ETF Portfolio Optimization

Shivakumar Ranganathan

I. **Problem Statement: Portfolio Optimization**

II. **Exploratory Data Analysis**

III. **Benchmarking [PyPortfolioOpt vs Monte Carlo vs Genetic Algorithm]**

IV. **Grid Search, Cross-Validation and Hyper Parameter Tuning**

V. **Optimal Portfolio ['SPY','XLF','QQQ','XLE',GDXJ]**

VI. **Optimal Portfolio ['QQQ','ARKK','XLK','XLV','XBI','IBB', 'IYR','XLRE','VNQ', 'XLF','KRE','KBE', 'XLE','XOP','ICLN']**

VII. **Conclusions**

I. Problem Statement: Portfolio Optimization

Objective:

- Maximize Sharpe: $S_a = \left(\frac{w^T.\mu}{\sqrt{w^T.\Sigma.w}}\right)$
- Minimize Volatility: $\sigma_a = \sqrt{w^T.\Sigma.w}$
- Maximize Returns: $\mu_a = w^T.\mu$

Constraints:

- Sum of weights must equal 1: $w^T.I = 1$
- Weights must be between 0 and 1: $0 \leq w_i \leq 1$

What values of w_i would maximize (or minimize) the objective function subject to the given constraints?

- Approach 1: Efficient Frontier optimization using **PyPortfolioOpt**
- Approach 2: **Monte Carlo Simulations**
- Approach 3: Optimization using **Genetic Algorithm**

II. Exploratory Data Analysis

- Use ETF screeners to identify tickers with most trading volume to form the portfolio from various sectors:

https://etfdb.com/etfs/sector/

Sector	AUM Rank	+/-	Assets Under Management ($MM)	# of ETFs
Technology	1	-	$422,157.16	98
Healthcare	2	-	$107,037.89	62
Real Estate	3	-	$88,491.77	47
Financials	4	-	$83,312.99	50
Energy	5	-	$72,840.33	68

ticker=['QQQ','ARKK','XLK','XLV','XBI','IBB', 'IYR','XLRE','VNQ', 'XLF','KRE','KBE', 'XLE','XOP','ICLN']

- Other strategies include screening ETFs with desired historic returns and Sharpe ratio or personal investment experience.

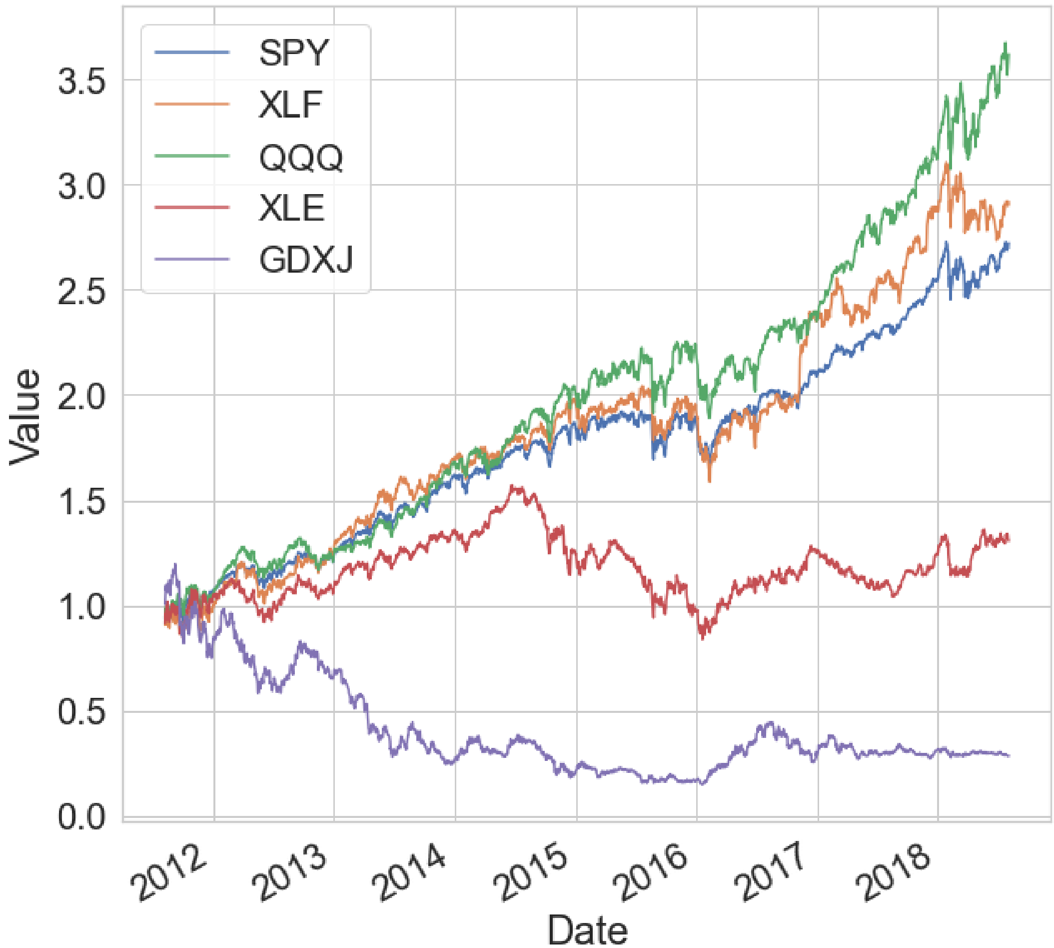
II. Exploratory Data Analysis

ticker=['SPY','XLF','QQQ','XLE', 'GDXJ']

Expect less (zero) weight for underperforming tickers

II. Exploratory Data Analysis

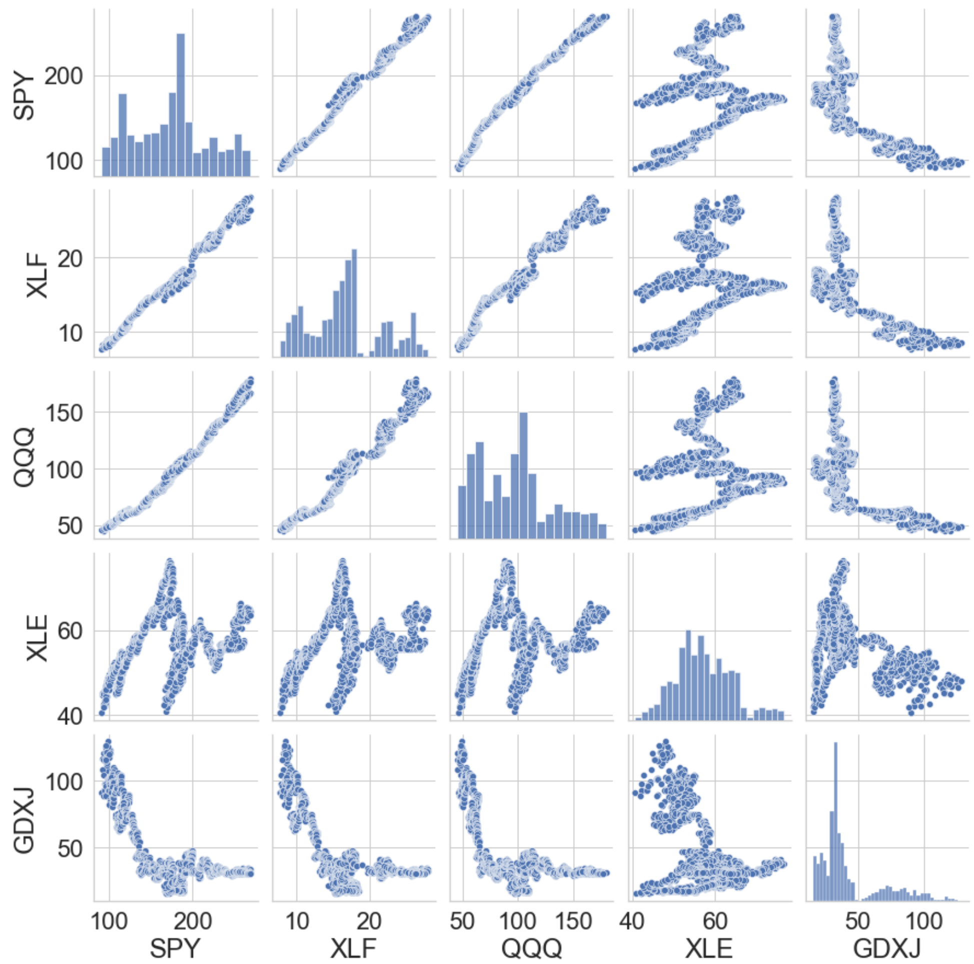
ticker=['SPY','XLF','QQQ','XLE', 'GDXJ']



