Statistical and Machine Learning Approaches for Portfolio Optimization: A study of NASDAQ stocks

Bautista Penayo
Bachelor of Business Mathematics and Economics
Zagreb School of Economics and Management
Zagreb, Croatia
bpenayo@student.zsem.hr

***Abstract*—In this paper, we compare six financial asset allocation strategies, plus one benchmark. We consider methods relying both in Machine Learning (ML) and common statistical procedures; and we run an out-of-sample back-test for each strategy. Through performance metric calculations, we determine that ML methods significantly improve returns, but do not always improve on the Sharpe Ratio.**

The study is carried out on 110 stocks composing the NASDAQ-100 and NASDAQ-Financial-100 indices.

***Index Terms*—Portfolio Optimization, Machine Learning, Time Series, Modern Portfolio Theory**

I. Introduction

This paper undertakes a comparative study of six asset allocation strategies, encompassing both traditional statistical methodologies and modern machine learning approaches, to assess their effectiveness in portfolio optimization.

The dataset utilized in this study comprises daily percentage returns of 110 stocks listed on the NASDAQ stock exchange, spanning from the onset of the year 2000 to April 4th, 2020. These stocks primarily represent constituents of the NASDAQ-100 and NASDAQ Financial-100 indices, offering a diverse spectrum of assets across various sectors of the economy. Daily returns are derived from the Adjusted Close price data of each stock, expressed as the daily fractional change in this price.

To establish a benchmark, we construct an equally weighted portfolio comprising all 110 stocks, meticulously back-tested out-of-sample to mitigate the risk of "look-ahead bias" or data leakage. Subsequently, six asset allocation strategies are examined, each subjected to the same rigorous out-of-sample back-testing using also the same dataset.

The Mean-Variance approach, a cornerstone of Modern Portfolio Theory, serves as the foundation for four of the strategies analyzed [1]. This includes two strategies employing statistical procedures to estimate expected returns and portfolio volatility: Minimum Variance Portfolio and Maximum Sharpe Ratio. Additionally, we explore the Mean-Variance framework augmented with machine learning techniques, utilizing Facebook Prophet for return forecasting and a GARCH model for volatility prediction.

In parallel, we investigate two alternative methodologies: Risk Parity and Hierarchical Risk Parity (HRP). Risk Parity optimizes portfolio weights by equalizing the contribution of each asset to total portfolio risk, while HRP leverages a Hierarchical Clustering algorithm to construct a correlational structure of assets, allocating weights as in Risk Parity but within and across clusters [2].

Our analysis reveals significant insights into the performance of these strategies. Notably, machine learning-based variations demonstrate a remarkable enhancement in portfolio returns, compared to their naive counterparts. However, the impact on the Sharpe Ratio, a key measure of risk-adjusted returns, varies across the different methodologies, showing that some strategies often increased risks in order to achieve such higher returns.

II. Methodology

A. Dataset Used

The dataset utilized in this study comprises daily returns of 110 selected stocks from the NASDAQ exchange spanning the period from 2000 to 2020.

Initially, more than 5000 stocks from the NASDAQ were considered. To filter the stocks, web scraping techniques were employed to gather information on stocks comprising the NASDAQ-100 and NASDAQ Financial-100 indices. Subsequently, CSV files corresponding to the selected stocks were imported into Python using the Pandas library. Through a combination of loops and built-in Python functions, the CSV files were concatenated into a single DataFrame, retaining only the necessary data fields - Dates and Adjusted Close prices. Afterwards, we dropped the rows of all the trading days before the start of the year 2000, thus using only data from 2000 till April 2020. The data-frame was then pivoted by the stocks, such that each stock comprised one column, and the daily returns of each stock were calculated using the pct_change() function in Pandas. Up to this point, the data-frame consisted of 179 columns, indexed by Date.

Exploratory Data Analysis (EDA) was conducted to understand the structure and characteristics of the dataset. Descriptive statistics revealed significant missing data in several stocks. Consequently, stocks with missing data were excluded from the analysis, resulting in a final dataset comprising of 110 stocks.

B. Overview of Strategies

- Two strategies through Naive Mean-Variance: estimated expected returns of the portfolio using arithmetic average of historical asset returns, and estimated portfolio volatility by calculating the covariance matrix of asset returns from past data. Based on these parameters, the Minimum Variance Portfolio, and the Maximum Sharpe Ratio Portfolio were found at every iteration.
- Two strategies through Mean-Variance with Machine Learning: similar to the Naive case, but through estimation of expected returns through Facebook Prophet, and utilizing a Generalized Auto Regressive Conditional Heteroskedasticity (GARCH) model to forecast volatility.
- Risk Parity Approach: using the inverse volatility method, optimized weights such that each asset contributes equally to total volatility.
- Hierarchical Risk Parity (HRP): created correlational clusters of assets based on Euclidean distance measures of asset returns, which account for similarity measures. Then, optimized using the Risk Parity Approach, but within and across clusters.

C. Explanation of the Models

1) Naive Mean-Variance Optimization: Mean-Variance optimization was employed to construct two distinct portfolios: the Minimum Variance Portfolio (MVP) and the Maximum Sharpe Ratio Portfolio. The expected asset returns and covariance matrix of asset returns, crucial parameters for optimization, were estimated using historical data. Specifically, the expected returns were calculated as the mean of historical returns using the Pandas mean() function, while the covariance matrix was computed using the Pandas cov() function. Before deciding for these simple methods, the approaches of rolling average and rolling covariance were tested, with different windows, but it was found that the simple methods utilizing the whole available historical data at every iteration achieved a higher portfolio value at the end of the back-test (suggesting they were better predictors).

The Minimum Variance Portfolio (MVP) minimizes the portfolio risk (variance) for a given level of return [1][3]:

$$\min_w \sqrt{w^T \Sigma w}$$

subject to

$$w^T \mu = r$$

$$\sum_{i=1}^{n} w_i = 1$$

$$0 \leq w_i \leq 1 \ \ \forall i$$

Where:

- w is the vector of portfolio weights w_i for asset i,
- Σ is the covariance matrix of asset returns,
- μ is the vector of expected returns,
- r is the target level of return, and
- n is the number of assets.

Since $Portfolio\ Variance = w^T \Sigma w$, the objective function optimizes through the standard deviation of the portfolio, which yields equivalent optimal solutions.

For our implementation, we did not specify a target level of return, but rather attempted to find the actual minimum risk portfolio at every iteration of the back-test. Also, the constraint where weights of assets should be between 0 and 1 imposes the "no-short sale" restriction, which has been demonstrated to improve mean-variance performance [4].

The Maximum Sharpe Ratio (MSR) strategy aims to construct a portfolio that maximizes the Sharpe Ratio, a measure of risk-adjusted returns. The Sharpe Ratio is defined as the ratio of excess return to volatility, where excess return is the difference between the portfolio return and the risk-free rate [5]. Mathematically, the objective function of the MSR strategy can be expressed as:

$$\max_w \frac{w^T \mu - r_f}{\sqrt{w^T \Sigma w}}$$

subject to the following constraints:

$$\sum_{i=1}^{n} w_i = 1$$

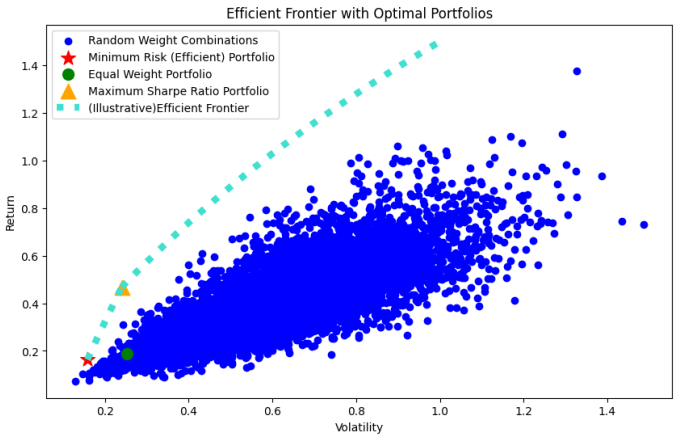
$$0 \leq w_i \leq 1 \ \ \forall i$$

Where:

- n is the number of assets,
- w is the vector of portfolio weights w_i for asset i,
- μ is the vector of expected returns,
- Σ is the covariance matrix of asset returns,
- r_f is the risk-free rate,
- $w^T \mu - r_f$ represents the excess return, and
- $\sqrt{w^T \Sigma w}$ is the portfolio volatility.

In this paper's implementation, a constant risk-free rate of zero was assumed. The reason for this is that such a rate has varied across the considered time period. Thus, for the sake of simplicity, the calculations of the Sharpe Ratio were done without this rate. However, the comparative purpose of this study is still achieved with this simplification: although the Sharpe Ratios will naturally be overstated in certain periods, they will experience this consistently for each strategy.

To illustrate what both strategies do, both in the Naive case and in the machine-learning variation, an efficient frontier plot was visualized:



Using a subset of data, we found the explained portfolios and plotted them along the (illustrative) efficient combinations of volatility (standard deviation of returns) and returns that form "the efficient frontier". We do this to understand the risk-return trade-offs, as well as visualizing what the strategies are doing at every weight re-balancing period when we perform the back-tests. The Naive Mean Variance and the Mean Variance with Machine Learning frameworks will attempt to find these two portfolios using both basic statistics and ML methods.

2) Facebook Prophet and GARCH Time Series Models: Incorporating machine learning techniques, the Mean-Variance framework was augmented to forecast expected returns and portfolio volatility.

For return forecasting, the Facebook Prophet time series additive model was utilized [6]. Hyperparameter tuning was performed using a subset of historical data, and optimal hyperparameters were determined.

Volatility forecasting was accomplished using the GARCH model, with the p (number of lagged squared residuals, ϵ_{t-i}^2) and q (number of lagged conditional variances, σ_{t-j}^2) parameters determined manually based on auto-correlation and partial auto-correlation plots. Additionally, a rolling correlation matrix with a 252-day window was calculated using the Pandas library. With these forecasts we can derive a future covariance matrix that is an input to our optimization models.

