A Work Project, presented as part of the requirements for the Award of a Master's degree in Finance from the Nova School of Business and Economics.

EVALUATION OF AN UNSUPERVISED LEARNING APPROACH FOR PORTFOLIO OPTIMIZATION

DARIO SANDRUCCI | 48436

Work project carried out under the supervision of:

DANIELE D'ARIENZO

12/01/2023

Abstract

Throughout this directed research, we aim to identify opportunities for machine learning to support portfolio optimization. Based on a thorough literature review we decide to pursue an unsupervised learning approach and test its performance by conducting benchmarking against classic portfolio optimization techniques. To ensure the validity of our findings we explore the model's robustness by conducting an array of experiments. In summary, we deem our version of the clustering algorithm to provide a suitable investment framework for return-focused investors with lower risk aversion. We suggest further research towards mitigating the algorithm's inconsistencies and exploring additional tuning methodologies.

Keywords: Unsupervised Learning, K-Means, Omega Ratio, Minkowski Distance, Portfolio Optimization

This work used infrastructure and resources funded by Fundação para a Ciência e a Tecnologia (UID/ECO/00124/2013, UID/ECO/00124/2019 and Social Sciences DataLab, Project 22209), POR Lisboa (LISBOA-01-0145-FEDER-007722 and Social Sciences DataLab, Project 22209) and POR Norte (Social Sciences DataLab, Project 22209).

Link to Algorithm

All relevant code and data to the algorithmic solutions presented throughout this report are stored on Github.com. They are accessible through the following LINK.

1 Introduction

With recent developments in financial technology, a major focus of quantitative finance is built around developing Machine Learning (hereafter ML) technologies to predict price movement and construct optimal portfolios, beating the performance of human fund managers. It comes as no surprise that the finance industry has been at the development forefront for prediction methodologies. In the early days, practitioners solely relied on their intellect to identify investment opportunities and execute trades. With the introduction of statistical models, especially the CAPM, every high-yielding strategy was tied to relatively high market betas and therefore risk exposure. The upsides of this simple and intuitive prediction methodology were quickly challenged by the rise of market anomalies, amongst these the predominant "value-effect", creating abnormal returns that were not able to be explained by the simple exposure to market returns. The introduction of multi-factor models which attempted to capture these anomalies in additional factors solved some of the issues in the short run but only at a certain cost. With the rising number of noisy and highly correlated return predictors needed for these models, the thread of overfitting and thus, producing unstable and unreliable predictions grew exponentially. Consequently, some researchers expressed the need for different methods from cross-sectional regressions and portfolio sorts. Here ML presented itself as particularly adequate as through regularization it provides means to contain overfitting biases, find complicated patterns and relations, and handle high-dimensional sets of predictor sets (Gu, Kelly and Xiu 2020). These potentials for ML in finance provide the inspirational basis for this directed research and will accompany us along the entire project. Throughout this report, we aim in particular to identify opportunities of ML for optimal asset allocation in portfolio management. The first part explores possible algorithms that suit our problem statement, identifies and analyzes parallel works on this subject, and finally derives a precise research objective. The following chapters document the implementation of the research objective into

an algorithm, evaluate whether it represents an upside to conventional optimization techniques, and investigate its robustness.

2 Theoretical Background

This second chapter performs a brief literature review of ML applications for finance. Starting with general definitions of ML we explore the opportunities and limitations of algorithms for finance, analyze the current state of research in related works, and finally derive the precise research objective for this directed research.

2.1 Introduction to Machine Learning

Most relevant literature defines ML as a subgroup of Artificial Intelligence, focused on the ability to learn without being explicitly programmed (Samuel 1953). In traditional programming software requires precise instructions, predefined by the manual inputs of a software engineer. For complex problem statements such as facial recognition, the instructions become very time-consuming and almost impossible to program through human inputs. ML promises to solve these issues by letting the computer learn from experience autonomously and program itself accordingly. Concerning its application scope, early practitioners stated *"the function of a machine learning system can be descriptive, meaning that the system uses the data to explain what happened; predictive, meaning the system uses the data to predict what will happen, or prescriptive, meaning the system will use the data to make suggestions about what action to take",* (Malone, Rus und Laubacher 2020). ML can be categorized into supervised, unsupervised, and reinforced learning algorithms. Supervised learning attempts to recognize patterns in labeled input data and apply these to predict the labels of a different unlabeled dataset. A proven application is the identification of fraudulent banking transactions based on past data of fraudulent and non-fraudulent instances. These algorithms range from simple linear regression functions to complex neural networks. Unsupervised learning (hereafter UL) algorithms explore unlabeled data for patterns, often identifying trends that users were not

explicitly looking for. The program scans the data points of a given set and determines whether and how they can be divided into subgroups. An example could be a marketing agency looking to divide its customer audience into segments for tailored advertisements. The algorithms determine data point similarity based on distance metrics between the underlying features and group them accordingly. Reinforced learning trains through trial and error, utilizing a reward system to incentivice the machine to determine the best set of actions. Exemplary applications are in the development of autonomous vehicles, where the computer learns from mistakes and avoids those for future trials. Within the scope of portfolio management, especially UL has established itself as a valuable tool to facilitate decision-making. We deem that for the purpose of our research this algorithm type could pose a suitable tool to analyze securities, identify patterns, and ultimately build a portfolio with optimal asset allocation.

