*Table of Contents*

[*1.Overview of the Project: 1*](#_Toc512377429)

[*1.1. Steps involved: 2*](#_Toc512377430)

[*2. Importing the required libraries 2*](#_Toc512377431)

[*3.Data Pre-processing: 3*](#_Toc512377432)

[*3.1. Collecting the list of Companies in S&P500 companies: 3*](#_Toc512377433)

[*3.2. Fetching the data from Yahoo Finance: 4*](#_Toc512377434)

[*3.3. Calculating Annual return and risk of all stocks in S&P 500: 5*](#_Toc512377435)

[*4.K-Means Clustering 6*](#_Toc512377436)

[*4.1. Applying Elbow Method: 6*](#_Toc512377437)

[*4.2. Applying K-means clustering 7*](#_Toc512377438)

[*4.3. Visualising the Clusters using Tableau 8*](#_Toc512377439)

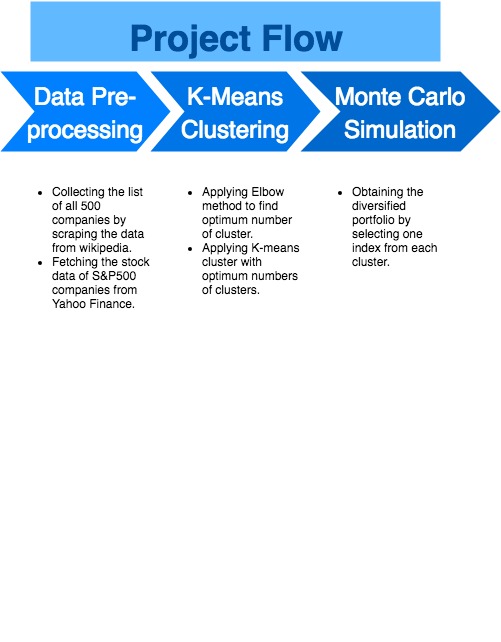
[*5. Monte Carlo Simulation: 10*](#_Toc512377440)

[*5.2. Calculating daily return and creating covariance matrix: 11*](#_Toc512377441)

[*5.3. Running Monte Carlo simulation: 12*](#_Toc512377442)

# 

# ***1.Overview of the Project:***



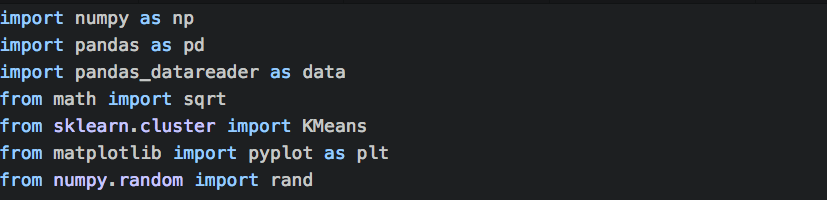
Source: Bhugul, Aaruni. 2018, tool used: [www.draw.io](http://www.draw.io)

## ***1.1. Steps involved:***

For this project, I will be creating a script in Python 3 to perform the following tasks:

1. Download stocks data of all companies listed in S&P 500 index.
2. Calculate their annual returns and annual risk.
3. Perform K-Means clustering algorithm to divide the stocks into distinct groups based on their annual returns and risk.
4. Design an optimised portfolio using Monte Carlo simulation by selecting return and risk of index (or company).

***2. Importing the required libraries:***



# ***3.Data Pre-processing:***

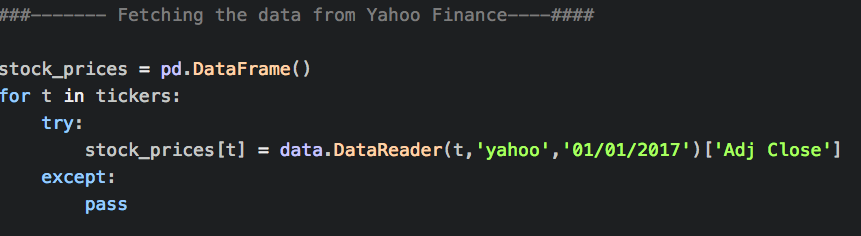
## ***3.1. Collecting the list of Companies in S&P500 companies:***

We will collect the data by running our import and then creating a simple data download script that scrapes the Wikipedia to collect the tickers (symbols) for all the indices of the S&P 500.