Returns Forecasting

Time series additive models, such as the Facebook Prophet model, decompose a time series into multiple components, including trend, seasonality, holidays, and noise. Mathematically, the additive model can be represented as [7]:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

Where:

- $y(t)$ represents the observed value of the time series at time t,
- $g(t)$ is the trend component, capturing the long-term direction of the time series,
- $s(t)$ is the seasonal component, representing periodic fluctuations in the data,
- $h(t)$ is the holiday component, accounting for special events or holidays that impact the time series,
- ϵ_t is the error term, representing random noise or residuals.

In the Facebook Prophet model, the trend component $g(t)$ is modeled using a piece-wise linear time-dependent function with constant growth rate, or a logistic time-dependent saturating growth model. The seasonal component $s(t)$ is modeled using standard Fourier series expansions, but without the intercept term because simultaneously the trend component is being modeled. The holiday component $h(t)$ is represented as indicator variables for each holiday or special event [7]. No holiday was specified in our model because we only have data of trading days, thus, there is no data of holidays.

Once the parameters of the additive model are estimated, forecasts of future values of the time series can be generated by extrapolating the trend, seasonality, and holiday components forward in time.

Portfolio Volatility Forecasting

GARCH models are commonly used in time series analysis to model the volatility clustering phenomenon, where periods of high volatility tend to be followed by periods of high volatility, and vice versa [8]. Mathematically, a GARCH(p, q) model can be represented as [9][10]:

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2$$

Where:

- σ_t^2 represents the conditional variance of the time series at time t,
- ω is the constant term or intercept,
- α_i are the parameters of the autoregressive component, capturing the impact of past squared residuals on current volatility,
- ϵ_{t-i} are the squared residuals or errors at time $t-i$,
- β_j are the parameters of the moving average component, capturing the impact of past conditional variances on current volatility, and
- σ_{t-j}^2 represents the conditional variance at time $t-j$.

The GARCH(p, q) model extends the traditional autoregressive (AR) model by incorporating the conditional variance of the time series as a predictor. It allows for the modeling of time-varying volatility (standard deviation of returns).

Let Σ represent the covariance matrix, Corr represent the correlation matrix, and σ represent the vector of standard deviations of returns.

The formula for obtaining the **future** covariance matrix from the rolling correlation matrix and the forecasted standard deviations of returns is as follows:

$$\Sigma = \text{diag}(\sigma) \times \text{Corr} \times \text{diag}(\sigma)$$

Where:

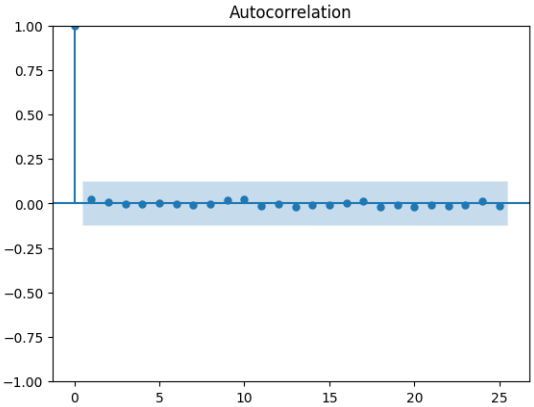
- $\text{diag}(\sigma)$ is a diagonal matrix with elements representing the standard deviations of returns,
- Corr is the correlation matrix of returns, and
- Σ is the resulting covariance matrix.

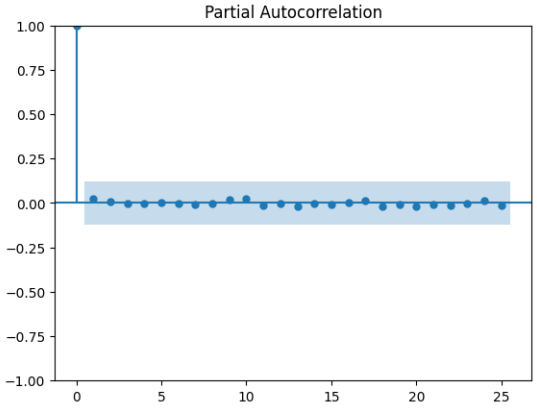
In this formula, the matrix $\text{diag}(\sigma)$ represents a diagonal matrix with the **forecasted** standard deviations of returns as its diagonal elements (forecasted using GARCH). Multiplying this diagonal matrix by the **rolling** correlation matrix Corr and then by another diagonal matrix of forecasted standard deviations effectively scales each pairwise correlation by the corresponding standard deviations, resulting in the **predicted** covariance matrix.

3) Manual Configuration of the GARCH Model: [9][10][11] To define the optimal number of lagged variances and squared residuals for our data, the Auto-correlation and the Partial Auto-correlation functions were used. The Auto-correlation of our data defines the number of lagged-variances (p parameter) to include in our model as part of the moving average component of the GARCH. The Partial Auto-correlation defines the number of lagged squared residuals (q parameter) as part of the auto-regressive component.

If $q = p = 0$, then the GARCH model converges to white noise (time series with a standard normal distribution) [9][12]. Because this would imply a constant variance over time and a mean of zero, and because we know from research the volatility clustering phenomena [8], we posit that white noise would not be appropriate to model volatility. Furthermore, if only $p = 0$, then the GARCH model converges to an ARCH model, where we would be excluding the component that captures the impact of past variances. Again, because of the volatility clustering finding, this alternative is a priori excluded.

