Portfolio Optimization with Financial Filters

Summary

The underlying idea behind this work is not simply to feed the Markowitz with a sample of stocks in which most of the resulting portfolio is very likely to belong to a specific sector, but to pass the best ones according to twelve filters. The first step was to obtain a reduced list of stocks for each industry sector that have been filtered based on several financial criteria and then applying Markowitz efficient frontier algorithm. The objective being to obtain a portfolio which has a high likelyhood of having exposure to several sectors and by extension achieving a higher than normal level of diversity. This idea does not guarantee that the resulting portfolios will be sufficiently diversified, but it might be more likely to create a portfolio compounded by different sectors.

It should be clarified that the use of two Python classes is not presented in this notebook to facilitate the understanding of our results and not to overwhelm with so much code. These classes are Get_Stocks and EF, the former functions as ETL and point system and the latter as the algorithm that will receive the output from Get_Stocks and define the two portfolios: maximum return and minimum volatility.

In [5]:
```
#Import the libraries and functions to be used in this notebook
import pandas as pd
import numpy as np
import yfinance as yf
import collections
from Get_Stocks import*
from EF import*
```

This is a Function that first clears the data and then divides the entire SP500 by sectors assigning points for every filter passed by each stock. The time interval considered was "2015-1-1" until now.

In [6]:
```
stock_filter = StockFilter()
stock_filter_particular = StockFilterParticular()

# Call the methods on the instance
data = down_stocks()
data = stock_filter.filtro_nulo(data)
data, sharpe_ratios = stock_filter.filtro_sharpe(data)
data,stock_counts = stock_filter_particular.filtro_sector(data, sharpe_ratios)
```

```
[**********100%**********]  503 of 503 completed

2 Failed downloads:
- BF.B: No data found for this date range, symbol may be delisted
- BRK.B: No timezone found, symbol may be delisted
```

In [7]:
```
#
data = stock_filter_particular.top_sector(data, stock_counts)
data1 = data[0]
data2 = data[1]
```

Here we can see one of the Get_Stocks class output. The output consists in how many stocks passed our filters. The method explanation is the following one: We take the list of stocks and the number of filters they passed and append to which sector each stock corresponds, then we choose for each sector the stocks that are in the 0.8 quantile of the sample based on the number of filters passed and we eliminate from the main dataframe the stocks that do not pass that restiction.

In [8]:
```
pd.set_option('display.max_rows',None)

data2
```

Out[8]:

	Stock	Number	Sector	Quantile_Number
0	A	8	Healthcare	6.0
10	EW	7	Healthcare	6.0
17	JNJ	7	Healthcare	6.0
23	RMD	7	Healthcare	6.0
24	RVTY	7	Healthcare	6.0
25	STE	8	Healthcare	6.0
34	REGN	7	Healthcare	6.0
36	VRTX	7	Healthcare	6.0
37	WAT	7	Healthcare	6.0
57	AAPL	9	Technology	8.0
61	AMAT	10	Technology	8.0
63	ANET	11	Technology	8.0
64	ANSS	9	Technology	8.0
66	AVGO	12	Technology	8.0
74	IT	9	Technology	8.0
76	KEYS	9	Technology	8.0
77	KLAC	11	Technology	8.0
79	LRCX	9	Technology	8.0
80	MPWR	10	Technology	8.0
81	MSFT	10	Technology	8.0
94	TXN	10	Technology	8.0
104	QCOM	9	Technology	8.0
126	ACGL	10	Financial Services	8.0
132	BRO	9	Financial Services	8.0
133	CBOE	9	Financial Services	8.0
136	FDS	11	Financial Services	8.0
139	MA	10	Financial Services	8.0
141	MKTX	11	Financial Services	8.0
142	MMC	10	Financial Services	8.0
149	SPGI	9	Financial Services	8.0
150	V	11	Financial Services	8.0
152	WTW	9	Financial Services	8.0
185	CHD	9	Consumer Defensive	8.0