2.2 Clustering Algorithms in Finance

As previously mentioned, UL algorithms analyze unlabeled datasets and assign the contained data points in subgroups based on their features, a process that in ML is called clustering. Clustering applications arise naturally in most processes in finance. The technique is widely applied to pattern recognition and anomaly detection. An investor may be keen to evaluate given assets based on a multiples method, which requires the identification of feasible counterparties for comparisons. Risk managers are concerned with maintaining a diversified portfolio and therefore avoiding the concertation of risk into securities with common traits (Baker und Filbeck 2015). A Trader may be interested in determining whether a particular rally or sell-off is idiosyncratic or might affect an array of multiple securities. It comes as no surprise that clustering analysis has developed into one of the most innovative means of portfolio selection. We deem that for our problem statement cluster analysis could pose a valuable opportunity to enable better portfolio allocation against conventional optimization techniques.

2.3 Opportunities and Limitations of Clustering for Asset Management

The key challenge of this directed research is to identify ML methodologies for portfolio optimization. In the previous chapters, we outlined clustering as a potential algorithm to support portfolio selection and create performant strategies. Supported by various studies, k-means has positioned itself as one of the most prominent tools in this area (Léon, et al. 2017). In this chapter, we will review the basic concepts of this algorithm, examine the underlying opportunities and challenges, and conclude the applicability to our problem statement. K-means is one of the most popular UL techniques, which divides a set of data points into a prespecified number "k" of clusters (Sharma, et al. 2019). Depending on the initialization technique, a quantity of "k" centroids is positioned within the dataset. The data points are then assigned to their closest centroid, forming a cluster. The centroids are then recalculated to constitute the center point of their respective cluster. Especially in early iterations this usually results in bigger centroid shifts and thus, some data points changing their cluster affiliation. This process is repeated iteratively until the centroids cease to move after recalculation. For portfolio selection, k-means includes several advantages. By construction k-means is an easily understandable and computationally efficient algorithm and therefore, adept to handle large datasets within a reasonable scope of time and resources. For application, it also features great flexibility and can work on a variety of different data structures. Some research also mentions positive aspects concerning data filtering and robustness to noise due to the finiteness of sample size (Tolaa, et al. 2008). A great number of professionals also highlight the potential of k-means for dimensionality reduction (Snow 2020). Much like methods such as Principle Component Analysis (PCA), the algorithm takes in a dataset with several features and reduces it to a lower dimensionality. Essentially, by providing a cluster assignment to each datapoint, k-means reduces the dataset to one dimension of the basis of the given feature inputs. Despite, these clear upsides, the methodology comes at a given cost of limitations. Most notably k-means requires

the number of clusters “k” to search for as an input (Scikit Learn 2022). For most problems, this parameter can’t be derived in a logical manner which poses a challenge for model tuning. Moreover, the algorithm does not guarantee a feasible and robust outcome as by default the initialization, the placement of centroids within the dataset, is random (Scikit Learn 2022). Some literature also warns about the algorithm’s sensitivity to outliers and dissuades its utilization for unprocessed datasets. Lastly, we should mention a critical view that recently gained popularity. It implies that real-world data often can’t be dissected into distinct clusters (Burney, et al. 2019). On basis of this view, an array of new so-called *“fuzzy”* clustering algorithms has been developed. Rather than allocating hard cluster assignments, these algorithms calculate the statistical probability of individual data points belonging to respective clusters (Duarte and De Castro 2020). As done by many parallel works on this topic, for this project we assume that asset data can indeed be classified into fixed clusters. Despite all assumptions, we are still left with the main challenge of k-means, being its initialization. An entire research field evolved around this issue of choosing an adequate number of clusters, providing a selection of simpler and more sophisticated methodologies to derive feasible input parameters. We will further elaborate on some applicable options to our problem statement in the implementation chapter. In summary, k-means provides us with a reliable, efficient, and flexible algorithm, suitable to the experimental approach we aspire to conduct throughout this research.

2.4 Related Work

The field of portfolio optimization is a widely researched area, being one of the most prominent issues in quantitative finance (Léon, et al. 2017). A large scope of research and implemented methodologies is based on Markowitz’s early works around portfolio selection (Ponsich, López Jaimes and Coello 2013). Observing his proposed framework, it is possible to make the logical implication of using multi-objective algorithms for such a problem statement (Duarte and De

