MARKOWITZ'S ESSENTIAL FRONTIER THEORY WITH A MONTE CARLO SIMULATION

PYTHON PROJECT FOR FINANCE CLUB UNIPI

Author: Petros Katerinis

Introduction

This modern portfolio theory project implements Markowitz's Essential Frontier theory to construct an optimal portfolio, utilizing a Monte Carlo Simulation to visualize the Efficient Frontier and identify two portfolios, one with the maximum Sharpe Ratio (Top Portfolio) and one with the minimum Volatility (Safest Portfolio). The project was done on two occasions with different weight constraints for each one. On the first occasion the weight constraints implemented were w= [0.2,1], meaning only Long selling, with a minimum weight of 0.2. On the second occasion, where both Long and Short selling were allowed, the constraints were at w= [-1,2]. Also, the number of simulated portfolios was 100.000 which is a big enough number, especially for only 3 assets, to understand the logic of the project and show the results we want in our plots.

Data

We used weekly data for 8 years for each one of the following assets:

1. Nvidia
2. S&P 500
3. Gold

Also, it's important to mention that the risk-free rate used was 3%, very close to average risk-free rate of the very turbulent time period used.

Code

There is one key difference between the two examples, with everything else being almost the same. On the first example (a) we have added a min_weight variable of 0.2 and a remaining_space variable on the for loop of the simulation. Overall though the logic and the code are almost identical.

def portfolio_returns <- the first function simply calculates a portfolio's return, volatility (standard deviation) and its Sharpe Ratio

for I in range(num_portfolios): <- this for loop is where the simulation takes place. The weights constraints are set (different for each variation), then the calculation of returns, volatility and Sharpe ratio takes place, storing the results in arrays to make it easier to work with. While using Vectorization is better and faster than a for loop I believe that the for loop

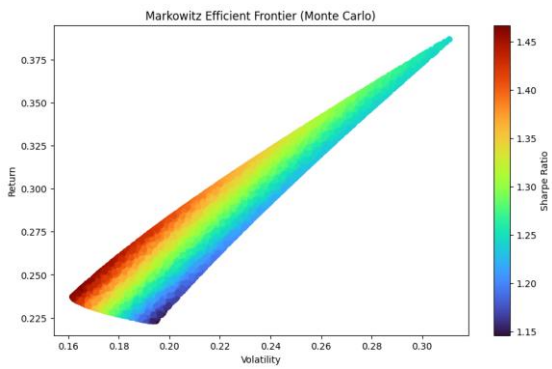
better explains the Monte Carlo simulation logic and makes the program easier to understand, sacrificing speed for simplicity.

Some notable commands that I think should be mentioned to further explain the code are:

columns.tolist() <- has been added to match the data frame column to the ticker list

plt.xlim(0, 1.0), plt.ylim(-0.5, 1.0) <- were used to zoom the plot and not show the outliers

Results

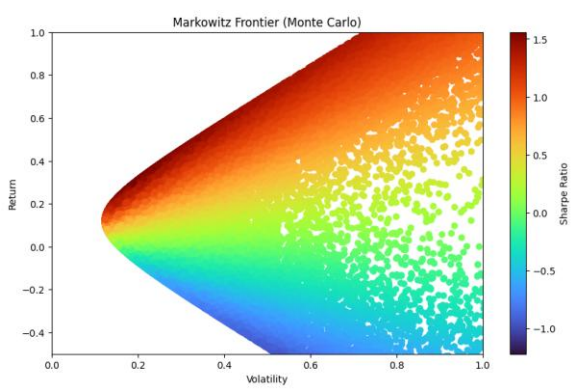



```
TOP PORTFOLIO RESULTS:
Return: 0.2391
Volatility: 0.1630
Sharpe Ratio: 1.4665

WEIGHTS:
GLD: 59.35%
NVDA: 20.63%
^GSPC: 20.01%
```

```
SAFEST PORTFOLIO RESULTS
Return: 0.2365
Volatility: 0.1618
Sharpe Ratio: 1.4617

WEIGHTS:
GLD: 58.88%
NVDA: 20.05%
^GSPC: 21.06%
```

(a)

On the first variation the results are almost identical for both Portfolios and we only see the upper part of the Markowitz frontier and not the entire parabola. This happens due to the weight constraints set, of w= [0.2,1], which create a small feasible region of examples. Gold, having excellent returns, low risk and providing diversification to the portfolio, the program allocates the highest possible weight, while the extremely high returns, but also high volatility Nvidia and the more neutral S&P500 are at the minimum allowed weight.




```
TOP PORTFOLIO RESULTS:
Return: 0.3108
Volatility: 0.1994
Sharpe Ratio: 1.5587

WEIGHTS:
GLD: 90.24%
NVDA: 35.26%
^GSPC: -25.50%
```

```
SAFEST PORTFOLIO RESULTS
Return: 0.1193
Volatility: 0.1235
Sharpe Ratio: 0.9657

WEIGHTS:
GLD: 58.99%
NVDA: -8.26%
^GSPC: 49.27%
```

(b)

Now we are in more familiar “territory” as we see the famous Markowitz bullet. Starting again with the Portfolio results, the Safest portfolio shorts NVDA which completely makes sense due to its high volatility as previously mentioned, to make space for the safer assets. The Top portfolio continues the trend set by the previous Top portfolio consisting of mostly GLD and some NVDA, emphasizing the astronomical returns of this asset, while this time shorting the SP500, further proving that the latter’s returns are not even close to the same ballpark as the ones set by the others.