Portfolio Evaluation With Equal Weighted Investing

By Hakeem Lawrence, BI Engineer

Introduction

As an investor, it can be challenging to navigate the landscape and select the best stocks for your portfolio. One approach is to select a group of stocks that you believe will perform well and combine them into a portfolio with equal weights. This approach is known as equal weighted investing and can potentially lead to higher returns than simply investing in a market index.

In this analysis, we will explore the concept of the 1/N portfolio strategy by selecting five of the most promising stocks on the market and combining them into a portfolio. Our goal is to analyze how this portfolio performs compared to the S&P 500, the benchmark for the stock market. By doing so, we hope to gain insights into the effectiveness of an equal weighted portfolio as an investment strategy.

Methodology

To begin our analysis, we first selected five stocks that we believe have strong potential and are representative of different sectors of the economy. These stocks are J.P. Morgan (JPM), Nvidia (NVDA), Eli Lilly (LLY), Microsoft (MSFT), and Walmart (WMT). We then combined these stocks into a portfolio and calculate the returns of this portfolio year-to-date.

To compare the performance of our portfolio to the broader market, we used the S&P 500 as our benchmark. The S&P 500 is a market index that tracks the performance of 500 large-cap companies listed on the New York Stock Exchange or NASDAQ. We calculated the returns of the S&P 500 over the same period as our portfolio.

Get Asset Data

In [ ]:
```
#import libraries

import quantstats as qs, pandas as pd, numpy as np, yfinance as yf, matplotlib.pyplot as plt
```

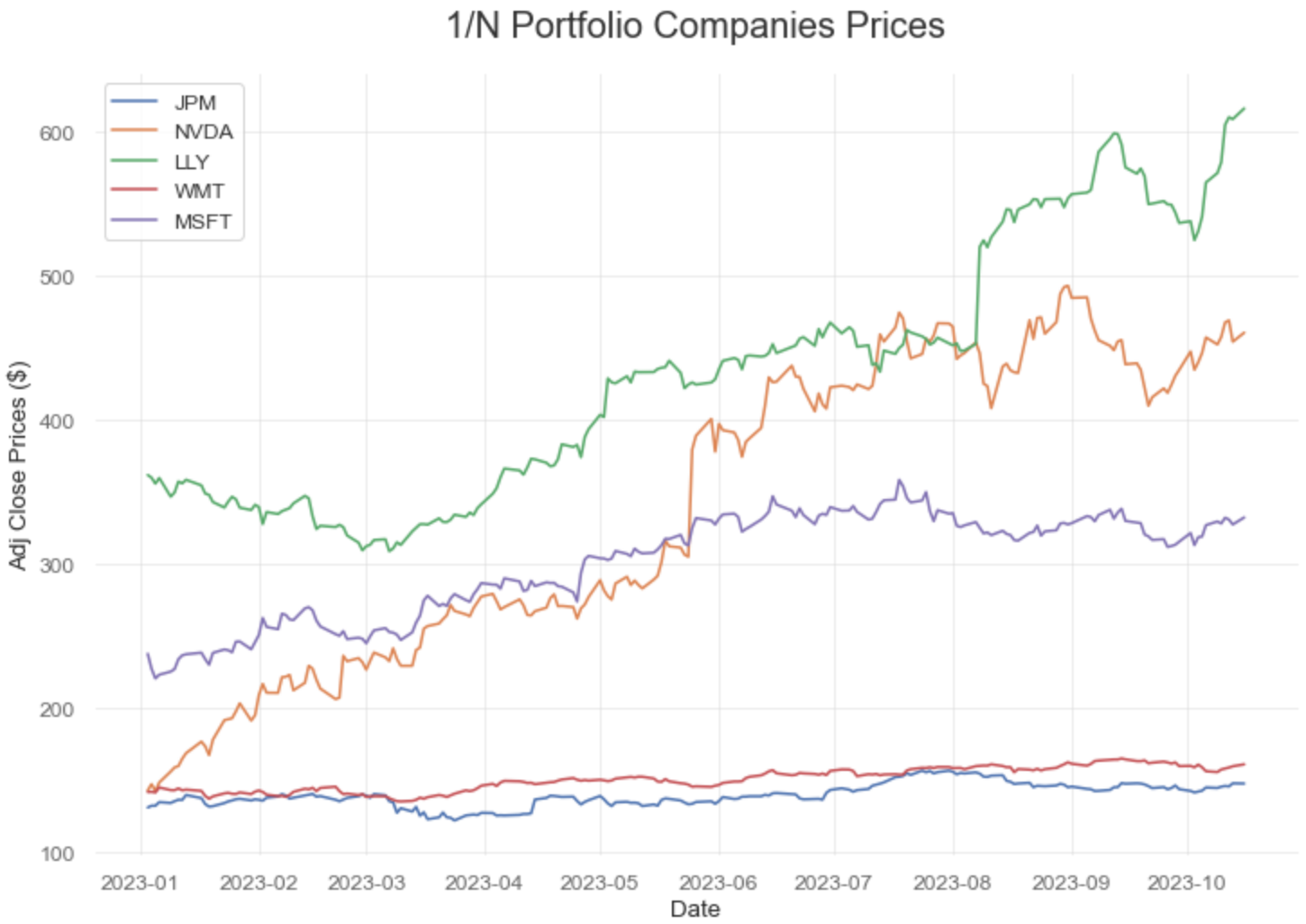
In [ ]:
```
# Portfolio Assets: J.P. Morgan, Nvdia, Eli Lilly, Walmart, Microsoft

assets = ['JPM', 'NVDA', 'LLY', 'WMT', 'MSFT']
n_assets = len(assets)

# Downloading historical adjusted close prices for each company

assetPrices = yf.download(
        assets,
        start='2023-01-01',
        end='2023-10-17'
        )['Adj Close'].reset_index()


# simple returns

assetReturns = (
    assetPrices
    .set_index('Date')
    .pct_change()
    .dropna()

)

# log returns
assetLogReturns = np.log(assetPrices.set_index('Date')/assetPrices.set_index('Date').shift(1)).dropna()
```

```
# cumulative returns
cumulativeLogReturns = assetLogReturns.cumsum()


# Generating the weights for each asset in our portfolio
portfolioWeights = n_assets*[1/n_assets]
```

```
[**********100%%**********]  5 of 5 completed
```