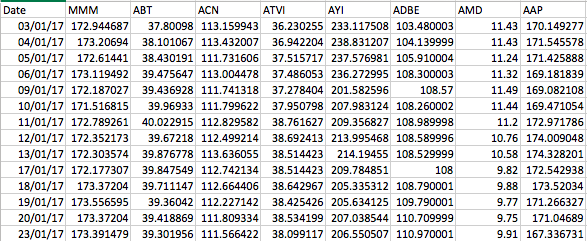


## ***3.2. Fetching the data from Yahoo Finance:***

Now we will collect the ‘Adjacent closing’ price of all S&P 500 companies using the Yahoo finance API.



An overview of the data frame that will be generated after running the above script:



Usually, it takes 15-20 minutes to run this script on my laptop, hence I personally prefer to fetch the data from yahoo finance and then store the data into a csv file, so that I don’t need to wait for so long every time I run this file.



[Here](https://github.com/aarunibhugul/Become-an-Investment-Banker-Clustering-Monte-Carlo-Simulation-/blob/master/S%26P500-06-April-2017.csv), you can find the GiHub link to the 'csv’ generated by the above code. Henceforth, I will be using the above-mentioned file as a data source for further calculations.

## ***3.3. Calculating Annual return and risk of all stocks in S&P 500:***

***1. Annual Return:***

Annual return is the return an index provides over a period of time, expressed as a time-weighted annual percentage. Also known as an annualized return, the annual return expresses the stock’s increase/decrease in value over a designated period of time. In order to calculate an annual return, information regarding the current price of the stock and the price at which it was purchased are required. If any splits have occurred, the purchase price needs to be adjusted accordingly. Once the prices are determined, the simple return percentage is calculated first, with that figure ultimately being annualized.

To calculate the annual return, we need to calculate the simple daily return of each stock. The formula for calculating the annual return of any stock is:

Return on stock

Based on the return obtained we can now calculate the annual return using the formula:

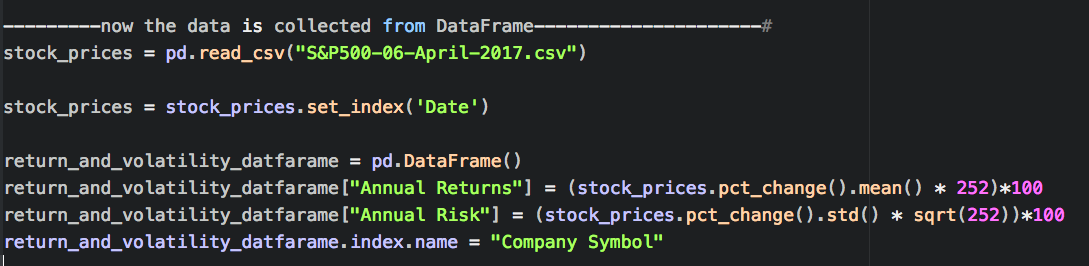
Annual return on stock =

***2. Annual Risk:***

Annual risk refers to the volatility or the fluctuation in stock prices of a particular stock. This risk return helps investors for taking on the relatively higher or lower risk of while investing in a stock. As a rule, high-risk investments are compensated with a higher return.

Annual risk of stock =

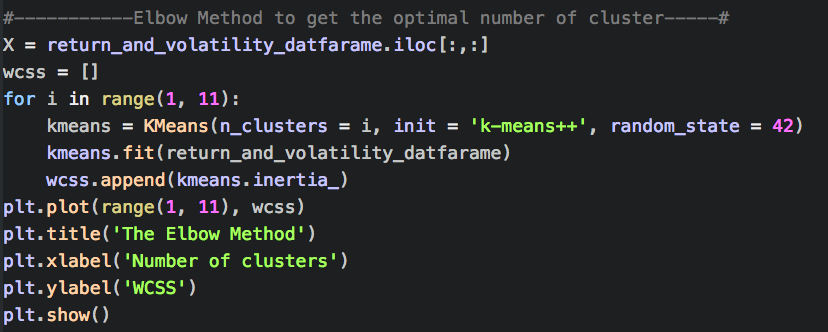
Using pandas, we will calculate the annual risk and return and store the result in a new data frame as follows:



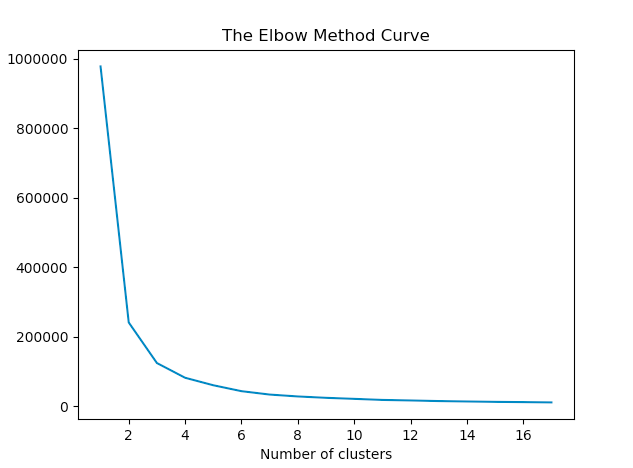
# 

# ***4.K-Means Clustering***

## ***4.1. Applying Elbow Method:***

Now let’s start applying K-Means clustering. The first step involved in K-Means clustering is calculating the optimum number of clusters to decide how many clusters do we need segment the data of all S&P 500 companies. The optimum number of clusters can be derived through the “Elbow method” which plot the WCSS (within cluster sum of square) on Y axis and number of clusters on X-axis. Now let’s run the elbow method:

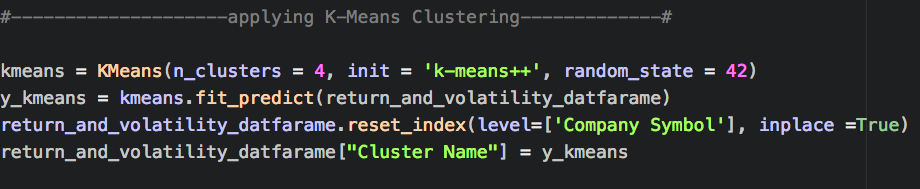
The above piece of code will show the following graph:



After plotting the result of “Number of clusters" vs "WCSS" we can notice that the number of clusters reaches 4 (on the X axis), the reduction the within-cluster sums of squares (WCSS) begins to slow down for each increase in cluster number. Hence, the optimal number of clusters for this data comes out to be 4. Therefore, let’s take number of cluster for k means = 4.

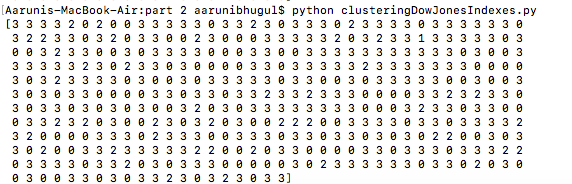
## ***4.2. Applying K-means clustering***

Now let’s apply kmeans clustering with number of clusters as 4. I am passing the ‘init’ argument as “k-means++” to avoid random initialisation trap.

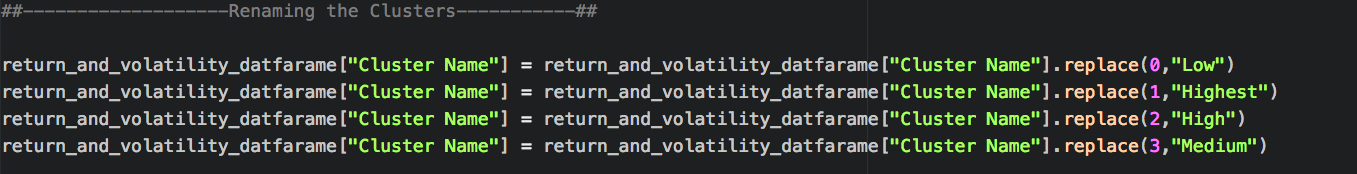
******

when we print y\_kmeans we can see the clustered are created as with names “0,1,2 and 3”.