Castro 2020). As clustering can be defined as an optimization task (Grötschel and Wakabayashi 1989), applying respective algorithms to solve an optimization problem for portfolio allocation is a valid inference (Duarte and De Castro 2020). Using this assumption as an inception point to our search for related work, we determine that there has been a vast number of contributions (Ponsich, López Jaimes and Coello 2013). The notions put out by Mantegna (Mantegna 1999), who was the first to suggest and employ time series clustering techniques using the correlation matrix to solve the portfolio allocation problem, were later used as cluster preprocessing by de Prado (M. L. de Prado 2016) and Raffinot (Raffinot 2018). The suggested strategy proved to be reliable in determining which types of securities are most vulnerable to and impacted by economic variables. De Prado presented a hierarchical allocation method based on the idea of hierarchical clustering, employing the data from the covariance matrix to produce more diversified portfolios than those produced by conventional techniques. The outcomes were portfolios that were more diversified and less volatile. Raffinot examined the Single Linkage, Complete Linkage, Average Linkage, Ward's Method, and Directed Bubble clustering techniques based on the work of de Prado. The author concluded that portfolios built using hierarchical clustering have superior risk-adjusted returns and are more diversified. By reflecting on the work of Mantegna and how the method he outlined became the foundation for time series clustering, Marti et al. (Marti, et al. 2021) analyzed financial market clustering. The analysis offers suggestions on clustering applications for trading, risk management, portfolio selection, and financial policy formulation. More recently León et al. (Léon, et al. 2017) tested a variety of clustering algorithms for risk-adjusted portfolio construction. They found that clustering algorithms generally produce less volatile portfolio performance compared to their traditional counterparts such as the mean-variance optimization. Concentrating on more recent works we identify that most ideas are still based on de Prado's earlier works around hierarchical clustering (de Prado 2016). Researchers such as Duarte (Duarte and De Castro 2020) or even

de Prado (de Prado 2020) himself made use of an allocation workflow including intra and inter-cluster allocation schemes. A concept that de Prado describes as *"Nested Clustering Algorithms"*. This concept shall also lay the foundation for our research project. Based on suggestions from the related literature, in the following chapter, we will derive a clear research objective.

2.5 Research Objective of the Directed Research

Exploring potential research objectives, it becomes apparent that the field of clustering algorithms for asset allocation has been widely covered by literature. The thorough literature review of the past chapter will thus, constitute a valuable reference in discovering niches to base our research and create a value-added project. Our inception points will be the *"Nested Clustering Algorithm"* employed by de Prado (de Prado 2020) and others. The study by Duarte (Duarte and De Castro 2020) employed such a framework in combination with a k-medoids algorithm. With regards to performance, they found that the proposed methodology was outperforming the returns of traditional options, such as the minimum variance portfolio, at a cost of slightly higher risk. The research indicates further research potential for applications of different clustering algorithms and trials in different market environments. The works by León et al. (Léon, et al. 2017) employ an array of different k-means clustering algorithms to benchmark against classic portfolio allocation theories. Concluding that cluster algorithms generally produce portfolios with less volatility, they highlight the possibility of utilizing the Omega (hereafter OR) rather than the Sharpe ratio for portfolio optimization in further research. Moreover, they propose experiments with different distance metrics instead of using the classical correlation matrix as basis for similarity identification. This proposal reflects the research propositions of some other prominent works in the field such as Raffinot (Raffinot 2018). In general, we observe that the research niches can be categorized. The most hinted concept is trying different distance metrics for similarity analysis. Within the scope of our

research we will explore using the frequently suggested Minkowski distance (Equation 1). The main benefit we seek to gain from this distance metric is its p-parameter which allows for optimal model tuning making it especially adept for ML applications. Furthermore, parallel research incentivizes experiments that derive adaptions of conventional algorithms and optimization methods. For our project, we will opt for the aforementioned k-means algorithm in combination with the proposed Omega optimization (Léon, et al. 2017). This represents a particularly promising optimization measure as it provides a balanced metric (Equation 2) between downside-risk and upside-potential (Winton Capital Management 2003), enabling a measure of risk-adjusted returns. Further aspects regarding our model specifications will be presented in the upcoming chapter. To guarantee the robustness of our model, next to conventional benchmarking techniques we also will leverage the proposal of testing our model in different market environments. Summarizing this chapter on the theoretical background, we were able to establish a solid basis to build our machine learning model for portfolio allocation. In the upcoming chapter, we will explain the detailed implementation of our concept.

3 Model Implementation

This section documents the implementation of our k-means-based clustering strategy to achieve optimal portfolio allocation. After a short review of the dataset, the sections present data processing techniques and distance metrics before detailing the implementation of the actual investment strategy.

3.1 Data Description

The dataset we chose to employ our investment strategy on includes the daily stock returns of the S&P 500 index' (hereafter SPY) components over the period of the last 10 years. The selected time range provides us with sufficient information to learn and test model robustness while remaining within the feasible range of our processing capabilities. The selected data

amounts to a total of 503[1][2] securities that we will use for our portfolio construction. For the train-test split methodology, we will follow the conventional train-validation-test split approach, separating our dataset along the time series into three components. The default proportions of the components will be 60-20-20 respectively, although we will explore other configurations to determine robustness in our analysis chapter. Our model will be trained on the train set and evaluated on the validation set. After completing training and tuning of the model we will finalize a recommended model specification to be applied to the test set. As indicated in the theoretical background chapter we will use the correlation matrix of the underlying SPY-component returns as input to our clustering algorithm to calculate the optimal allocation.

3.2 Denoising and Detoning

The validity of model output is mainly driven by two determinants: the quality of input data and the fit of model characteristics with the underlying problem statement. Our procedure involves the k-means clustering algorithm with input data in form of a correlation matrix. By default, these two components come with their challenges. The k-means algorithm is not immune to outliers, which could potentially result in a distortion of findings (Shrifan, Akbar and Isa 2022). Moreover, the correlation matrix by construction features significant amounts of noise that pose a challenge to identifying actual signal. In this chapter, we will discuss two methodologies proposed in de Prado's work (de Prado 2020) to process our input into a format that both reflects the essential signaling of our data and accounts for our algorithm's characteristics. The first mean for processing the empirical correlation matrix is a denoising mechanism. It is based on the *"Marchenko Pastur Theorem",* a theory of random matrices that

[1] Only stocks were used that composed the S&P500 on the date of research (03.11.2022)

[2] Number might deviate due to potentially missing data

describes the asymptotic behavior of singular values in large rectangular random matrices (Yaskov 2016). Essentially, it provides through the characteristics of the underlying dataset a range of eigenvalues that would for the corresponding correlation matrix be deemed insignificant. Thus, we form a framework to discriminate between eigenvalues associated with noise and those associated to signal. We implement our denoising strategy through the constant residual eigenvalue method (de Prado 2020), left with a denoised correlation matrix. Figure 1 depicts a comparison between eigenvalues before and after denoising. The second concept we apply to our correlation matrix is the detoning method, which also was proposed by de Prado. The basis for this instrument follows the thought that financial matrices usually follow at least one underlying market component. Thus, every element of the correlation matrix is at least in part driven by a common force. Data that features a strong market component is recommended to be denoted, supporting the algorithm in finding dissimilarities across clusters. As our dataset is composed of the largest companies that make up and thus, follow the U.S. market we will perform this step for our purposes. By removing one eigenvector, the detoned correlation matrix becomes singular. Since the majority of methods do not need the invertibility of the correlation matrix, this is not an issue for clustering applications (de Prado 2020). However, it is notable that mean-variance portfolio optimization cannot be done directly using a detoned correlation matrix but rather would require a preprocessing step.

3.3 Distance Metrics

To enable our clustering algorithm to identify patterns in the inserted dataset it is essential to provide an underlying metric that describes the relationship between two separate variables. A helpful indicator of linear codependence is correlation. After being denoised and detoned, a correlation matrix can provide crucial structural details about a system. This although, is only possible after some necessary technical adjustments as described by de Prado. Technically, correlation is not a metric as it doesn’t fulfill the criteria of nonnegativity and triangle inequality

(Dubrovskiy 2022). De Prado emphasizes the importance of metrics as they provide an *"intuitive topology"* to a given set. Without this mentioned topology comparing non-metric measures would increasingly lead to incoherent outcomes. In light of this perspective, we can observe the need for a suitable distance metric to describe the stock return relations that compose our dataset. For our instance we will use the Minkowski distance (hereafter MD) metric due to its high adequacy for ML applications, providing an optimal baseline for model tuning through its hyperparameter "p" (Vandeginste, et al. 1998). In essence, the MD combines the concepts of the Euclidian and Manhattan distance metrics (Kamble and Dale 2022). Through the p-parameter which usually but not necessarily lies between 1 and 2, we can adjust the scope of influence for both underlying distance metrics to return the final Minkowski distance. With a p-parameter of 1 the MD accurately replicates the Manhattan distance. Vice versa, a value of 2 implies as distance metric corresponding to the Euclidian distance. This characteristic provides us with a helpful tool for model tuning, allowing us to adapt to specific application cases in an optimal manner. To understand how to utilize our p-parameter, in a first experiment we want to identify whether a unique parameter provides superior performance to our SPY dataset. Therefore, we implement a basic version of our model, further detailed in the upcoming sections, over different investment periods and with different p-parameters. Subsequently, we run a cross-sectional regression over the models' performance metrics with the respective p-parameter as the independent variable. In Table 1 we can observe the regression output for our experiment. With very low R2 and high p-values for those regressions, we determine that the p-parameter does not have a statistically significant effect on model performance. Thus, we can't determine that a specific p-parameter will deliver superior performance over our dataset. We conclude that we need to tune the p-parameter within each training phase to suit the individual application cases. Chapter 3.6 provides further details on the p-parameter tuning procedure within our algorithm.

3.4 Portfolio Construction

In this section we lay out the methodology for optimal asset allocation within our portfolio. It will be applied in combination with the clustering algorithm presented in section 3.5. While classical portfolio optimization is usually built around minimum variance or maximum Sharpe ratio optimization, for our problem statement we want to test a different approach. Our research will optimize according to the OR as proposed in related works such as (Bernard, Vanduffel und Ye 2019) or (Sehgal und Mehra 2021). The mentioned ratio was first introduced by Con Keating and William F. Shadwick (Keating and Shadwick 2002) and represents a performance metric for asset returns based on a return target threshold. In essence, the ratio is calculated as the probability-weighted ratio of gains versus losses for that defined threshold. An investor would choose an asset with a higher Omega ratio as it generates greater returns relative to losses for his selected return benchmark. This metric promises to provide a useful basis for portfolio optimization as it balances between downside-risk protection and exploitation of upside potential. Compared to the Sharpe ratio it also considers by construction all moments of return distributions, while the former only incorporates the first two (Winton Capital Management 2003). Observations have shown that information drawn from higher moments of return distributions can contradict the conclusions drawn from the traditional mean-variance analysis. Thus, some research argues that highly significant information can be overlooked by overreliance on the Sharpe and even Sortino ratio (Winton Capital Management 2003). To test this hypothesis, we will implement this concept into our algorithm framework and compare it to classic portfolio optimization techniques such as mean-variance optimization.

3.5 Nested Clustering Algorithm

This section describes the implementation of the ML-based algorithm to obtain the optimal portfolio allocation. For our instance the base framework is the Nested Clustering Optimization (hereafter NCO) methodology introduced by de Prado (de Prado 2016). As its description might

suggest, the structure of this algorithm follows a nested approach to perform the optimal portfolio allocation. It aims to counter the effects of the *"Markowitz's curse"* (de Prado 2020). This theory states that although Markowitz's theory for portfolio optimization is mathematically sound, its practical implementation has issues. Due to noise and signal, particularly large financial covariance matrices yield increasing condition numbers. Their inverses amplify related estimation errors, resulting in unstable calculation outputs. A little change to the observation's matrix may result in completely different allocations. To avoid these effects the NCO algorithm splits the allocation problem into multiple smaller portions of the problem. The proposed framework proceeds in three steps to yield the final and optimal allocation from the inserted distance matrix. The first step consists of a clustering algorithm that identifies patterns in the underlying distance matrix and clusters the set into a defined number of subsets. In our case, we apply the k-means algorithm, as we deemed it suitable to our problem statement in the Theoretical Background chapter. As previously mentioned, the issue with this clustering algorithm arises with its hyperparameter "k" which has to be specified upon initialization. To identify the optimal number of clusters for our given dataset, we will utilize a combination of three different methodologies, each yielding a projected number of clusters suitable for the problem statement. We then select the proposed "k" parameter which yields the most distinct and accurate clustering. The respective methods to support the initialization are the Elbow Method (Scikit YB 2019), the Silhouette Score (Towards Datascience 2019) and the Davies-Bouldin Index (Davies and Bouldin 1979). After obtaining our clusters and thus, dividing the allocation problem into smaller portions, we calculate our optimal intra-cluster weights. This can be done for any desired optimization objective. For our instance, we use the portfolio allocation methodology based on the OR, laid out in the previous section. In the third and final step, we calculate the inter-cluster weights, according to the same methodology utilized for the intra-cluster weights. This allows us to take the product of inter and intra-cluster weights and

produce a final array of portfolio weights, proposing an optimal allocation. Figure 2 provides an overview of the NCO algorithm's procedures.

3.6 Investment Strategy

This section briefly explains the implementation procedure of the methodologies presented throughout this chapter. Our implementation follows the basic ML principle of train, validation, and test splitting. Before our point of strategy implementation, the algorithm learns the underlying pattern of the train set and proposes certain allocation possibilities. These then get tested over the validation set, which is situated right after the train set until the point of implementation. This allows for tuning of the model parameters and provides a final specification to be applied to the test set. The finalized model suggests an optimal allocation based on its learnings and invests over the test period accordingly, starting at the point of implementation. The performance metrics calculated throughout this test set allow us to determine our algorithm's relevance, especially compared to classic investment strategies. Figure 3 provides a detailed overview of the algorithm's workflow for strategy implementation. The following chapter analyses the performance resulting from this chapter's implementation specifications.

4 Performance Analysis

In this chapter, we explore and validate the performance of our ML-based investment strategy. Throughout the first half, we provide a benchmarking analysis of our algorithm's results compared to other classic strategies. Throughout the second part of our analysis, we test the robustness of our model by applying it to different scenarios and environments.

4.1 Benchmark Analysis

In this section we employ our main investment strategy, the OR optimized NCO algorithm, implementing our learnings about clustering algorithms and distance metrics from the previous