Expect less (zero) weight for underperforming tickers

II. Exploratory Data Analysis

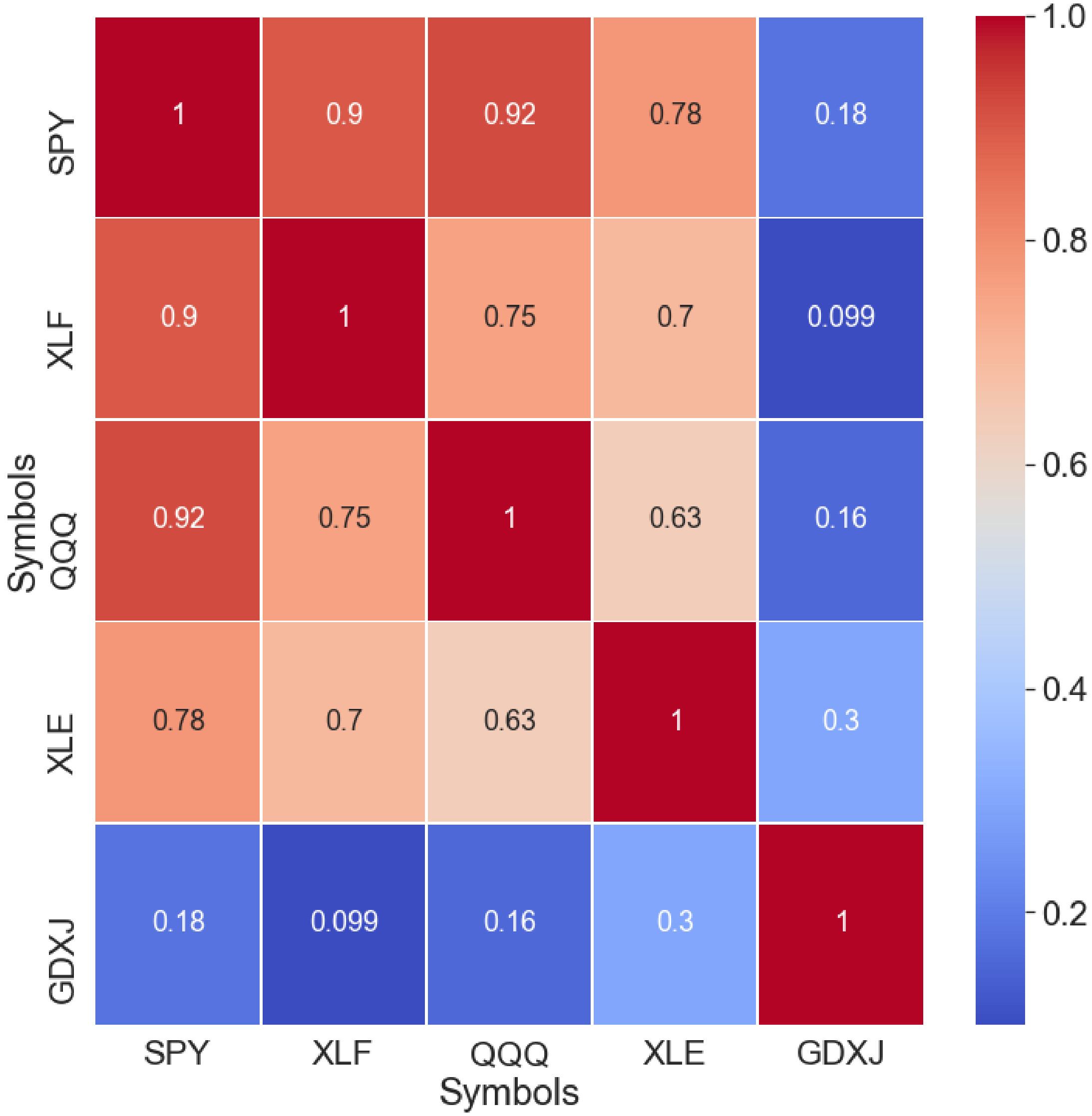
ticker=['SPY','XLF','QQQ','XLE', 'GDXJ']



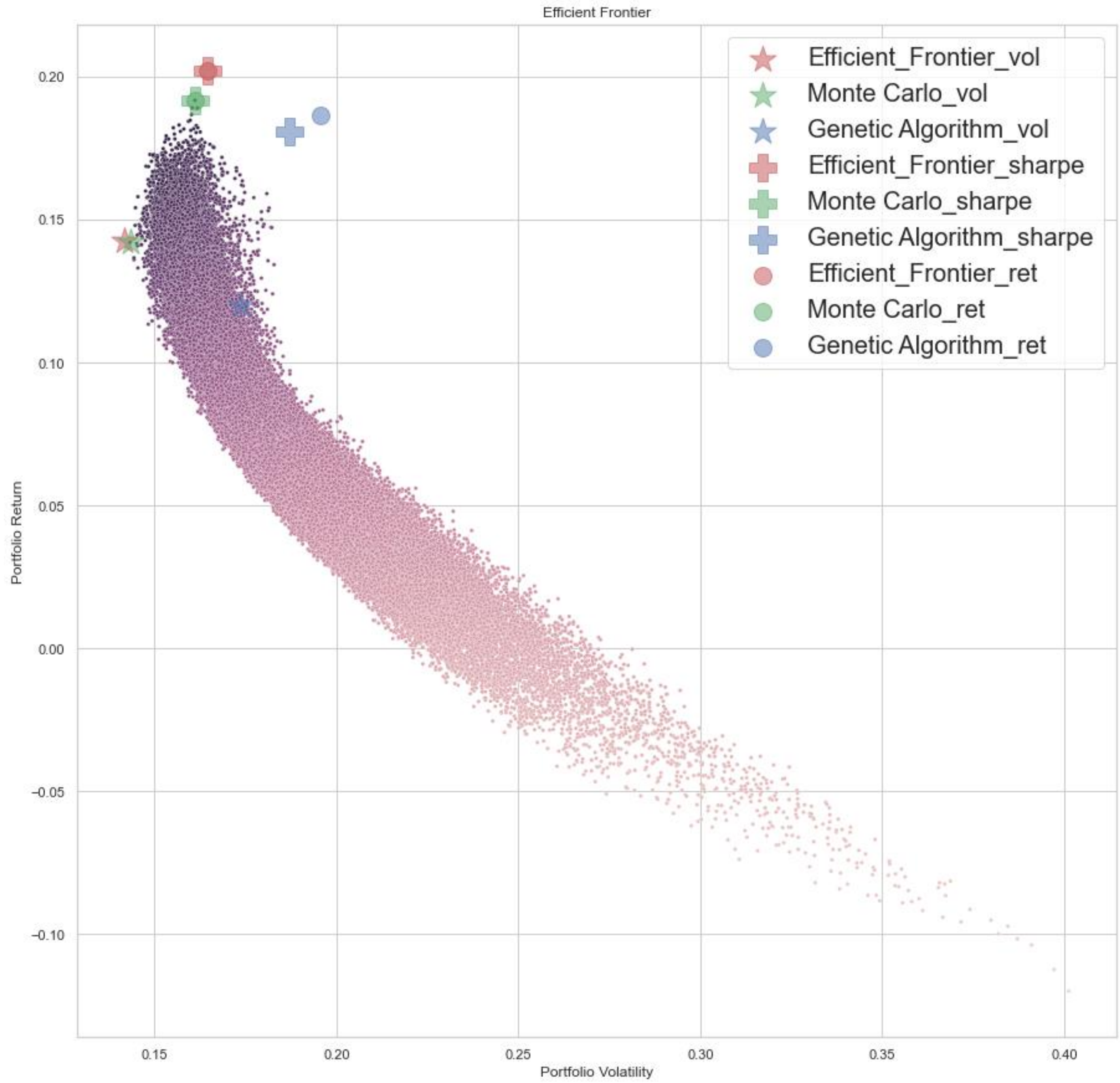
{‘similar’:[‘SPY’, ‘XLF’,‘QQQ’], ‘dissimilar’:[‘XLE’, ‘GDXJ’]}