Explore Price Action and Returns

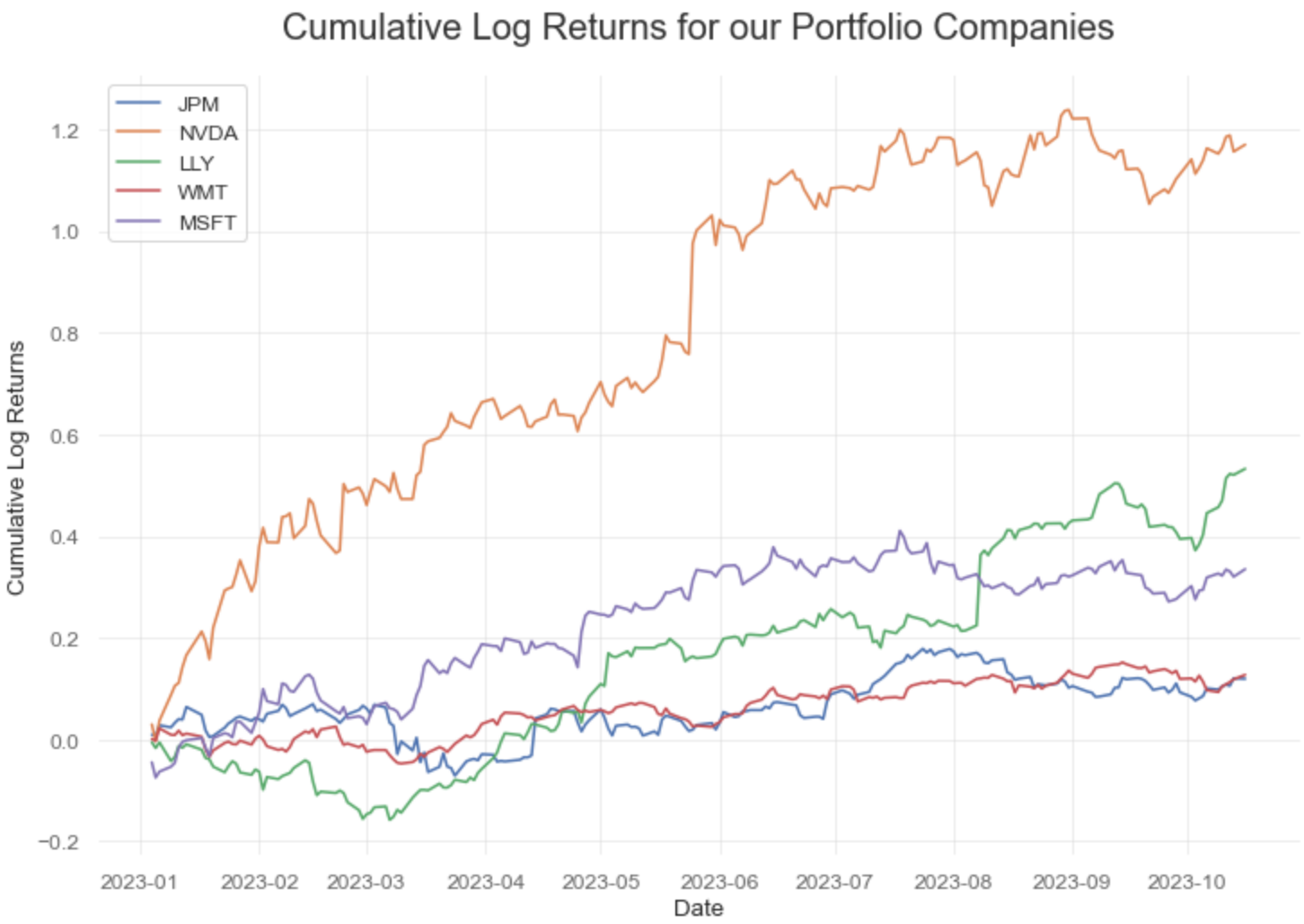
In [ ]:
```
fig, ax = plt.subplots(figsize=(12, 8))
for asset in assets:
    ax.plot(assetPrices['Date'], assetPrices[asset], label=asset)
ax.legend()
plt.ylabel('Adj Close Prices ($)')
plt.xlabel('Date')
plt.title('1/N Portfolio Companies Prices', fontsize =20,pad=20)
plt.show()
```