chapters. To enable a performance reference, we additionally implement some traditional investment strategies over the same investment period which use an identical basket of securities. We will implement an equal-weighted (hereafter EWP), a mean-variance optimized (hereafter MVO), and a risk-parity portfolio (hereafter RPP). Further, we provide the simple option of a direct investment into the SPY. The investment period for all portfolios will be over the test set. To identify their optimal weights the MVO, RPP, and NCO portfolios will use the train and validation sets as a basis. Concerning the learning technique of our main strategy, we will proceed with the training and validation phases as proposed in Chapter 3. For the threshold parameter of the OR, we use the mean annualized return of the underlying SPY over the past 10 years, as it provides an adequate reference benchmark. Initial tests show that indeed a threshold around the mean returns of the underlying index provides the best performance. Further distant parameters cause bad performance both with regard to risk and returns. After the implementation of all strategies, we are left with 4 constructed portfolios and one underlying index over the test period. A first look at the cumulative returns chart allows us to observe that all options are making a profit over the invested period between November 2020 and 2022 (Figure 4). Nevertheless, we are more interested in the performance spread of the NCO strategy to the reference portfolios. As a first impression, we detect that all portfolios are moving in highly similar patterns. The lowest correlation index between two of the observed portfolio's returns is indeed still over 83,6%. This makes sense as all portfolios (mostly) contain the same underlying securities, just with different weight specifications. Despite similar moving patterns we can identify some noticeable differences. In general, all constructed investment strategies were able to outperform the SPY's returns. The best performance is provided by our NCO strategy which considerably outperformed all other portfolios. This though appears to come at the cost of higher volatility, indicated by steeper return movements compared to the other benchmarks. The options MVO, RPP, and EWP yield very similar results to one another,

featuring lower volatility while still outperforming the SPY throughout the entire investment period. As a first takeaway, we observe three different behavioral characteristics of portfolios. The groups consist of the NCO strategy, the classic portfolios, and the underlying Index. The precise performance metrics confirm our initial assumptions (Table 2). All strategies outperform the SPY's returns (ARet) by at least 3 percentage points, with our NCO strategy leading the spread with impressive 13.6 percentage points. Moreover, the NCO is almost able to yield alpha (Alpha) over the market returns (SPY), with p-values (p Alpha) relatively low but only significant at an 83% confidence level. Regarding the risk metrics, we can determine that the NCO strategy indeed featured higher volatility (AVol), with 5,5 percentage points to the closest reference. With regards to downside risk (MDD) though, it performed much closer to the other portfolios, featuring a spread of only 1.5 percentage points to the SPY portfolio. Regarding the risk-adjust payoff metrics, we can observe that despite its increased volatility the NCO portfolio represents a valuable investment opportunity, providing the best Sharpe (Sharpe) ratio by a considerable amount. Further, it yielded an outstanding Sortino ratio (Sortino) with spreads of as much as 83 percentage points to the reference portfolios, underlining its capabilities for downside protection. As anticipated from Figure 4 the reference group of MVO, RPP, and EWP provided rather similar metrics, outperforming the SPY in all performance categories. The great results of all portfolios over the market returns are amplified by high information ratios (IR). Especially the NCO stands out with a ratio over 1.0, implying great risk-adjusted returns compared to the market portfolio (SPY). In conclusion, we were able to detect that all built investment strategies provided considerable upsides against the market portfolio. Our NCO strategy was especially successful in risk-adjusted returns with downside protection. This reflects the findings of related literature mentioned in chapter 2.4. The RPP, MVP, and EWP portfolios provided a more risk-averse opportunity featuring lower volatilities and high information ratios. Despite these positive observations, we have to acknowledge that

neither of our portfolios provides a statistically significant alpha, nor are we able to confirm that the positive results would persist throughout different scenarios outside our test set. Therefore, throughout the remainder of this chapter, we will perform various robustness checks, to facilitate confidence in our findings.

4.2 Model Robustness

Throughout this section, we analyze the robustness of our NCO model over an array of different experiments. We start by testing the persistence of our cluster algorithm's pattern identification. By applying our algorithm to different investment periods and trying different train-test set split ratios, we aim to confirm our algorithm's robustness over different periods. Moreover, we compare the performances of the static base NCO strategy against a monthly rebalanced one to observe whether our base strategy provides stable results throughout time. Finally, we apply our strategy to a different market environment to determine robustness throughout environmental changes.

4.2.1 Cluster Persistence

In this section we analyze the persistency of our clustering algorithm in two experiments. In the first experiment, we initialize our clustering algorithm many times with equal specifications and inputs to determine whether it consistently finds the same clusters. In the second experiment, we analyze whether our algorithm finds similar clusters throughout time, tested in monthly increments for each iteration. To measure consistency, we implement a persistence counter function that counts for every iteration the number of equal securities in each cluster compared to the initial initialization. For our first experiment, we see that our algorithm performs very robustly, consistently yielding almost identical cluster compositions (Figure 5). In the second experiment, we notice that the cluster consistency does not persist as much throughout time (Figure 6). From the perspective that economic environments change over time, this does appear to be logical. On a positive note, we detect that all cluster's persistence

scores move with each other, indicating a common underlying driver for their movements. With each time shift, all clusters are supposedly altered similarly to one another, providing optimal allocations to changing environments. Moreover, cluster changes happen linearly rather than producing volatile results with each period change. This lets us believe that the algorithm follows a certain pattern rather than providing random and volatile allocations each period. This also confirms that indeed we were able to counter the mentioned Markowitz's curse to a certain extent. Summarized, our underlying clustering algorithm behaves robustly and adequately to its application scope.