II. Exploratory Data Analysis

ticker=['SPY','XLF','QQQ','XLE',GDXJ]

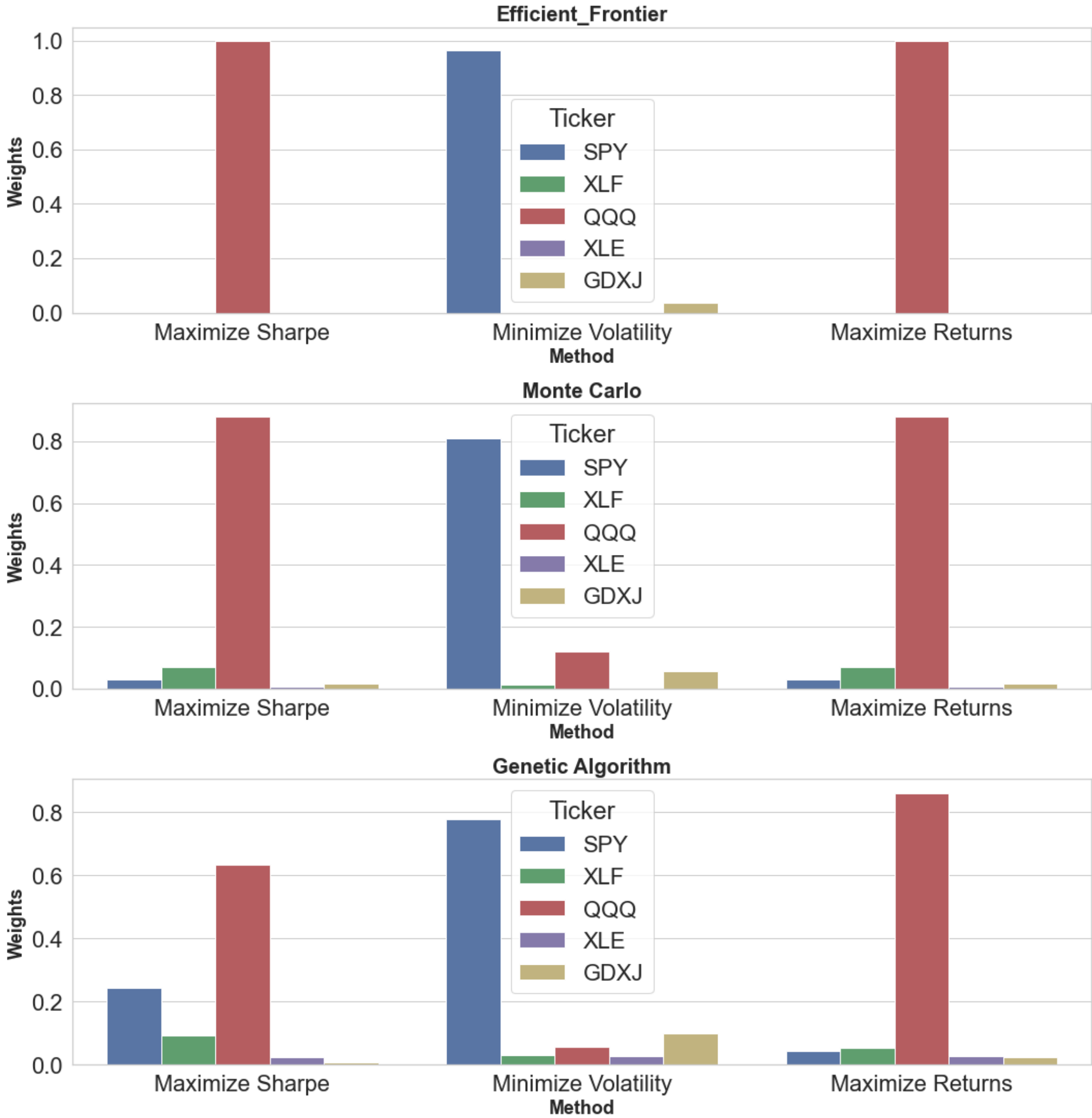


III. Benchmarking [PyPortfolioOpt vs Monte Carlo vs Genetic Algorithm]



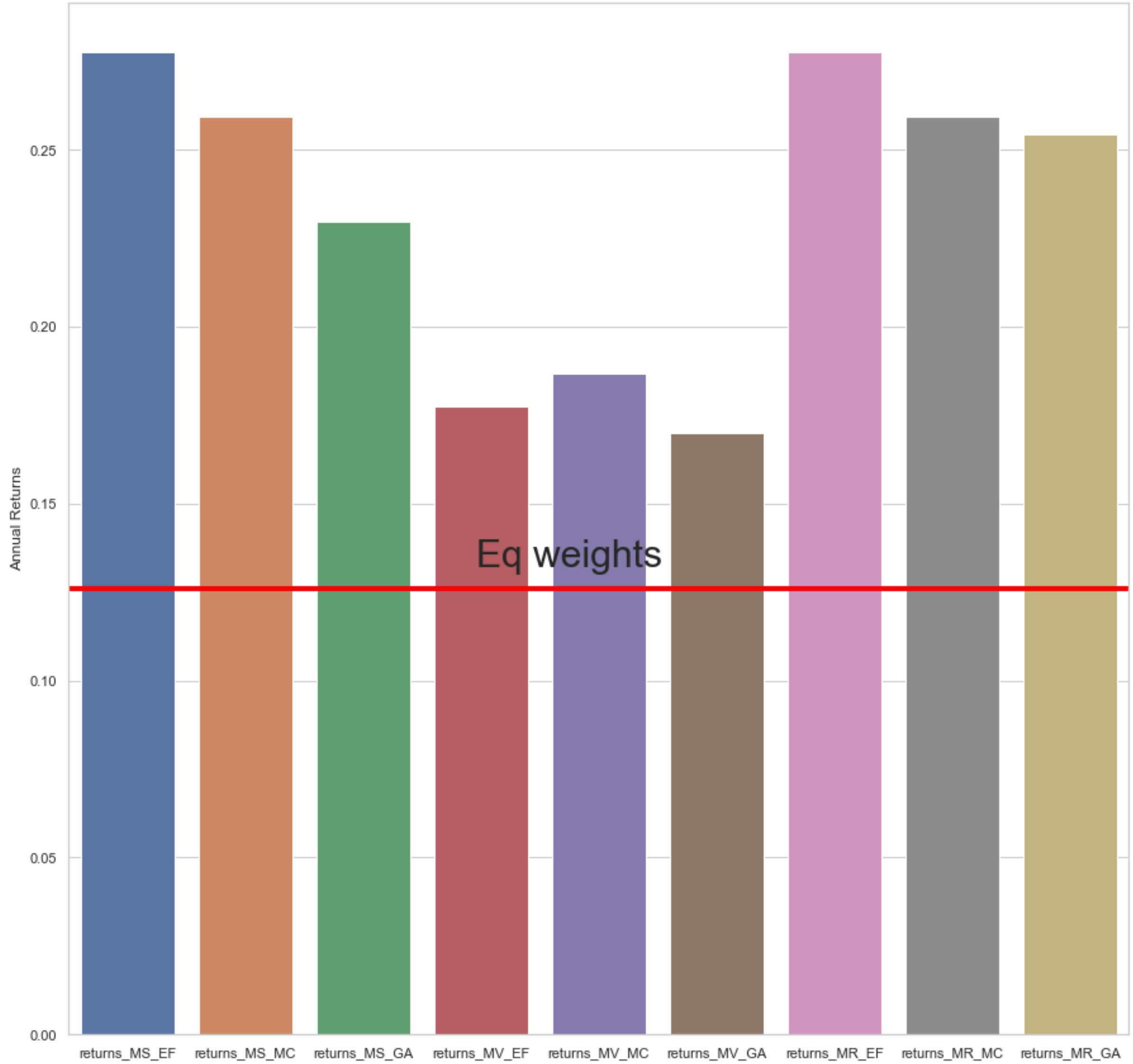
Simulation time: PyPortfolioOpt << Monte Carlo <<< Genetic Algorithm

III. Benchmarking [PyPortfolioOpt vs Monte Carlo vs Genetic Algorithm]



Similar portfolio composition and weights

III. Benchmarking [PyPortfolioOpt vs Monte Carlo vs Genetic Algorithm]



All portfolios better than an equally weighted portfolio!

IV. Grid Search, Cross-Validation and Hyper Parameter Tuning

Hyper Parameter(s):

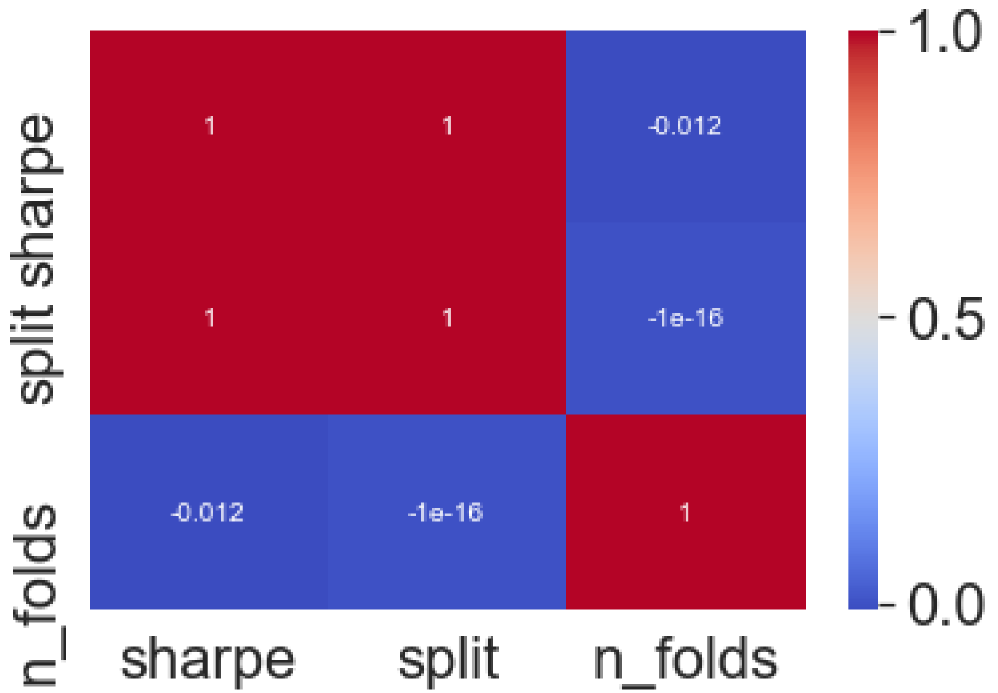
- n_splits→ [TimeSeriesSplit]
- spl→[train/test split]
- Perform ParameterGrid search to identify n_splits and spl that will maximize the training set sharpe ratio

```
from sklearn.model_selection import ParameterGrid
param_grid = {'n_splits': np.arange(2, 25, 1, dtype=int), 'spl' : [0.7,0.8]}
resul = []
grid = ParameterGrid(param_grid)

for params in grid:
    resul.append([params,obj_GA(params['n_splits'], params['spl'])])
```

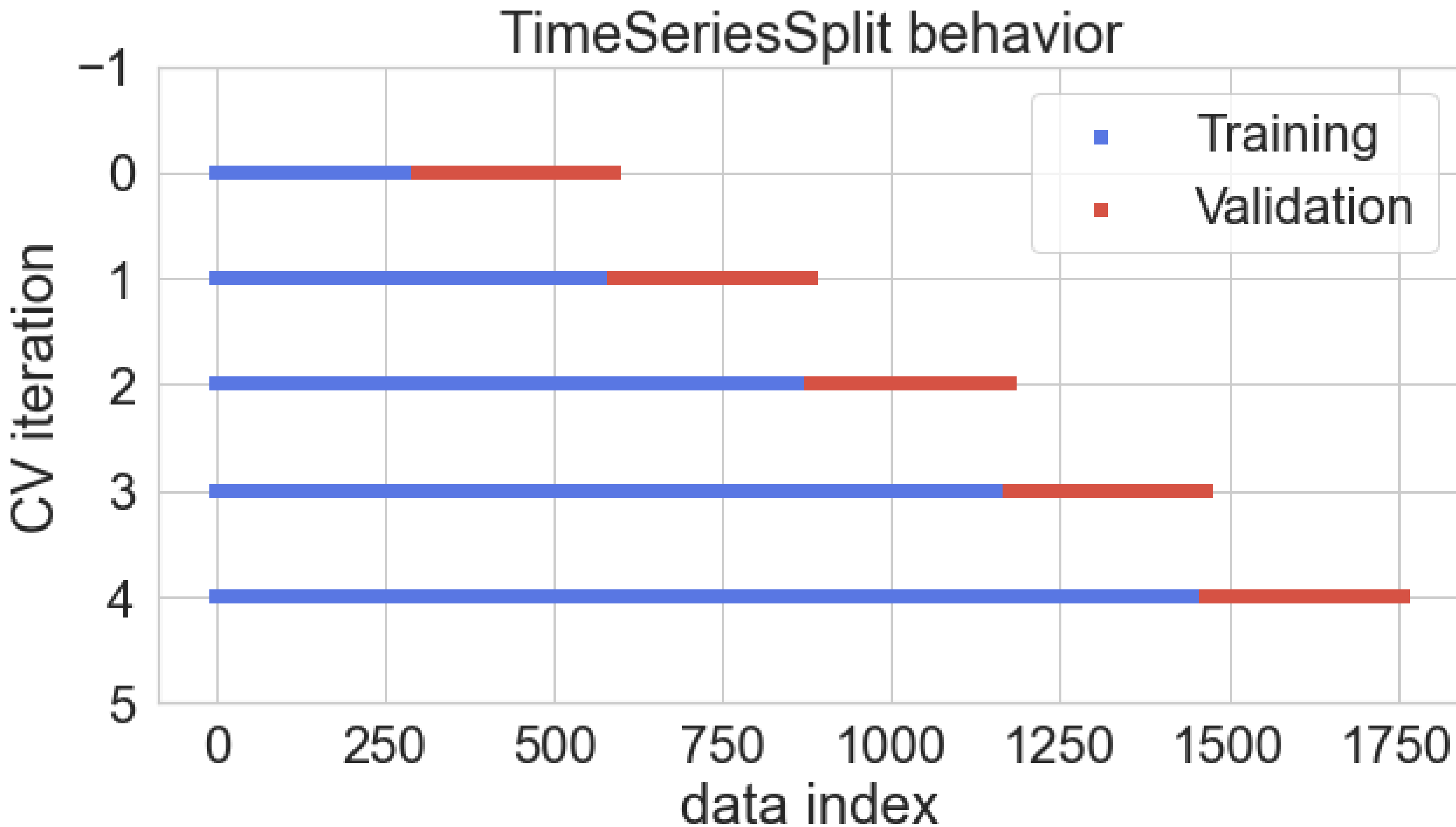
IV. Grid Search, Cross-Validation and Hyper Parameter Tuning

0	{'n_splits': 2, 'spl': 0.7}	-1.221261
1	{'n_splits': 2, 'spl': 0.8}	-1.027832
2	{'n_splits': 3, 'spl': 0.7}	-1.202281
3	{'n_splits': 3, 'spl': 0.8}	-1.037451
4	{'n_splits': 4, 'spl': 0.7}	-1.212207
5	{'n_splits': 4, 'spl': 0.8}	-1.034402
6	{'n_splits': 5, 'spl': 0.7}	-1.225038
7	{'n_splits': 5, 'spl': 0.8}	-1.037191
8	{'n_splits': 6, 'spl': 0.7}	-1.219758
9	{'n_splits': 6, 'spl': 0.8}	-1.040970
10	{'n_splits': 7, 'spl': 0.7}	-1.218063
11	{'n_splits': 7, 'spl': 0.8}	-1.040620
12	{'n_splits': 8, 'spl': 0.7}	-1.219144
13	{'n_splits': 8, 'spl': 0.8}	-1.038371
14	{'n_splits': 9, 'spl': 0.7}	-1.215605
15	{'n_splits': 9, 'spl': 0.8}	-1.039671