	Stock	Number	Sector	Quantile_Number
190	HSY	10	Consumer Defensive	8.0
195	MNST	9	Consumer Defensive	8.0
221	CPRT	9	Industrials	8.0
222	CSX	10	Industrials	8.0
223	CTAS	10	Industrials	8.0
224	DE	10	Industrials	8.0
227	EXPD	9	Industrials	8.0
228	FAST	10	Industrials	8.0
229	GWW	10	Industrials	8.0
231	IEX	10	Industrials	8.0
238	PAYX	10	Industrials	8.0
240	POOL	9	Industrials	8.0
255	SNA	9	Industrials	8.0
304	ALB	10	Basic Materials	8.6
308	MLM	9	Basic Materials	8.6
311	STLD	9	Basic Materials	8.6
313	CF	9	Basic Materials	8.6
325	CMG	9	Consumer Cyclical	8.0
327	DHI	9	Consumer Cyclical	8.0
335	NVR	10	Consumer Cyclical	8.0
337	PHM	10	Consumer Cyclical	8.0
343	ULTA	9	Consumer Cyclical	8.0
373	EXR	12	Real Estate	9.0
375	MAA	11	Real Estate	9.0
377	PSA	11	Real Estate	9.0
390	ATVI	10	Communication Services	9.0
392	GOOG	10	Communication Services	9.0
393	GOOGL	10	Communication Services	9.0
416	VLO	12	Energy	11.0

In [10]:
```
table = data1
noa = len(table.columns)
# Assuming you have the necessary inputs:
log_returns = np.log(table / table.shift(1))
mean_returns = log_returns.mean()
cov_matrix = log_returns.cov()
```

```
num_portfolios = 30000
#We set the risk free rate considering this:
#As of June 29, 2023, the best CD interest rate is 5.65% APY with NASA Federal C
risk_free_rate = 0.0565

# Create an instance of the PortfolioOptimization class
portfolio = PortfolioOptimization(table)

# Call the methods of the class using the instance
portfolio.display_calculated_ef_with_random(mean_returns, cov_matrix, num_portfo
```

```
--------------------------------------------------------------------------------
Maximum Sharpe Ratio Portfolio Allocation

Annualised Return: 0.25
Annualised Volatility: 0.23


            AAPL  ANET  ATVI   AVGO   CHD   CPRT  CTAS    DE   MSFT  POOL
allocation  3.22  3.27  1.49  13.34  3.47  45.37  6.72  3.16  18.06  1.89
--------------------------------------------------------------------------------
Minimum Volatility Portfolio Allocation

Annualised Return: 0.11
Annualised Volatility: 0.15


            ATVI   CBOE    CF    CHD  CMG    DE  EXPD   GWW   HSY    JNJ  \
allocation  5.89  11.39  0.41  15.61  5.0  1.06  7.07  0.17  8.02  28.09

            MKTX    PSA  REGN  RVTY   SNA   WTW
allocation  0.17  10.37  1.44  0.09  0.79  4.41
```

In this section we can see which stocks make up the two different portfolios. It is logical that the maximum return portfolio is less diversified than the minimum volatility portfolio in terms of sectors and number of stocks. This is because diversification tends to decrease returns at the same time as volatility decreases. To test this, there is a method called the "Law of Diversification" which states that as the number of stocks in a portfolio increases, volatility will decrease. The key is to find the number of stocks where the decrease tends to be marginal.

In [11]:
```
max_return = ['AAPL', 'ANET', 'ATVI', 'AVGO', 'CHD', 'CPRT', 'CTAS', 'DE', 'MSFT

data_dict = {'Sector':{}}

for symbol in max_return:
    ticker = yf.Ticker(symbol)
    stock_info = ticker.info
    sector = stock_info.get('sector')
    data_dict['Sector'][symbol] = sector

data_dict
```

Out[11]:
```
{'Sector': {'AAPL': 'Technology',
  'ANET': 'Technology',
  'ATVI': 'Communication Services',
  'AVGO': 'Technology',
  'CHD': 'Consumer Defensive',
  'CPRT': 'Industrials',
  'CTAS': 'Industrials',
  'DE': 'Industrials',
  'MSFT': 'Technology',
  'POOL': 'Industrials'}}
```

In [12]:
```
min_volatility  = ['ATVI', 'CBOE', 'CF', 'CHD', 'CMG', 'DE', 'EXPD', 'GWW', 'HSY

data_dict2 = {'Sector':{}}

for symbol in min_volatility:
    ticker = yf.Ticker(symbol)
    stock_info = ticker.info
    sector = stock_info.get('sector')
    data_dict2['Sector'][symbol] = sector

data_dict2
```

Out[12]:
```
{'Sector': {'ATVI': 'Communication Services',
  'CBOE': 'Financial Services',
  'CF': 'Basic Materials',
  'CHD': 'Consumer Defensive',
  'CMG': 'Consumer Cyclical',
  'DE': 'Industrials',
  'EXPD': 'Industrials',
  'GWW': 'Industrials',
  'HSY': 'Consumer Defensive',
  'JNJ': 'Healthcare',
  'MKTX': 'Financial Services',
  'PSA': 'Real Estate',
  'REGN': 'Healthcare',
  'RVTY': 'Healthcare',
  'SNA': 'Industrials',
  'WTW': 'Financial Services'}}
```

In [ ]: