|  |  |
| --- | --- |
| Portfolio Analysis with python  **Specialised Certificate in Data Analytics for Finance** | **Project Report**  **Real world portfolio optimization project utilizing Python to gain insights from a financial dataset.**  **Neil Conlon**  **Specialised Certificate in Data Analytics for Finance** |

Project Report

The first task was to setup a GitHub repository for the project. [<https://github.com/classredefined/UCDPA_NeilConlon>]

This allowed me to store, manage & track progress from inception to completion. Having completed setting up the repository, I downloaded VSCode as my selected IDE (Integrated Development Environment).

It was then imperative to find the most suitable data source for the Portfolio Analysis. Having researched various finance blogs, I opted for Yahoo Finance as my data source for historical stock information. An open-source library, ‘yfinance’ enabled me to import data directly in pandas dataframes/series which streamlined the data cleaning process before I could start the analysis. Alternative methods would have required me to manually convert lengthy JSON’s into dataframes which is time consuming.

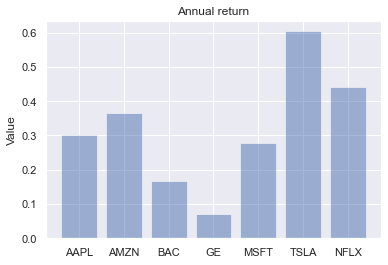
Given my level as a novice with Python upon commencing the course, I decided that my best approach was to begin with Introductory Python on the Datacamp platform. Once I had become comfortable with the basics (types/variables/lists/functions/NumPy), I then proceeded with the customized course designed by the UCD Academy which focused on financial datasets.

From here, I worked through the programme at my own pace, one Datacamp course at a time & saved all the code that I believed would assist me with the Portfolio Analysis.

As illustrated in the accompanying code, the first seven lines of the code are commands to install the packages required for the analysis. CVXPY is a Python-embedded modeling language for convex optimization problems & I stumbled upon it on a python blog (https://pypi.org/project/cvxpy/). I had worked with all the other packages/libraries on DataCamp.

Once these packages had been successfully installed, I imported my CSV file from Yahoo Finance, onto VSCode & using the **pd.DataFrame()** command, I was able to convert the CSV file into a Pandas DataFrame.

My next step was to carry out a basic analysis on the daily stocks returns (The first 5 daily returns for a start) of the 7 large cap stocks I selected from the dataset that I held in the portfolio. I wanted to illustrate this analysis from a risk averse investors perspective hence the profile of my stock selection (All large caps). As you will see in the graph below, the x-axis labels represent stocks from the list which I created:

tickers=['AAPL','AMZN','BAC','GE','MSFT','TSLA','NFLX'].

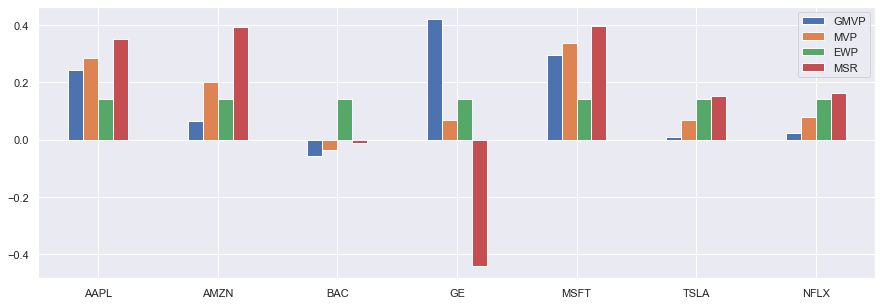
The y-axis represents the % daily increase in stock returns. Note the use of the mean function on NumPy multiplied by 250 (number of annual trading days) to calculate the daily mean. Also note the DataFrame of the last 5 days of stock returns: print(returns.tail(5))

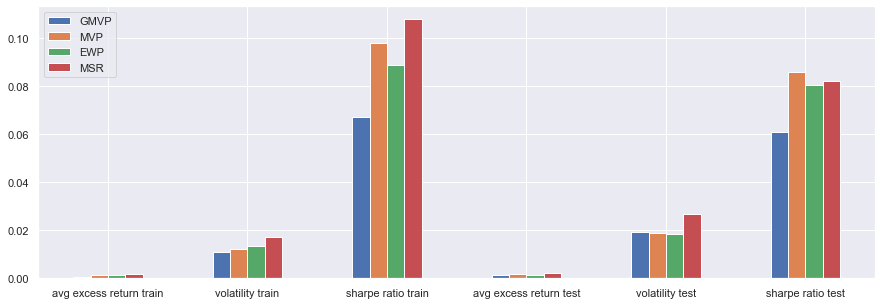
It should be no surprise from our data that Tesla & Netflix have been the best performers over the selected timeframe (01/01/10 – 12/07/21) given their profile as growth stocks.

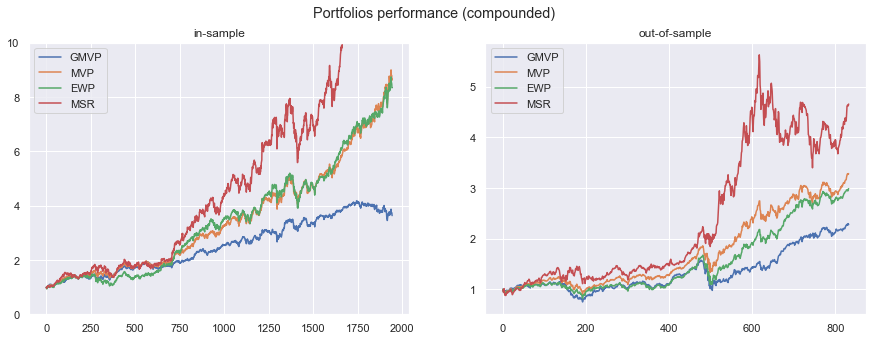
My original plan was to use Quantopian's ‘Pyfolio’ tool, for the analysis but I was unable to format my dataset for it to function properly. The ‘tear-sheet’ which can be generated provides a detailed breakdown of the portfolio including multiple alternative risk measures including the Sortino & Calmar ratios. This difficulty with formatting is a major drawback for what is otherwise an extremely powerful package, but it is my intention to master it after this project where time constraints are not a factor.

The train-test split procedure is used to predict stock returns in the future using the entire dataset imported. I decided that a 70/30 split for the training/test data would be appropriate for this model. This 70/30 split allocates 70% to the training set & 30% to the test set which will forecast the future stock returns. We are basically leveraging data of past stock returns to forecast the future.

Having carried out the train-test split, next step in my analysis was run a comparison between popular portfolio strategies, specifically; an equal weighted portfolio (EWP), a minimum variance portfolio (MVP), an MVP with short selling(underweight a security) & the maximum Sharpe-ratio portfolio (MSR). Below you will the graph with optimal weights of securities across the various strategies. Unsurprisingly, the securities with the least variance in returns dominate the GMVP (AAPL, GE & MSFT) while the MSR results in a large short position in GE stock. The results in other portfolios are largely as we would have expected. Note also: the EWP has the same weight for all 6 stocks.

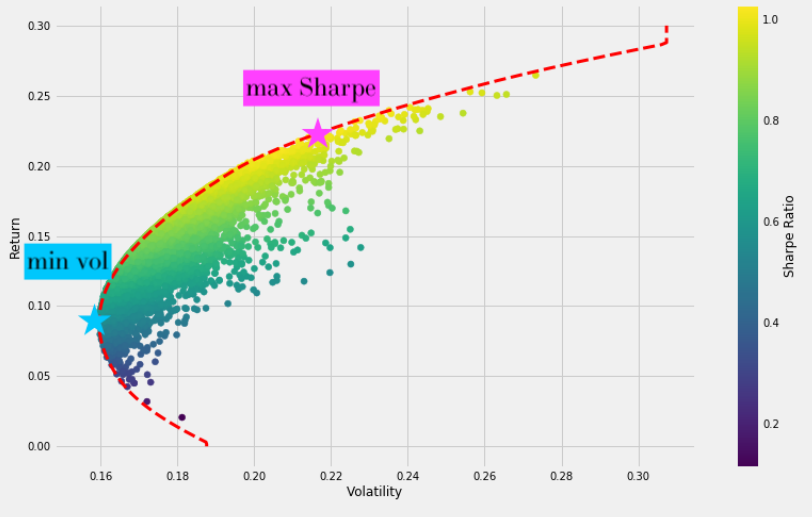


The next task in our portfolio analysis was to compare the train & test results from the train-test split discussed earlier. As is illustrated below, there is a slight discrepancy in the respective output, but this is to be expected. Overfitting is the most common issue when the test data fits the training data too closely. This occurs when models are usually too complex (high number of explanatory variables relevant to the number of observations). I do not believe that overfitting is an issue in this instance as the results explain the relationship between the variables sufficiently without being overly conclusive. There is a slight variation between the volatility & Sharpe ratio output between the train & test set as illustrated.

In the final part of the portfolio analysis, we modeled the compounded portfolio returns for all 4 strategies against one another, using the same 70/30 train-test split. As illustrated in the graphs below, the MSR portfolio which included a short position in GE stocks outperformed all the other portfolios when measured using the ‘in-sample’ set but showed a significant dip in the ‘out-of-sample’ test set. This portfolio which Maximizes the Sharpe Ratio (Illustrated in lower diagram in pink) is where the optimal risk/return profile exists for the ‘in-sample’ set when back tested. It does however carry more inherent risk than the other portfolios & is thus prone to periods of greater volatility than any portfolio positioned to the left along the efficient frontier. It is thus only suited to investors that have a high-risk threshold.

The GMVP portfolio is the one on the extreme left along the efficient frontier which exhibits the lowest level of volatility. It displays consistency in both the ‘in-sample’ & ‘out-of-sample’ sets as illustarated above. This global minumum variance porfolio also has the lowest returns as we would also expect. This portfolio is only suitable for risk-adverse inestors that have little or no capacity to take on extra risk.

For the minimum variance portfolio, we set the expected return to equal to the EWP portfolio. Although we did include the short-selling option in building this portfolio, the allocation does not have the short position in GE stock. This portfolio shows higher returns than the GMVP with a similar risk profile. Theoretically, the MVP is the optimal portfolio for risk-adverse investors.

Lastly, the Equally weighted portfolio outperformed my expectations as in most cases, it is a very imprudent approach to building a portfolio. Here as you can see in both the ‘in-sample’ & out-of-sample’ returns, it almost matches the MVP discussed in the previous paragraph. This is largely down to the fact that all the stocks have a similar growth profile over the period which was back tested.

This project opened my eyes up to the vast number of applications that Python can execute in finance. From portfolio analysis such as my project to risk management, credit risk modelling & even quantitative risk modelling . I now fully comprehend the importance that coding languages play in manipulating datasets to generate insights & answer commonly posed questions by asset managers. While there is still an entire universe of functions for me to learn with Python, I am pleased with my progress over the past 16 weeks with something I had little experience with before enrolling. The Datacamp platform is exceptional & enabled me to review code I had difficulty recalling while doing the analysis.