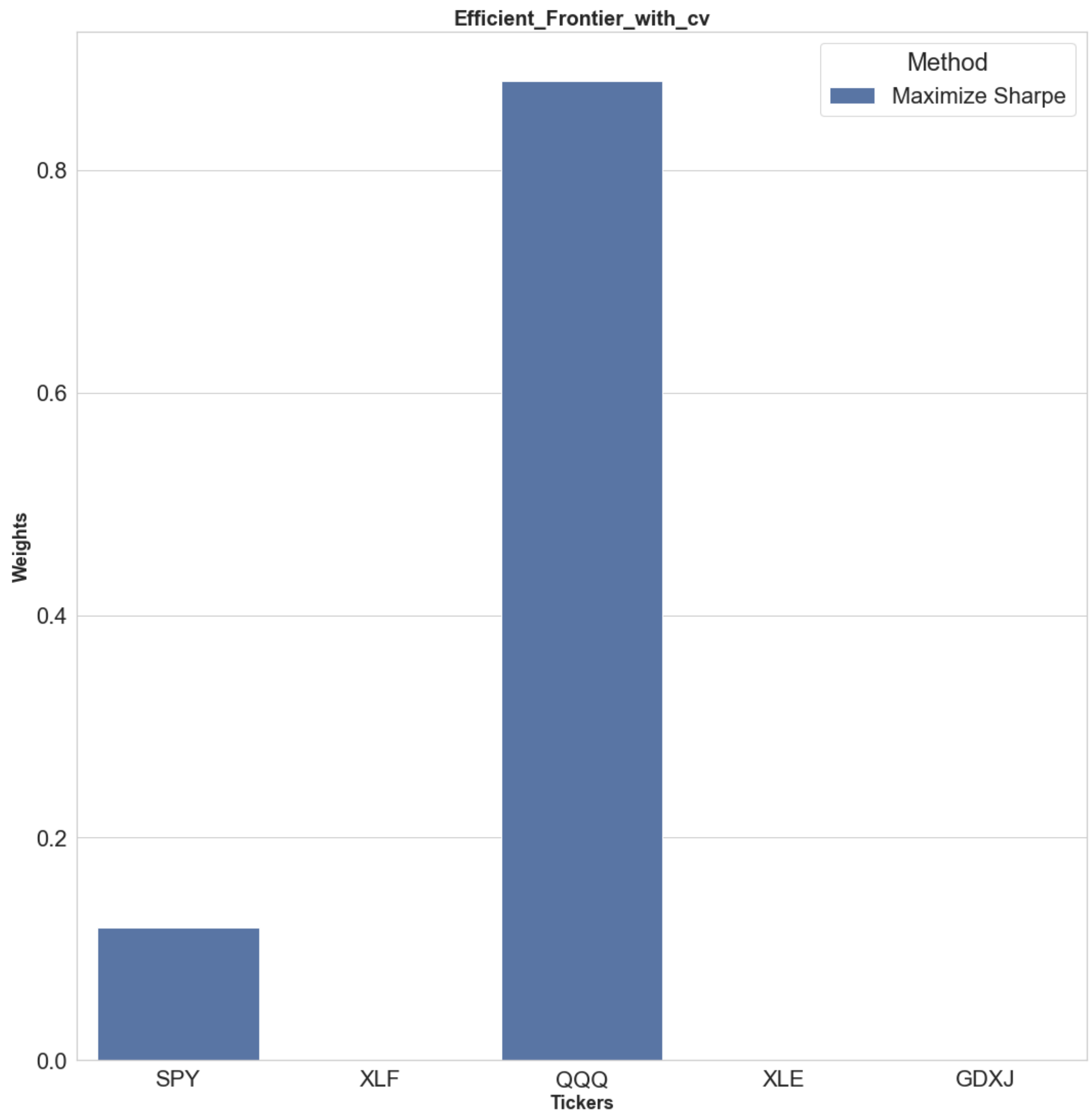
Identifying the best Hyper-Parameters

IV. Grid Search, Cross-Validation and Hyper Parameter Tuning



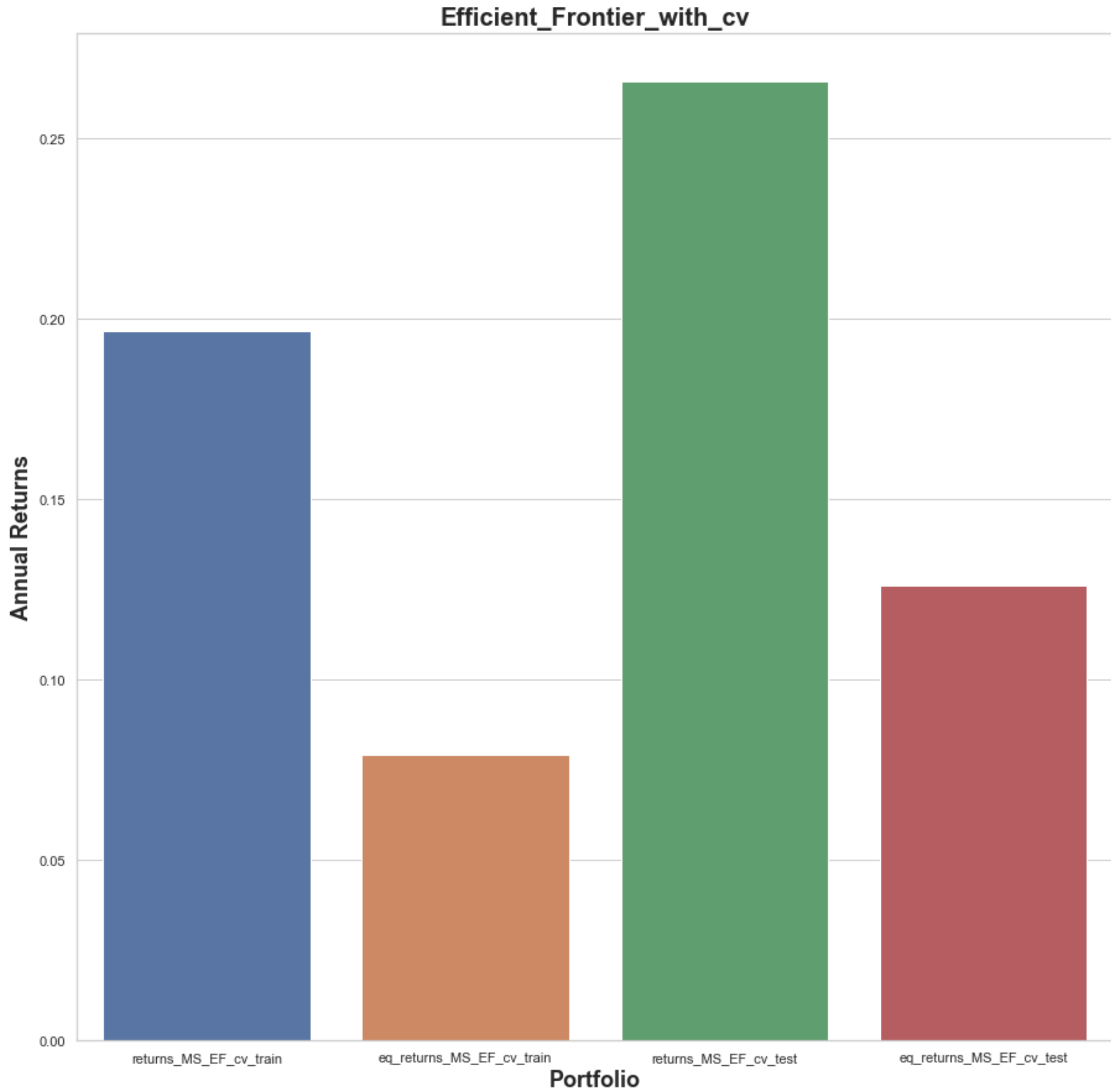
Cross-Validation using a simple Time Series Split

V. Optimal Portfolio ['SPY','XLF','QQQ','XLE',GDXJ]



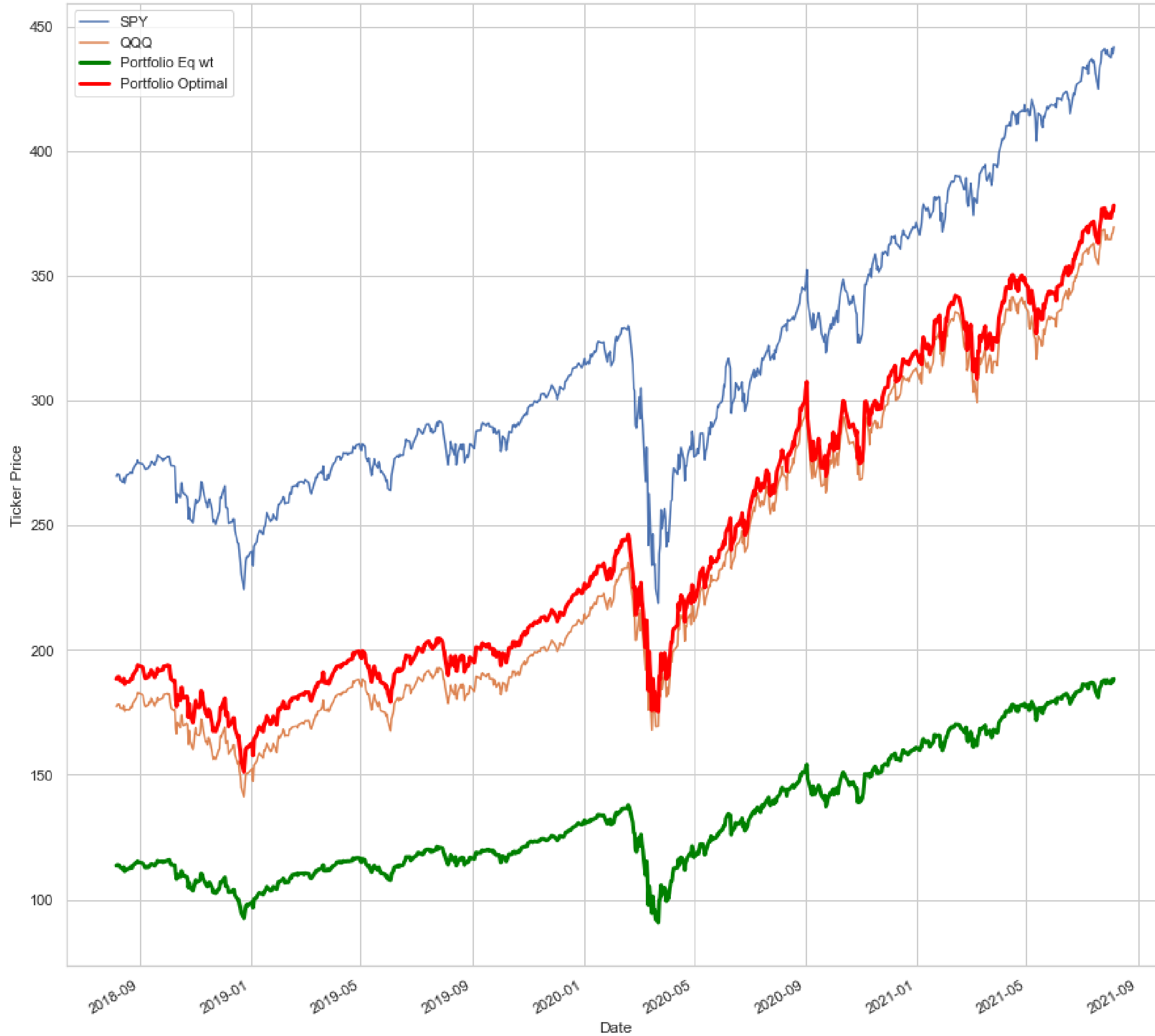
Optimization results in zero weights to underperforming assets (similar to Lasso)

V. Optimal Portfolio ['SPY','XLF','QQQ','XLE',GDXJ]: Performance



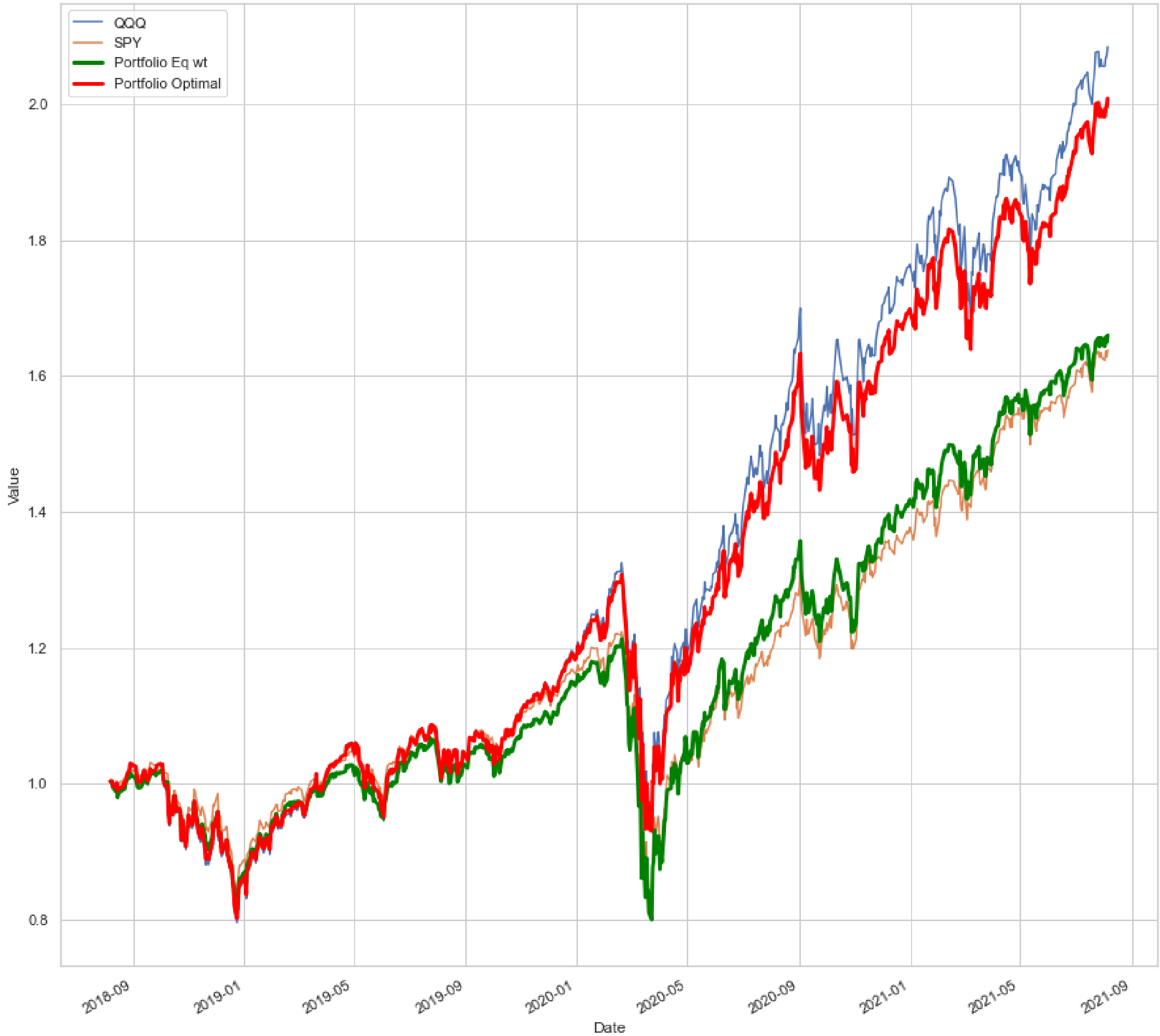
Weighted portfolio outperforms an equally weighted portfolio