However, when we plotted the Auto-correlation and Partial Auto-correlation functions of the variance of returns of our NASDAQ data (we did the same for returns and the conclusions were the same), we obtained the following:





Both plots suggest that serial correlation and partial serial correlation are not significantly different from zero (the blue rectangle represents the 95% confidence interval). Because of this, we would expect to not include any lags. However, due to what was explained before, we configure the GARCH parameters to be $p = q = 1$.

4) Risk Parity Model: The Risk Parity methodology sought to optimize portfolio weights by equalizing the contribution of each asset to total portfolio risk. Each assets' risk was calculated through a rolling standard deviation with several windows, and through a static standard deviation of returns of the whole available dataset, at every iteration. By comparing the performance of risk parity under these different calculations, we determined that the best way to optimize weights is by using the latter, the simple standard deviation of all asset returns at every re-balancing period.

The Risk Parity model aims to allocate portfolio weights such that each asset contributes equally to the total portfolio risk. Mathematically, the weight of each asset i is determined based on the inverse of its volatility:

$$w_i = \frac{\frac{1}{\sigma_i}}{\sum_{j=1}^{n} \sigma_j}$$

Where:

- w_i represents the weight of asset i in the portfolio,
- σ_i is the standard deviation (volatility) of asset i, and
- n is the number of assets.

The Risk Parity model promotes diversification by assigning higher weights to assets with lower volatility and vice versa. By equalizing the risk contribution of each asset, the portfolio aims to achieve a balanced risk profile.

To ensure that the weights sum up to 1, a normalization step is performed:

$$\sum_{i=1}^{n} w_i = 1$$

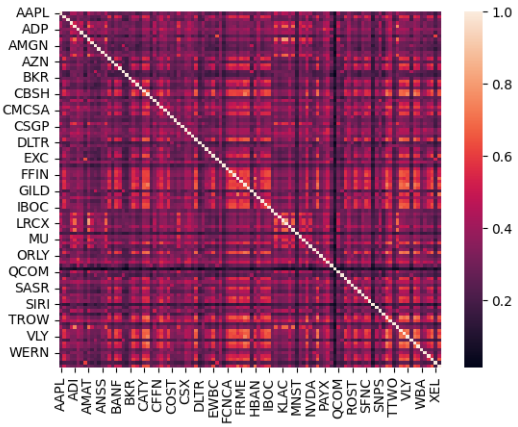
Where n is the total number of assets in the portfolio.

5) Hierarchical Risk Parity (HRP) Model: The HRP method leverages machine learning to balance the risk contribution, not only to total risk, but also within created correlational clusters of assets. It involves several key steps [13]:

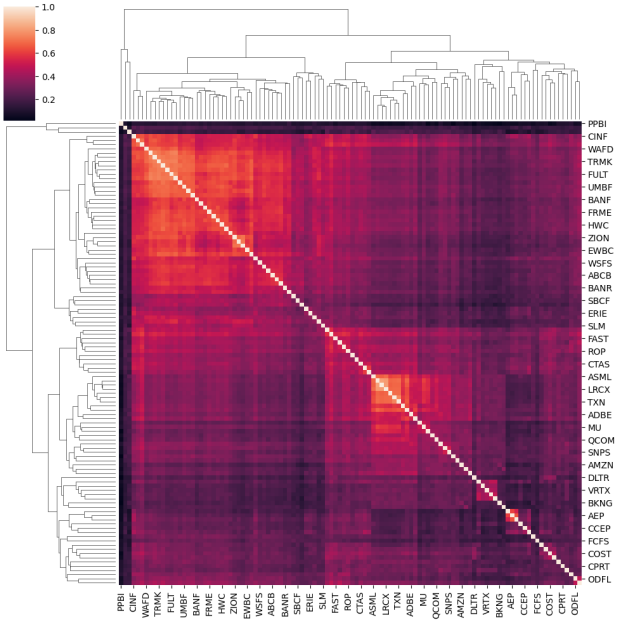
1. **Hierarchical Clustering:** HRP starts by clustering assets based on their pairwise correlations, forming a hierarchical tree (dendrogram) that represents the relationships between assets. This hierarchical structure is constructed using techniques such as agglomerative clustering, where assets are successively merged into clusters based on their similarity, calculated based on a Euclidean distance of two assets in the correlation matrix.

2. **Matrix Seriation:** Once the hierarchical tree is constructed, the HRP algorithm performs matrix seriation, which involves sorting the assets in the dendrogram to minimize the distance between leaf nodes (assets). This step ensures that assets with similar risk profiles are grouped together in the dendrogram.

Correlation Matrix of Stocks



Seriated Correlation Matrix and Dendrogram



3. **Cluster Weighting:** After seriation, the algorithm assigns weights to each cluster in the dendrogram based on risk parity principles. The weights are determined by recursively splitting the total weight of each parent cluster among its child clusters, ensuring that risk contributions are balanced at each level of the hierarchy.

4. **Asset Weighting:** Finally, the algorithm assigns weights to individual assets within each cluster based on their contribution to the total risk of the cluster. Assets with higher volatility or risk are assigned lower weights, while assets with lower volatility are assigned higher weights, ensuring that risk contributions are proportional to each asset's volatility.

D. Back-Testing

The set-up for all the investment strategies follows the same framework. The initial invested capital is 10000 units of currency, which invests initially in an equally-weighted portfolio. The re-balancing period is set to once a year (once every 252 days). This way, we are dealing with a passive investment strategy.

As explained before, the exceptions are the Mean Variance strategies with ML, where its models require data and where the back-testing algorithm is of high computational complexity since at every re-balancing period these models have to be run. Thus, for these two strategies, we established an annual re-balancing period, but the re-balancing starts in the 5th year (the strategy holds the equally-weighted portfolio and, though we eliminate this constraint from the 4th year onwards, the next re-balancing period in the algorithm becomes the 5th year). Note that, because of this situation, MVP ML and Max Sharpe ML are much more constrained that the rest of the strategies.

1) Transaction Costs: Since we are dealing with passive investment strategies, the effect of transaction costs on strategy performance is significantly minimized. Therefore, a simplification of the calculation of transaction costs would not affect our conclusions: we define transaction cost as 1% of the change in weights at each re-balancing period. We subtract the portfolio gross return calculated with the optimized new weights by this transaction cost. This makes sense only because the calculated asset returns are also fractional, since we derived them from the Adjusted Closed Prices. And both the weight changes, the transaction cost and the returns are thus, proportions or fractions based on the invested capital.

E. Back-test Algorithm Pseudo-code

Input: Returns_data, transaction_rate
Output: portfolio_values, adjusted_returns, portfolio_volatilities

Initialization :

num_assets ← number of assets in Returns
num_days ← number of days in Returns
rebalancing_period ← 252 {Number of trading days in a year}

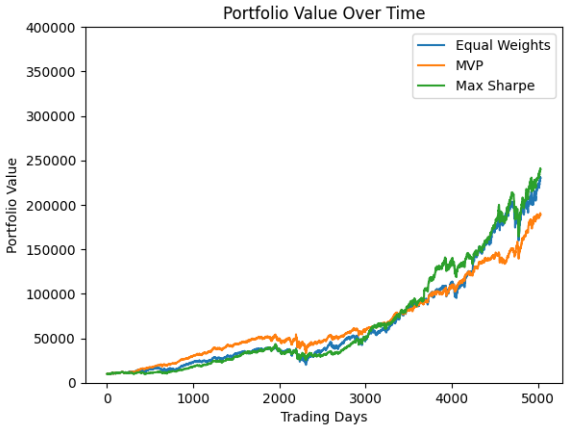
init_capital ← 10000 {Initial capital}
transaction_cost ← weight_difference × transaction_rate
init_portfolio_value ← init_capital - transaction_cost × init_capital
portfolio_values ← [init_portfolio_value]
adjusted_returns ← [ ] {Empty list}
portfolio_volatilities ← [ ] {Empty list}
previous_weights ← array of ones of length num_assets divided by num_assets

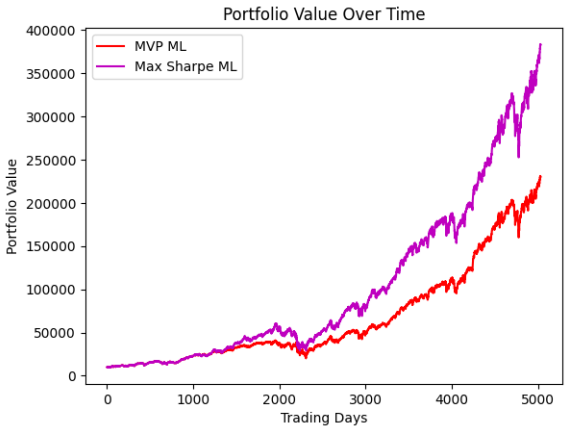
Loop over days:
for day ← 0 **to** num_days - 1 **do**

  if day $\bmod$ rebalancing_period $= 0$ **and** day > 0 **then**
    in_sample ← Returns up to day
    exp_returns ← mean of in_sample or forecast
    covar_matrix or standard_dev ← covariance matrix or std of in_sample_data; or predicted covariance matrix using std from GARCH and rolling correlation
    Optimize Weights ← Run a model we discussed
    weights ← variable with optimized weights
    portfolio_return ← returns with optimized weights
    adjusted_return ← adjust for transaction costs
    portfolio_value ← portfolio_values[-1] × (1 + adjusted_return)
    portfolio_values.append(portfolio_value)
    portfolio_volatility ← volatility with optimized weights
    portfolio_volatilities.append(portfolio_volatility_value)
    previous_weights ← weights
  else
    portfolio_return ← returns[day] with previous weights
    adjusted_return ← portfolio_return[day]
    portfolio_value ← portfolio_values[-1] × (1 + adjusted_return)
    portfolio_values.append(portfolio_value)
    covar_data ← like in_sample data
    covar_matrix or std ← covariance matrix of covar_data or std of covar_data
    portfolio_volatility_value ← portfolio volatility with previous weights
    portfolio_volatilities.append(portfolio_volatility_value)
  end if
end for
return portfolio_values, adjusted_returns, portfolio_volatilities

