporting the expected yearly return and risk level of the portfolio. This estimation is a key step in portfolio performance evaluation, allowing investors to understand the long-term potential and variability of returns.

Conclusion

On this portfolio analysis we demonstrate the application of quantitative techniques to evaluate and compare the performance, risk, and interrelationships of selected financial assets. By normalizing prices, we were able to compare asset performance on a relative scale, highlighting their growth trajectories over time. The histogram and descriptive statistics of daily returns offered insights into return distributions and volatility across individual assets.

Correlation analysis using a heatmap revealed the strength of relationships among the assets, aiding in diversification decisions. The dual-axis plot effectively separated Bitcoin's scale from traditional assets, enhancing visual clarity. A simulated equally-weighted portfolio was constructed and tracked, with its cumulative performance visualized and evaluated.

We further applied Modern Portfolio Theory (MPT) using the Markowitz Efficient Frontier to identify optimal portfolios, and assessed the combined portfolio's positioning using the Capital Market Line (CML). The Sharpe ratio confirmed the portfolio's risk-adjusted return.

Finally, a Security Market Line (SML) analysis was performed for TSLA, comparing its actual performance to CAPM expectations based on beta, helping identify potential under- or overvaluation.

Overall, this comprehensive analysis not only highlighted portfolio-level dynamics but also provided asset-specific insights, laying the groundwork for informed investment strategy and risk management.

Git Link: https://github.com/PurushothamanShanmugam/Financial-Analytics-Project

Sources:

1. Wall Street Mojo – *"Portfolio Analysis – Meaning, Steps, Tools, Advantages & Examples."* Definition and key aspects of portfolio analysis wallstreetmojo.com.
2. awork Glossary – *"Portfolio Analysis."* Explanation of Markowitz's modern portfolio theory and its assumptions about risk and return awork.com.
3. Investopedia – *"Modern Portfolio Theory (MPT)."* Discussion of diversification, risk-return trade-offs, and the efficient frontier in portfolio construction investopedia.com.
4. Vanguard – *"Portfolio Diversification: What it is and how it works."* Emphasis on diversification as a risk management strategy for individual investors investor.vanguard.cominvestor.vanguard.com.
5. Yale News – *"David Swensen's Coda."* Real-world example of Yale Endowment's diversified portfolio strategy outperforming a traditional 60/40 portfolio by applying MPT principles news.yale.edu.
6. IMD (I by IMD) – *"The Great Diversification: Why pension funds are moving beyond stocks and bonds."* Description of how pension funds diversify into alternative assets to improve returns and manage risk imd.org.
7. awork Glossary – *"Portfolio Analysis."* Summary of the importance of portfolio analysis in optimizing investments and informing decisions awork.com