V. Optimal Portfolio ['SPY','XLF','QQQ','XLE',GDXJ]: Performance



Visualizing the performance of the portfolios

V. Optimal Portfolio ['SPY','XLF','QQQ','XLE',GDXJ]: Performance

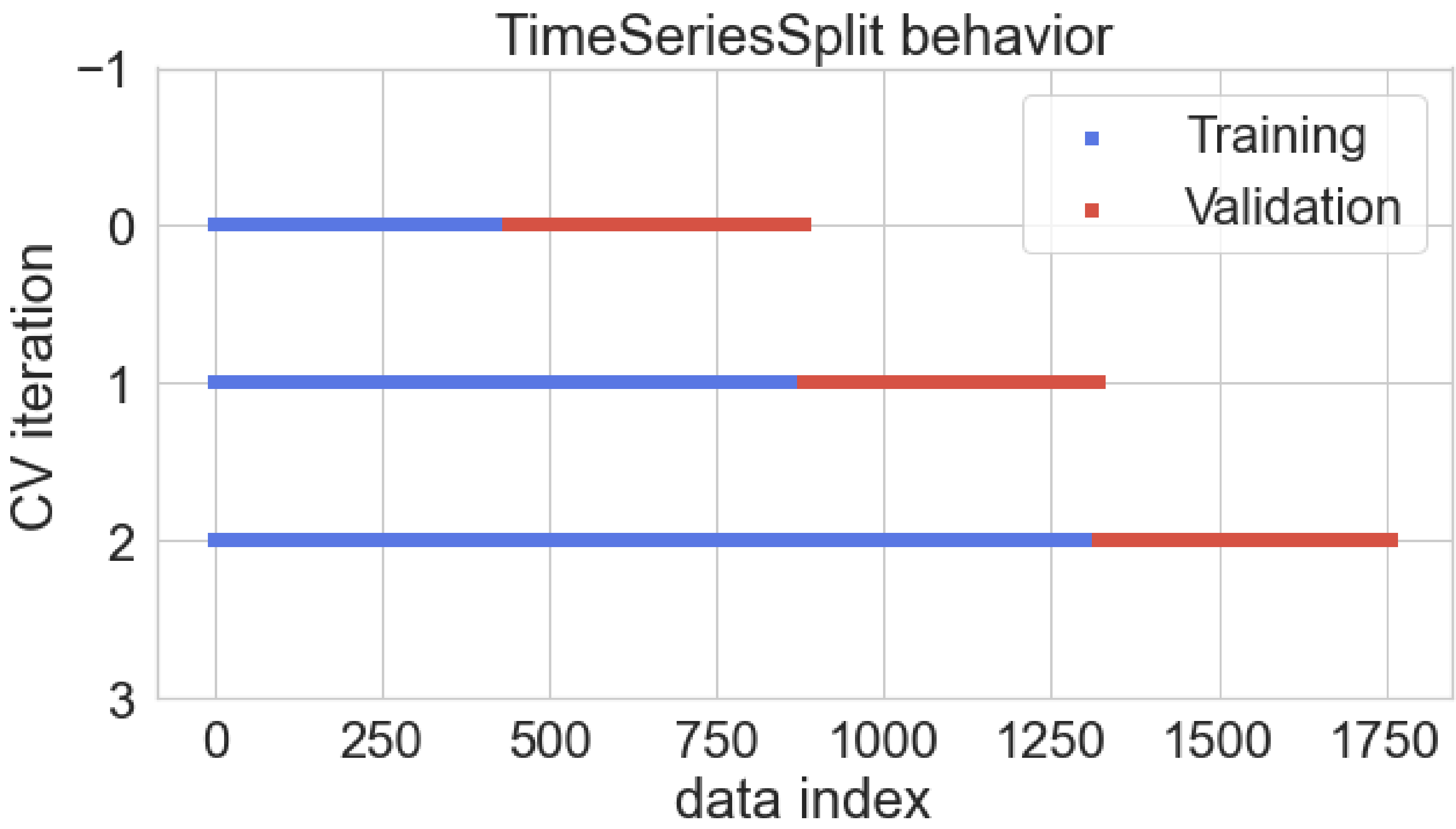


Visualizing the performance of the portfolios

VI. Optimal Portfolio

ticker=['QQQ','ARKK','XLK','XLV','XBI','IBB', 'IYR','XLRE','VNQ', 'XLF','KRE','KBE', 'XLE','XOP','ICLN']

	0	1
2	{'n_splits': 3, 'spl': 0.7}	-1.352684

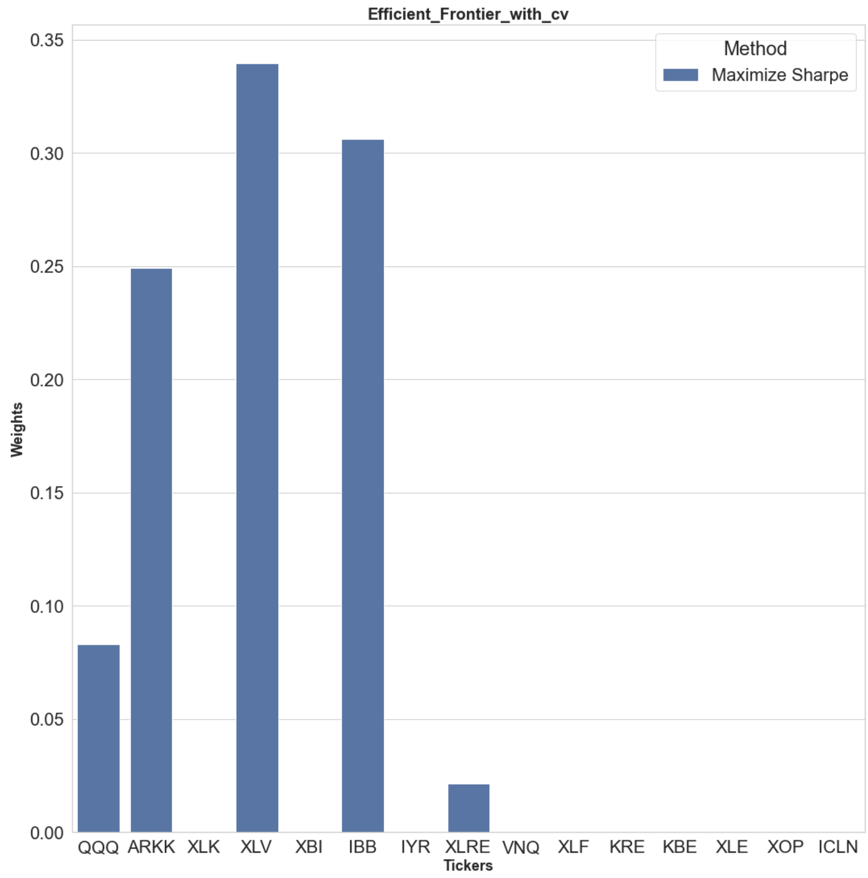


Cross-Validation using a simple Time Series Split

VI. Optimal Portfolio: Composition

ticker=['QQQ','ARKK','XLK','XLV','XBI','IBB', 'IYR','XLRE','VNQ', 'XLF','KRE','KBE', 'XLE','XOP','ICLN']