III. Results

A. Portfolio Returns





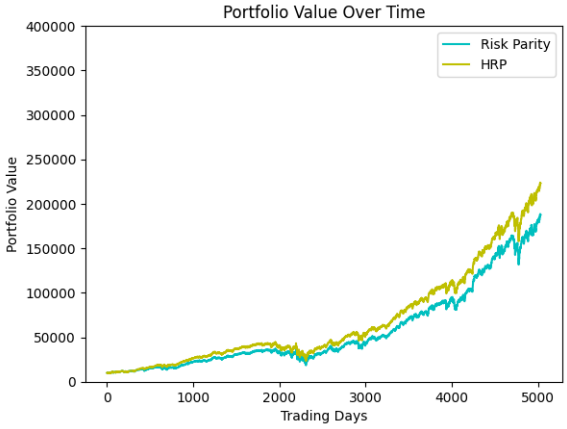


TABLE I
Portfolio Values by 2020

Strategy	Portfolio Value
Max Sharpe ML	381,492.04
MVP ML	229,702.81
Naive Max Sharpe	239,796.27
Benchmark	229,702.81
HRP	222,477.10
Naive MVP	188,672.58
Risk Parity	187,353.56

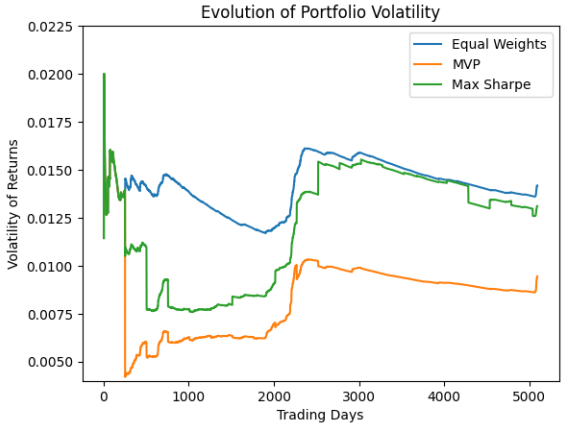
The evolution of portfolio values across the evaluated period shows how each strategy performed in terms of returns.

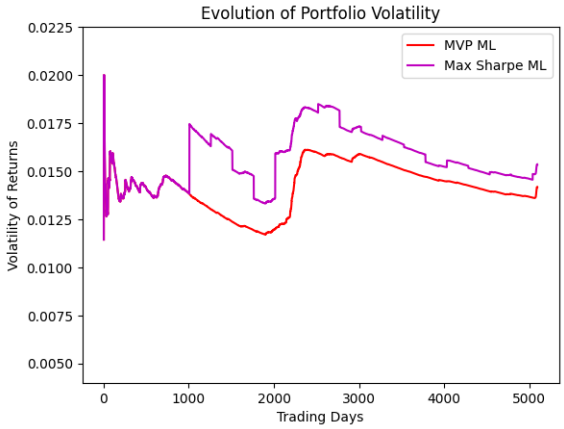
It is evident that Risk Parity did not offer convincing returns for this period. It is the worst performer among all analyzed strategies (return-wise). The second worst performer (return-wise), was the Minimum Variance Portfolio (MVP) from the naive Mean-Variance approach.

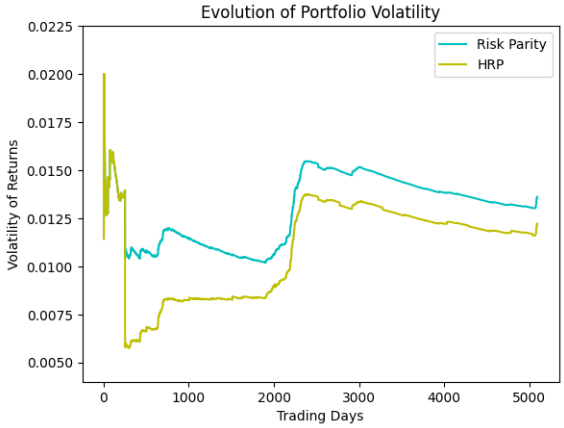
Although HRP presents a significant improvement over Risk Parity, it is approximately equal return-wise to the Benchmark. Also, it was outperformed by the naive Max Sharpe strategy.

Evidently, the MV models with ML were among the best in terms of returns, especially the Max Sharpe ML strategy.