Eli Lilly takes the number one spot for the highest adjusted close price. However, this doesn't mean they've had the largest growth. We will normailze the values to get a better understanding of price returns.

In [ ]:
```
fig, ax = plt.subplots(figsize=(12, 8))
for asset in assets:
    ax.plot(cumulativeLogReturns.reset_index()['Date'], cumulativeLogReturns.reset_index()[asset], label=asset)
ax.legend()
plt.ylabel('Cumulative Log Returns')
plt.xlabel('Date')
```

```
plt.title('Cumulative Log Returns for our Portfolio Companies', fontsize =20,pad=20)
plt.show()
```



In [ ]:
```
cumulativeLogReturns.iloc[-1]
```

Out[ ]:
```
JPM     0.119408
LLY     0.532385
MSFT    0.334974
NVDA    1.169752
WMT     0.127107
Name: 2023-10-16 00:00:00, dtype: float64
```

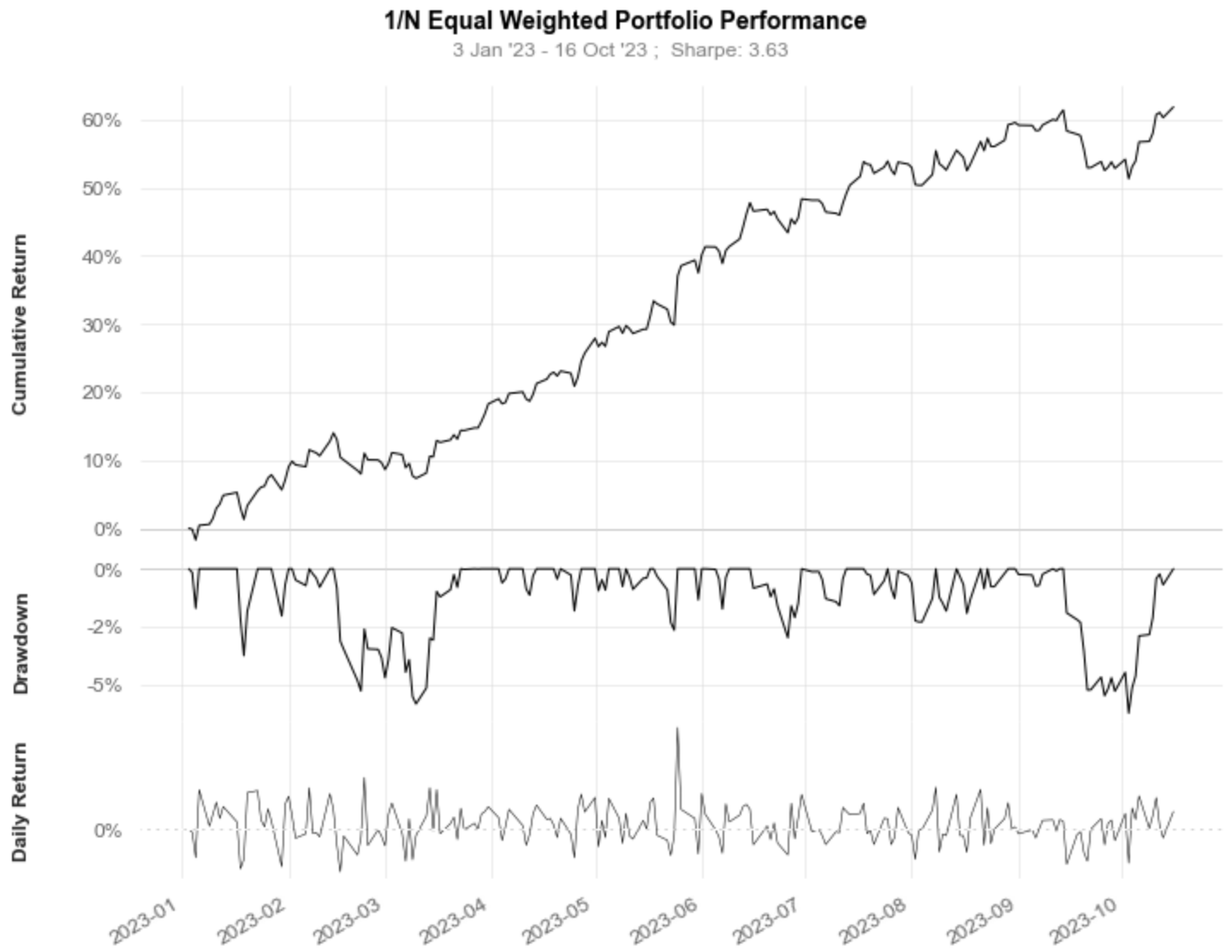
After normalizing our values, we can see Nvidia putting on an incredible display by outperforming all other assets.