4.2.2 Investment Periods

In this section, we apply our algorithm to multiple investment periods in an attempt to prove that our findings are not specific to our initial investment period. We choose 6 random periods within the last 10 years and apply our NCO algorithm to them. As a benchmark we will use again the SPY, which is composed of the same securities. In Table 3 we can observe the performance of the portfolios over the 6 different investment periods. In terms of returns, the NCO algorithm outperforms the SPY in 5 of the 6 test cases. It was able to achieve considerable return spreads in 3 cases of up to 72 percentage points, even resulting in statistically abnormal returns at an 85% confidence level for one period. Concerning the risk metrics, as in our initial implementation, the NCO algorithm consistently provides higher volatility compared to the benchmark. Despite that, the performance for maximum drawdowns was almost identical to that of the SPY, except for period 5, reaffirming the algorithm's strength for downside protection. For the overall risk-return metrics though, we notice certain scenarios that appear to not suit the NCO algorithm. Compared to the SPY the NCO portfolio only provided higher Sharpe and Sortino Ratios in 4 out of 6 test cases. Summarized, we can mostly reaffirm the NCO algorithm's characteristics of a solid risk-return strategy, especially for its strength in downside protection. Nevertheless, as simple economic drivers such as a period's underlying

volatility can't explain our algorithm's inconsistencies, we need to the view our positive findings under a more critical perspective.

4.2.3 Train Test Split Ratios

In this section, we explore whether the size ratios of the respective train and test splits have an impact on model performance. We try 7 different model train-test split configurations for our NCO algorithm over the same period of the initial implementation (Chapter 4.1.). In Table 4 we can see the different model performances against the usual SPY benchmark portfolio. With regards to return performance, we detect a solid positive spread over all configurations, with a slight trend towards higher spreads with increasing train set proportions. Concerning risk metrics, we detect slightly higher volatility than the benchmark with a mostly stable spread over all configurations. The performance for downside risk protection (MDD) behaves similarly with stable spreads over all configurations. For the risk-return ratios, we observe a familiar pattern with slightly better spreads towards higher train set proportions. Concludingly, our model does not appear to be impacted that much by train-test-split configurations, although we can observe slightly better performance when the train set is bigger than the test set.

4.2.4 Rebalanced vs. Static Weights

For the next experiment of this chapter, we want to evaluate whether a monthly rebalancing of the portfolio weights throughout time can deliver inferior performances to our initial option with statics weights. This allows us to determine whether our base algorithm produces stable results or if each period change produces considerably different results. Therefore, in addition to our base NCO strategy, we will implement a strategy that invests over the same test period but with monthly rebalancing. As usual, we will provide the SPY portfolio as a benchmark. Looking at the cumulative returns chart (Figure 7) we can observe that the static and rebalanced strategies perform very similarly, both outperforming the SPY throughout the entire test set. The initial impressions are supported by the corresponding performance metrics (Table 5), with

both options providing similar results. At a closer look though we observe a slight edge of the rebalanced strategy. From a returns perspective, the rebalanced algorithm performs a bit better, delivering about 2.5 percentage points more in mean annualized returns. Despite that, both options still lie at least 13 percentage points above the SPY. These aspects are also reflected in the statistical significance of the generated alpha by both strategies. Again, the rebalanced option provides slightly better results with a p-value of 12.4% compared to the 16.9 % of its static counterpart. Concerning risk metrics and downside protection, the performance is almost identical for both volatility and maximum drawdown. With slightly better returns and similar risk thus, the rebalanced strategy appears to overall provide a slight edge, yielding a better Sharpe, Sortino as well as Information Ratio. Despite that slight edge, we can determine that rebalancing did not contribute significant upsides or even deviations to the initial model's performance. This leads us to believe that our base model's predictions are quite robust and adept for investments over longer holding periods.

4.2.5 Application to different Markets

As the last part of our robustness check, we want to determine whether our model and respective findings are only applicable to our S&P500 dataset, or if we can detect robustness towards application environments. We opt to explore an application to the Japanese Nikkei225 index. As for our base model's implementation, we will use the NCO algorithm to create an optimally allocated portfolio of the underlying index' components. The benchmark portfolios for this experiment, as for our S&P500-based portfolio (Chapter 4.1.), will be the MVO, RPP, and EWP portfolios, as well as the Nikkei225 index itself. Looking at the cumulative returns chart (Figure 8), we detect that all constructed portfolios outperform the underlying index, with the NCO portfolio leading the cumulative returns. Observing the performance metrics (Table 6), we identify that the NCO algorithm leads the mean returns, despite not providing a statistically significant alpha over the Nikkei225 index. Concerning volatility, the NCO algorithm performs

slightly better than the underlying index. Despite that, the MVO, RPP and EWP portfolios still provide inferior risk aversion. For the maximum drawdown the NCO portfolio performs quite similarly to all other benchmarks, despite yielding the most negative value. Thus, our algorithm can defend its title as the leading risk-adjusted performer, topping the Sharpe, Sortino, and Information ratios by a significant amount. Overall, we observe quite similar results to our base experiment, featuring increased volatility, higher performance ratios, and good downside protection. In conclusion, we can state that our algorithm proved to be quite robust to environmental shifts, providing similar positive results to our base experiment despite being implemented in a completely different market environment.