B. Portfolio Volatilites

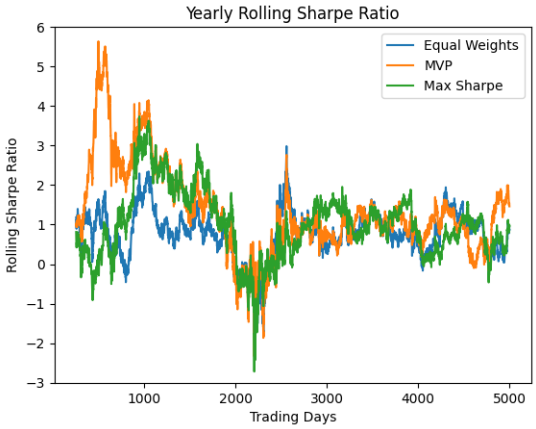


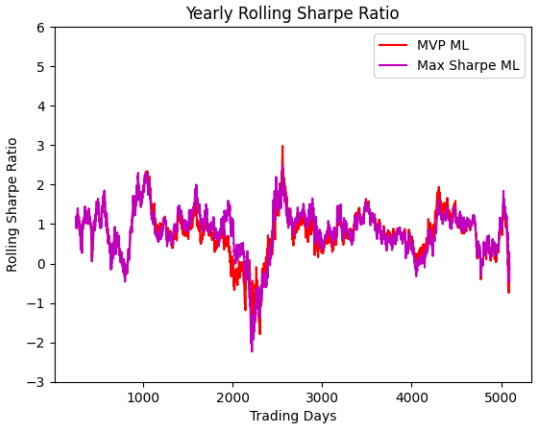


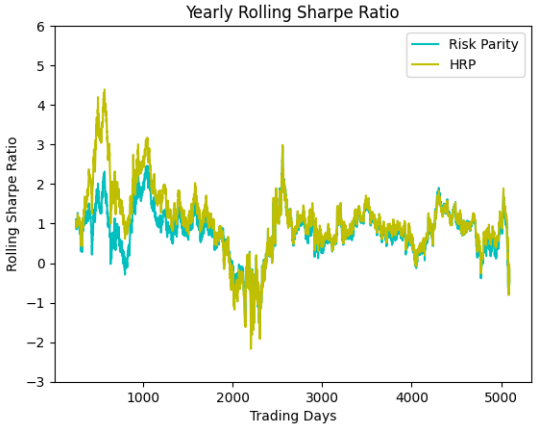


Evidently, the naive MVP and the HRP models were the most effective at minimizing the volatility (risk) of the portfolio. Close to them but not as good was the Risk Parity portfolio. The Equal Weights benchmark is worse at minimizing risks than the Naive Max Sharpe strategy. Finally, the Mean Variance approaches that used Machine Learning were highly risky, even the MVP ML, which is only a bit more effective than the benchmark at minimizing risk. Max Sharpe ML resulted in the most risky strategy.

C. Sharpe Ratios







These Yearly Rolling Sharpe Ratios show the evolution of the risk-return trade-offs of the corresponding strategies across the invested period. The metrics are calculated by taking the rolling average portfolio returns with a 252 window

divided by the average rolling portfolio volatility with the same window length (portfolio returns and volatility obtained from the back-testing algorithm). The ratio of these metrics (the rolling Sharpe Ratio) is annualized using the simple method of multiplying the metric by $\sqrt{252}$, which works only with certain distributional assumptions that not always hold [14]. Because of this, and because we assume a constant risk-free rate of 0 (which is only true on certain time periods), we know that these Sharpe Ratios can be overstated. However, these metrics are still useful for comparability of the strategies because the calculations are all consistent, and because all calculations are scaled by the same constant. Yet, note that the numbers yielded by the Sharpe Ratios should not be considered for an investment advice.

These Sharpe Ratios still show that the HRP and the Naive MVP strategies are very efficient at achieving risk-adjusted returns. At several time periods, they achieve the highest Sharpe Ratios across all the strategies. In many other time periods, however, all the strategies revolve around the same level of risk-adjusted returns. The Mean Variance methods with ML did not achieve the best risk-return trade-offs.

IV. Discussion

Evidently, machine learning improve their Naive counterparts.

On the one hand, HRP performed better than Risk Parity in every single way. It achieved higher returns while effectively having lower volatility over time. This is what lead to a higher yearly rolling Sharpe Ratio for the whole period. It achieved significantly better risk-adjusted returns. By this analysis there is no reason why one should not use Hierarchical Risk Parity over the Risk Parity approach.

Besides, the Machine Learning variation of Mean-Variance drastically improved returns in comparison to Naive Mean Variance, and, as a matter of fact, in comparison to all portfolio strategies. Note that, MV with ML was the most constrained strategy, since, as we said before, it was only allowed to make use of the models and move away from the Equally Weighted portfolio from the 4th year onwards. Still, their returns have been the highest, particularly the Max Sharpe ML strategy. The MVP with ML ended up slightly below the Naive Max Sharpe Strategy in terms of returns, but this suggests that without those extra constraints, this ML strategy would have probably beaten the Naive Max sharpe method even though MVP is a much more risk-averse investment profile.

Across models, we see that the HRP model also outperformed the Naive MVP model, not only by means of returns, but, for most of the time, it also achieved a higher Sharpe Ratio, except for the beginning where the MVP achieved an unusually high Sharpe ratio. In tough economic periods, HRP, MVP and Risk Parity were the best performers. But during bull markets they are all surpassed.

HRP could barely beat the benchmark, and it did not reach the naive Max Sharpe strategy, even though it is a machine learning algorithm. Having said that, the Max Sharpe strategy did not achieve a higher Sharpe Ratio than HRP or MVP, meaning that its risk-adjusted returns were not necesarilly better, it needed to take up on more risk to achieve better returns.