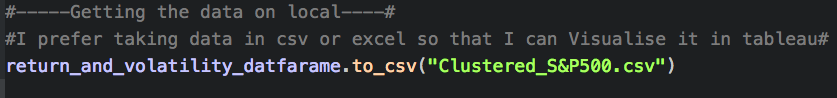


now let’s replace the numeric value of y\_kmeans, 0,1,2 and 3 with string “low”, “medium”, “high” and “higher” respectively:



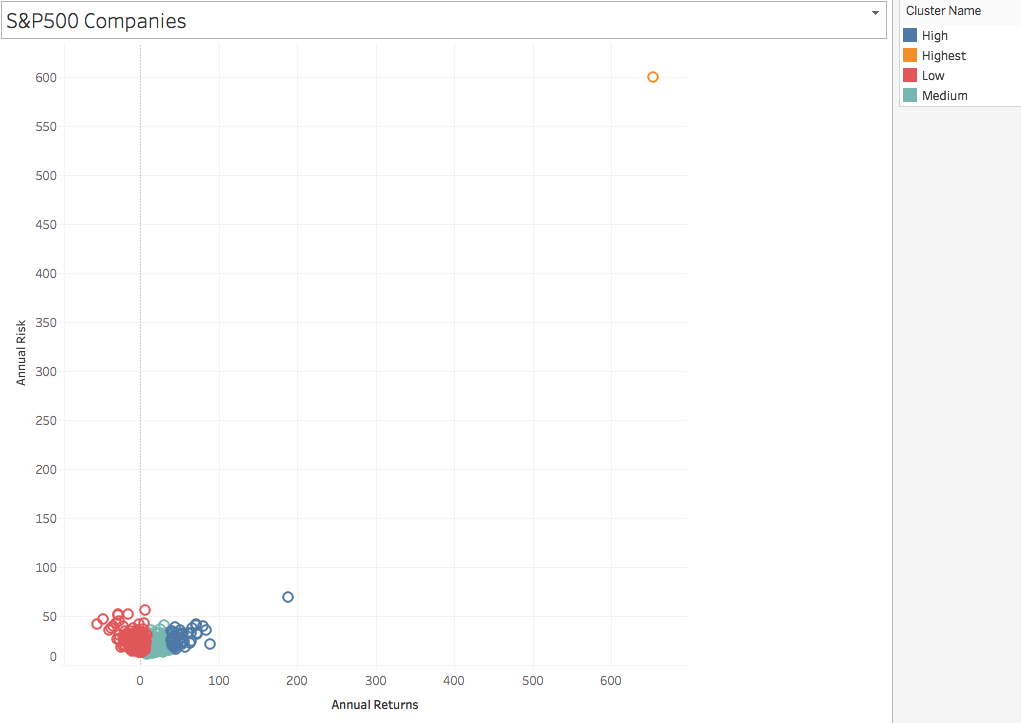
## ***4.3. Visualising the Clusters using Tableau***

Before visualising the data, let’s first insert the “return\_and\_volatility\_datafarame” into the csv file, so that we can treat the it as the data source for tableau visualisation:



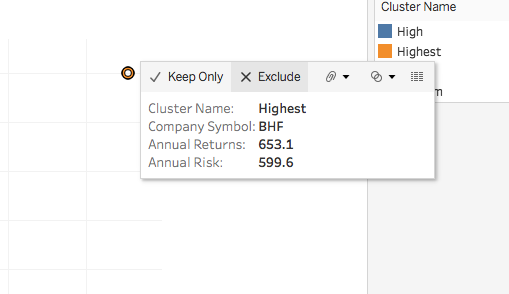
[Here](https://github.com/aarunibhugul/Become-an-Investment-Banker-Clustering-Monte-Carlo-Simulation-/blob/master/Clustered_S%26P500.csv) is a link to the “Clustered\_S&P500,csv” file of my GitHub repository