Evaluate the portfolio

In [ ]:
```
# get returns for our portfolio

portfolioReturns = pd.Series(
    np.dot(portfolioWeights, assetReturns.T),
    index= assetReturns.index
)

portfolioReturns
```

Out[ ]:
```
Date
2023-01-04   -0.001633
2023-01-05   -0.015519
2023-01-06    0.021652
2023-01-09    0.001754
2023-01-10    0.008456
                ...
2023-10-10    0.007568
2023-10-11    0.017308
2023-10-12    0.002005
2023-10-13   -0.004708
2023-10-16    0.009752
Length: 197, dtype: float64
```

In [ ]:
```
# View our Cumulative Returns, Daily Returns & Drawdowns

qs.plots.snapshot(portfolioReturns,
                  title='1/N Equal Weighted Portfolio Performance',
                  grayscale=True)
```



In [ ]:
```
#Generate Metrics to Evaluate the performance of our portfolio vs the S&P 500

qs.reports.metrics(
    portfolioReturns,
    benchmark='SPY',
    mode='basic',
    prepare_returns=False
)
```

```
[**********************100%%**********************]  1 of 1 completed
```

	Benchmark (SPY)	Strategy
Start Period	2023-01-04	2023-01-04
End Period	2023-10-16	2023-10-16
Risk-Free Rate	0.0%	0.0%
Time in Market	99.0%	100.0%
Cumulative Return	13.62%	62.15%
CAGR%	11.96%	53.32%
Sharpe	1.28	3.66
Prob. Sharpe Ratio	87.12%	99.97%
Sortino	1.94	6.66
Sortino/√2	1.37	4.71
Omega	1.83	1.83
Max Drawdown	-7.91%	-6.25%
Longest DD Days	104	36
Gain/Pain Ratio	0.23	0.83
Gain/Pain (1M)	1.47	11.65
Payoff Ratio	0.99	1.42
Profit Factor	1.23	1.83
Common Sense Ratio	1.23	2.28
CPC Index	0.66	1.44
Tail Ratio	1.0	1.25
Outlier Win Ratio	3.13	2.11
Outlier Loss Ratio	2.84	2.73
MTD	2.0%	5.89%
3M	-2.95%	7.65%
6M	5.72%	33.51%
YTD	13.62%	62.15%
1Y	13.62%	62.15%
3Y (ann.)	11.96%	53.32%
5Y (ann.)	11.96%	53.32%
10Y (ann.)	11.96%	53.32%
All-time (ann.)	11.96%	53.32%
Avg. Drawdown	-1.79%	-1.64%
Avg. Drawdown Days	14	6

```
Recovery Factor     1.7               7.92
Ulcer Index         0.03              0.02
Serenity Index      0.64              7.0
```

In [ ]:
```
# get tearsheet

qs.reports.html(
    portfolioReturns,
    benchmark='SPY',
    title='1/N Portfolio - Hakeem Lawrence',
    download_filename='1/N Portfolio Eval.html'
)
```

```
[**********************100%%**********************]  1 of 1 completed
```

```
c:\Users\hakee\AppData\Local\Programs\Python\Python39\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```

Results

After reviewing our performance metrics, we can see that the sample portfolio has outperformed the S&P 500 signficantly. Our cumulative return and CAGR was more than 40% that of the S&P. Additionally, our Sharpe ratio was much better than the market. This suggests that our portfolio does a better job at generating higher returns while taking on more risk. This conclusion is also supported by our Serenity Index score. Our results show that the portfolio had a lower risk to to the down side versus the S&P.