This leads us to the final argument. MV with ML strategies performed the best in terms of returns, but they also took up on the highest risks to achieve those returns. Overall, the paid off, due to the effectiveness of the forecasting methods. The forecasting methods allowed the strategy to take up on more calculated risks without losing big, suggesting that time series forecasting methods are much better for predicting Mean Variance parameters than other statistical procedures.

Overall, we can conclude that Machine Learning methods are definitely an improvement, in most cases, for upgrading the portfolio optimization strategies of investors. These methods also target both risk averse and risk taking investors, shown by the HRP model and the Max Sharpe with ML model.

We can also conclude that even when using strategies that seek to maximize the Sharpe Ratio, in practice, because it is much harder to forecast future returns than future volatility (the former's estimation errors are usually larger) [4], one can achieve a better risk-return trade-off by seeking to minimize the risk through MVP or HRP, since, in several periods, they show a better Yearly Sharpe Ratio than the risk-taking strategies.

Acknowledgment

Thanks to the professors Andrej Novak and Andrija Stajduhar of the Capstone course of the Zagreb School of Economics and Management for the consultations and the advice for completing this project.

References

[1] Markowitz, H. (1952). Portfolio selection. The Journal of Finance, 7(1), 77. https://doi.org/10.2307/2975974

[2] López de Prado, M. (2016). Building diversified portfolios that outperform out of sample. The Journal of Portfolio Management, 42(4), 59–69. https://doi.org/10.3905/jpm.2016.42.4.059

[3] Dai, Z. (2019). A closer look at the minimum-variance portfolio optimization model. Mathematical Problems in Engineering, 2019, 1–8. https://doi.org/10.1155/2019/1452762

[4] Jagannathan, R., & Ma, T. (2003). Risk reduction in large portfolios: Why imposing the wrong constraints helps. The Journal of Finance, 58(4), 1651–1683. https://doi.org/10.1111/1540-6261.00580

[5] Sharpe, W. F. (1966). Mutual Fund Performance. The Journal of Business, 39(S1), 119. https://doi.org/10.1086/294846

[6] Facebook. (n.d.). Prophet. Retrieved from https://facebook.github.io/prophet/

[7] Taylor SJ, Letham B. 2017. Forecasting at scale. PeerJ Preprints 5:e3190v2 https://doi.org/10.7287/peerj.preprints.3190v2

[8] Cont, R. (2007). Volatility clustering in financial markets: Empirical facts and agent-based models. Long Memory in Economics, 289–309. https://doi.org/10.1007/3-540-34625-2_10

[9] Machine Learning Mastery. (n.d.). Time series forecasting with Prophet in Python. Retrieved from https://machinelearningmastery.com/time-series-forecasting-with-prophet-in-python/

[10] Machine Learning Mastery. (n.d.). Develop ARCH and GARCH models for time series forecasting in Python. Retrieved from https://machinelearningmastery.com/develop-arch-and-garch-models-for-time-series-forecasting-in-python/

[11] Machine Learning Mastery. (n.d.). Time series forecast study with Python: Monthly sales of French champagne. Retrieved from https://machinelearningmastery.com/time-series-forecast-study-python-monthly-sales-french-champagne/

[12] White noise and random walks in time series analysis. QuantStart. (n.d.). https://www.quantstart.com/articles/White-Noise-and-Random-Walks-in-Time-Series-Analysis/
[13] Vyas, A. (2023). The hierarchical risk parity algorithm: An introduction. Hudson & Thames. https://hudsonthames.org/an-introduction-to-the-hierarchical-risk-parity-algorithm/
[14] Lo, A. W. (2002). The statistics of Sharpe ratios. Financial Analysts Journal, 58(4), 36–52. https://doi.org/10.2469/faj.v58.n4.2453
[15] Weber, A. (2017). Annual Risk Measures and Related Statistics. SSRN. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3054517
[16] Cris Velasquez. (2021, August 4). Optimize portfolio performance with risk parity rebalancing in Python. Medium. https://medium.com/@crisvelasquez/optimize-portfolio-performance-with-risk-parity-rebalancing-in-python-b9931d9e785b
[17] Cris Velasquez. (2021, July 15). Optimizing portfolio allocation with hierarchical risk parity in Python. Medium. https://medium.com/@crisvelasquez/optimizing-portfolio-allocation-with-hierarchical-risk-parity-in-python-19b1813af618
[18] Velasquez, C. (2021, August 2). Portfolio optimization with Python: Mean-Variance optimization (MVO) and Markowitz's efficient frontier. https://medium.com/@phindulo60/portfolio-optimization-with-python-mean-variance-optimization-mvo-and-markowitzs-efficient-64acb3b61ef6
[19] NUS Fintech Lab. (n.d.). ML optimisation for portfolio allocation. Retrieved from https://medium.com/@nusfintech.ml/ml-optimisation-for-portfolio-allocation-9da34e7fe6b1
[20] Tidy Finance. (n.d.). Constrained optimization and backtesting. Retrieved from https://www.tidy-finance.org/python/constrained-optimization-and-backtesting.html#out-of-sample-backtesting
[21] Towards Data Science. (n.d.). Overcoming bias-variance in machine learning. Retrieved from https://towardsdatascience.com/overcoming-bias-variance-in-machine-learning-31169dc649ed