Now let’s visualise the data frame in Tableau:

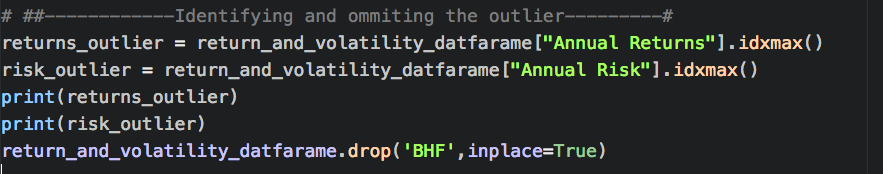


After visualising, I have discovered an outlier. In case of stock data, outliers can be interesting for those investors who would like to take higher risk in exchange of higher return.

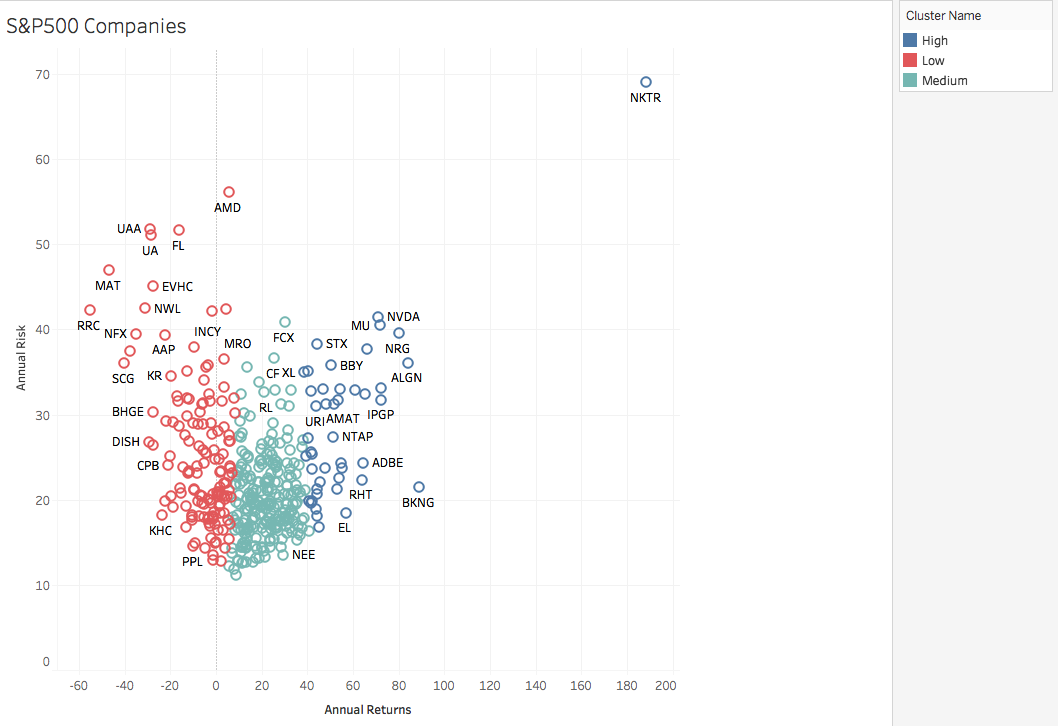
We can the obtain the name of the company easily using Tableau and then exclude it, as follows:



Using python I would detect the outlier using the idxmax() function of pandas and exclude it using the ‘drop’ function as follows:



On tableau, after removing the outlier the data points appear as:

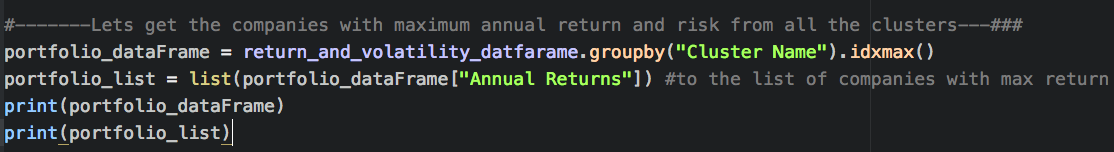


# ***5. Monte Carlo Simulation:***

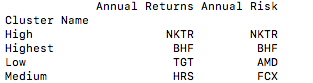
Now that we have created the segmented (clustered) the stocks on the basis of their annual risk and return. It’s time to create a portfolio that gives the optimized combination of number (or weight) of shares of a stock that should be present in a portfolio. Usually, we can select any number of companies of our interest from all three/four clusters. For this example, I am selecting companies with maximum annual return from each cluster.

***5.1. Extracting the list of companies to create a portfolio:***

Let’s create our portfolio selecting those companies from each cluster with highest annual return. We can do this easily using ‘groupby’ and ‘idxmax’ function of panda:



the result for ‘portfolio\_dataframe’ on my console is:

******

and the result for portfolio list on my console is :

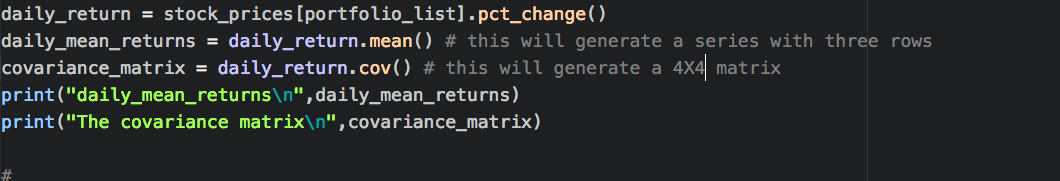
******

Hence, our portfolio will consist of the following companies:

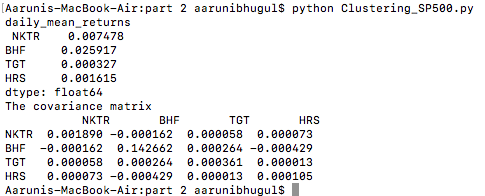
1. NKTR
2. BHF
3. TGT
4. HRS

## ***5.2. Calculating daily return and creating covariance matrix:***

I can still remember, when I used to work for a financial client me and my colleague had designed an algorithm to calculate the portfolio based on the “annual return”. It had resulted in drastic change in the calculations and had resulted into a huge mess and followed by escalation. But it is wisely said, “Everything happens to the best of us”, hence now I can remember very well that we need to calculate the “daily return” and not the “annual return” to design a portfolio. So, let’s first calculate the daily return for each company listed in our portfolio:



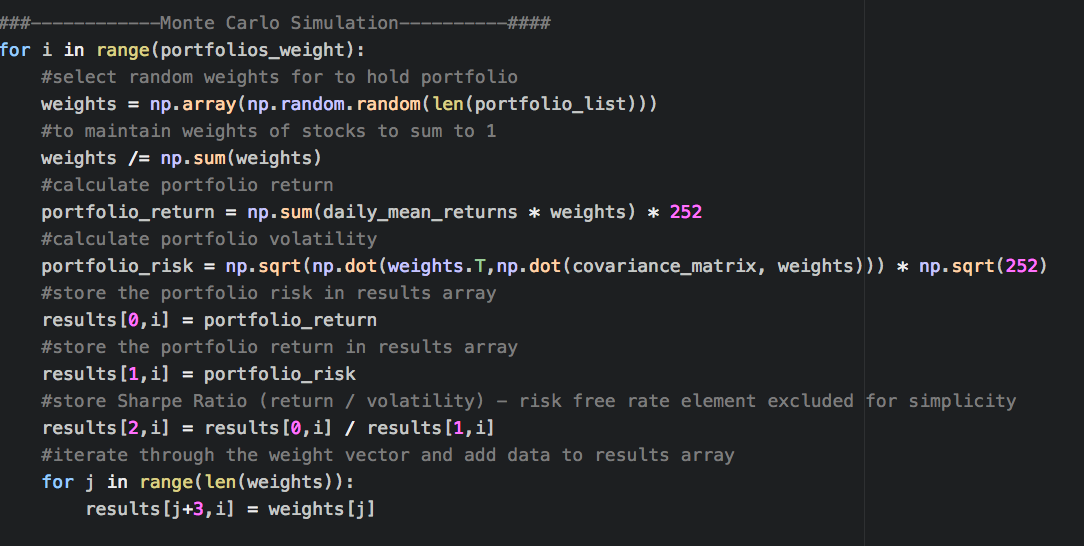
The above print commands will generate the following results:



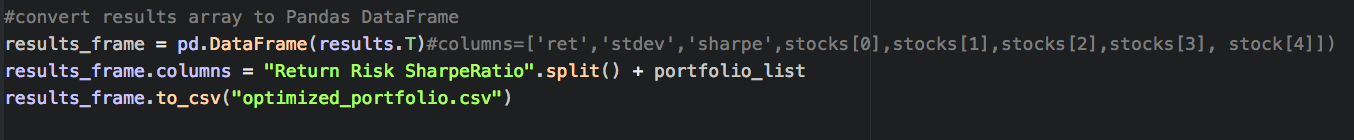
## ***5.3. Running Monte Carlo simulation:***

We can manually change the weights of the stocks (or companies) and run the script every time and look for the expected return and risk/volatility of that particular set of weights. But that’s the most manual way to proceed and it might take ages to design a portfolio and there are technically an infinite number of sets of portfolio weights to test, so obviously that’s not practically feasible.

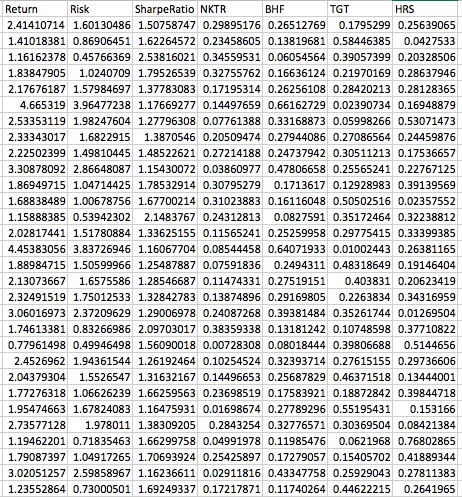
And this is where Monte Carlo simulation comes to our rescue. Monte Carlo simulation can run 1000s of different randomly generated weights for the individual stocks and then calculate the expected return, expected volatility and Sharpe Ratio for each of the randomly generated portfolios. The code below is the most complex part of this entire project and is very to understand if we are able to understand the concept of weighted average (sum of all weight should not exceed by more than 100%). So, let’s proceed with Monte Carlo simulation:



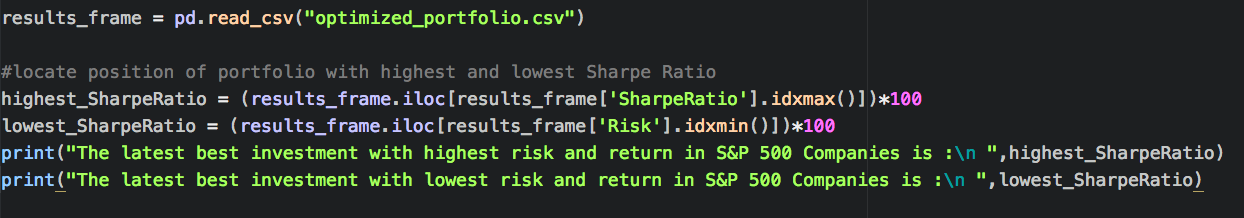
The variable ‘result’ is a numpy array with 7 rows and 5000 columns. Let’s put the numpy array into the pandas ‘results\_frame’ dataframe and name the columns. Finally, we will extract the entire data of results\_frame into a csv. [Here](https://github.com/aarunibhugul/Become-an-Investment-Banker-Clustering-Monte-Carlo-Simulation-/blob/master/optimized_portfolio.csv) the GitHub link to the ‘results\_frame’.



Below, you can find the preview of the result\_frame:



***Obtaining the portfolio with highest and lowest Sharpe ratio:***

******

The above code will give the percentage of stock of the respective company a portfolio should hold along with the subjected risk and return.