	0	1
2	{'n_splits': 3, 'spl': 0.7}	-1.352684

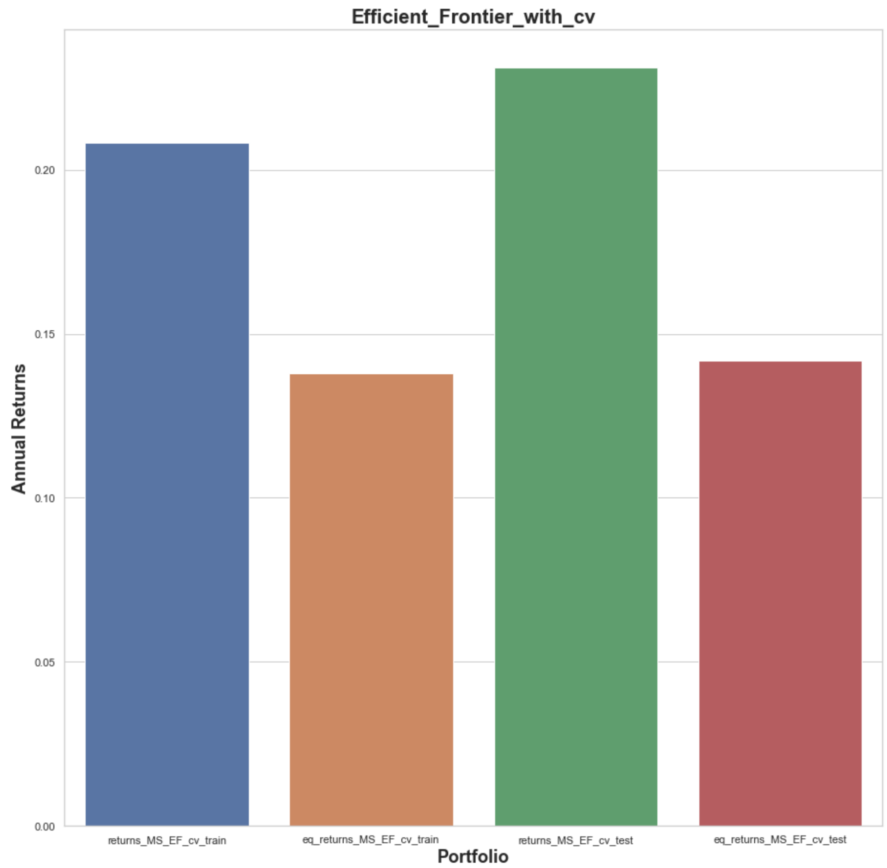


Optimization results in zero weights to underperforming assets (similar to Lasso)

VI. Optimal Portfolio: Performance

ticker=['QQQ','ARKK','XLK','XLV','XBI','IBB', 'IYR','XLRE','VNQ',

'XLF','KRE','KBE', 'XLE','XOP','ICLN']

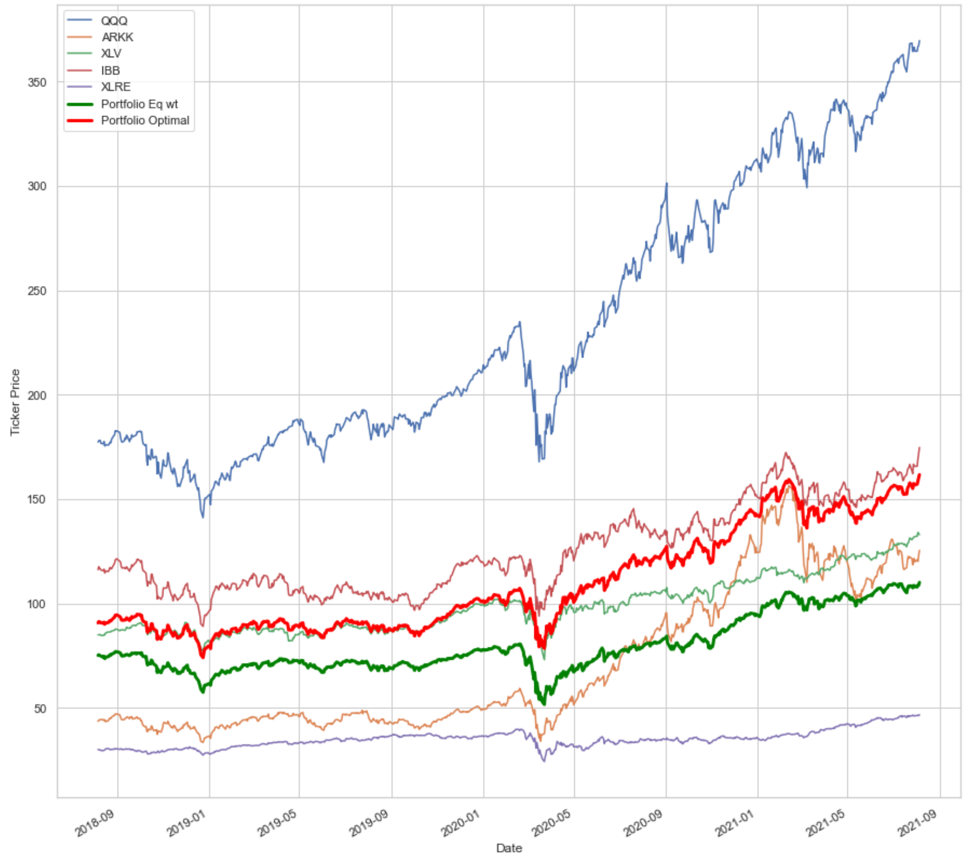


Weighted portfolio outperforms an equally weighted portfolio

VI. Optimal Portfolio: Performance

ticker=['QQQ','ARKK','XLK','XLV','XBI','IBB', 'IYR','XLRE','VNQ',

'XLF','KRE','KBE', 'XLE','XOP','ICLN']

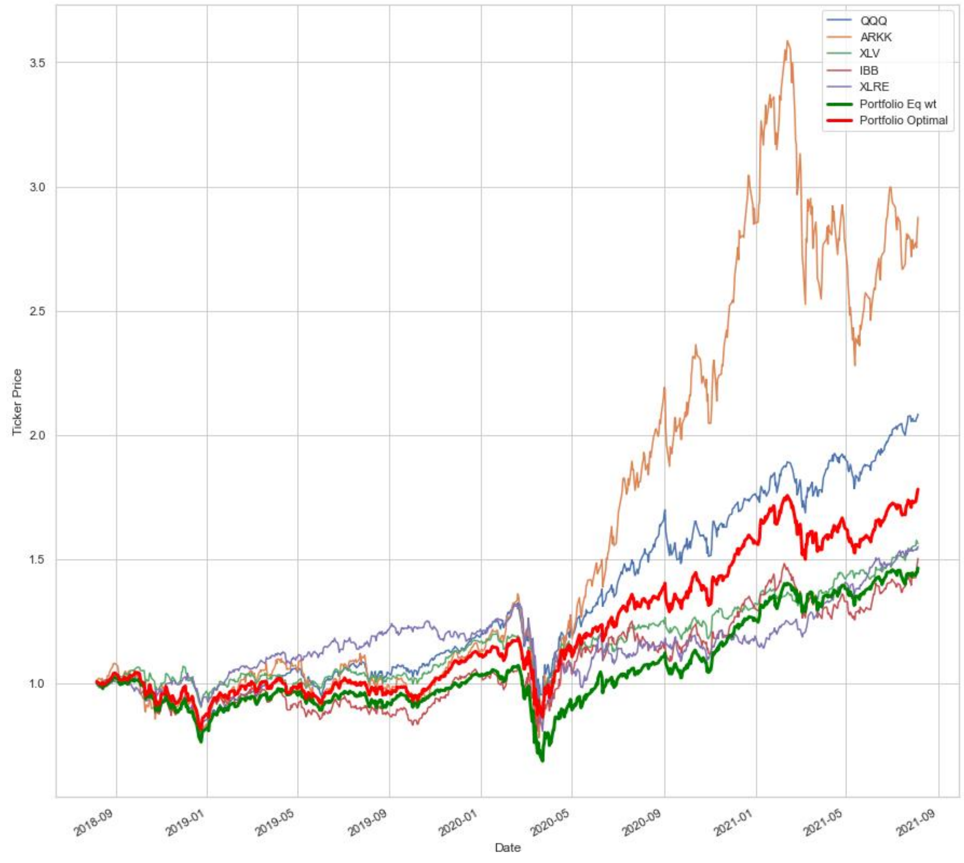


Visualizing the performance of the portfolios

VI. Optimal Portfolio: Performance

ticker=['QQQ','ARKK','XLK','XLV','XBI','IBB', 'IYR','XLRE','VNQ',

'XLF','KRE','KBE', 'XLE','XOP','ICLN']



Visualizing the performance of the portfolios

VII. Conclusions

- All techniques (PyPortfolioOpt vs Monte Carlo vs Genetic Algorithm) converge towards optimal portfolios on the Efficient Frontier plot.
- However, Genetic Algorithm was time expensive compared to Monte Carlo and PyPortfolioOpt techniques.
- Grid Search, Cross-Validation and Hyper Parameter Tuning helped unravel the best hyperparameters that maximize the train sharpe ratio.
- Similar to feature selection in Lasso-Regression, resulting portfolios from PyPortfolioOpt assign zero weights to underperforming assets.
- Weighted portfolio outperformed an equally weighted portfolio in the cases examined.