5 Conclusions

In this chapter, we summarize our project's scope, discuss findings, raise limitations, and propose areas for further research.

5.1 Project Summary

In this project, we aimed to explore ML techniques to improve asset allocation for portfolio management. As a baseline for our research, we provided a theoretical background to ML applications for asset management, particularly UL. We identified the opportunity of applying a k-means-based NCO framework to our problem statement and followed the current research's recommendation of testing other distance metrics and optimization techniques. For the distance metric, we deemed the Minkowski distance as suitable, enabling model tuning through its p-parameter. We opted for the Omega ratio as optimization score for inter and intra-cluster weights. The ratio promised a balance between downside-risk protection and exploitation of upside potential. Further, it provided means to extract information out of all moments of return distributions. We implemented our developed strategy over the components of the S&P500 index, performed benchmarking and performance tests, and evaluated the model's robustness through multiple experiments.

5.2 Discussion of Findings

In the benchmarking analysis, we found that our algorithm provides solid risk-adjusted returns visible in the Sharpe Ratio, outperforming all benchmark portfolios and the underlying index. This also reflects the findings of parallel work to this research field. Although the created portfolio provided great relative downside protection visible in the maximum drawdown and Sortino ratio metrics, it was notable that the NCO's performance came at the cost of increased volatility when compared to all other benchmarks. This is a contradicting finding to some of the presented research which deemed clustering as a mean for lower volatility through diversification. It appears that the Omega ratio optimization rather yields a "protected-return-maximization" strategy, than being a pure risk diversification technique. Thus, the NCO portfolio could represent a valuable option for investors looking for good risk-adjusted returns but who are not averse to taking on slightly more risk. In our robustness checks, the results proved to be valid for a variety of train-test split configurations, although we noticed a slight tendency towards better performance with higher proportions of train set sizes. The application to other economic environments proved successful as the algorithm provided similar positive results compared to the initial application. Furthermore, by analyzing the underlying clustering algorithm, we were able to deem the clustering as robust and adequate for its application. Despite all positives, applying the algorithm to different periods raised the issue that the algorithm's performance does fall short in some instances. In a brief analysis, we were not able to identify an underlying economic driver causing these inconsistencies. In summary, we find that our version of the NCO algorithm provides a solid framework to create performant return-oriented strategies. Nevertheless, additional research is required to determine its preferred application scenario. In the last section, we elaborate further on this research's limitations and possibilities for further development.

5.3 Limitations and Suggestions for Further Research

Our research findings come with certain limitations, both to our methodology and equipment. In general, our optimization techniques for the intra and inter-cluster weight optimization are based on Monte-Carlo simulations which by construction require high computational performance to be effective. Since in the scope of this project we were only able to use an average personal computer, the precision of the optimizations and thus, the accuracy of our findings cannot be guaranteed to the fullest extent. This could constitute one of the causes for our algorithm's inconsistencies across periods. For further research, we recommend extending exploration on these model inconsistency issues. One could explore the implications of different OR thresholds for model performance and explore their potential as countermeasures. Moreover, it would be possible to extend the tuning of the Minkowski distance parameter "p" with a larger scope of possible values and smaller intervals. With regards to the NCO framework, next to trying other clustering algorithms it would be interesting to evaluate whether separating found clusters into additional clusters, thus, adding a layer to the hierarchical clustering structure, would add different and potentially increased performance.

6 List of References

Snow, Derek. 2020. "Machine Learning in Asset Management—Part 1: Portfolio Construction—Trading Strategies." *The Journal of Financial Data Science.*

Burney, S. M. Aqil, Tahseen Jilani, Humera Tariq, Zeeshan Asim, Usman Amjad, and Syed Shah Mohammad. 2019. "A Portfolio Optimization Algorithm Using Fuzzy Granularity Based Clustering." *BRAIN – Broad Research in Artificial Intelligence and Neuroscience* 159-173.

Gu, Shihao, Bryan Kelly, and Dacheng Xiu. 2020. "Empirical asset pricing via machine learning." *The Review of Financial Studies 33* 2223-2273.

Samuel, Arthur L. 1953. "Computing Bit by Bit or Digital Computers Made Easy." *Proceedings of the IRE.*

Malone, Thomas W., Daniela Rus, and Robert Laubacher. 2020. *Artificial Intelligence and the Future of Work.* Research Brief, MIT Sloan School of Management.

Baker, H. Kent, and Greg Filbeck. 2015. *Investment Risk Management.* Oxford University Press.

Léon, Diego, Arbey Aragón, Javier Sandoval, Gérman Hernández, Andrés Arévalo, and Jaime Nino. 2017. "Clustering Algorithm for Risk-Adjusted Portfolio Constructions." *Procedia Computer Science 108C* (Procedia Computer Science) 1334-1343.

Sharma, Dhruv, Krishnaiya Thulasiraman, Di Wu, and John N. Jiang. 2019. "A network science-based k-means++ clustering method for power systems network equivalence." *Computational Social Networks.*

Tolaa, Vincenzo, Fabrizio Lilloc, Mauro Gallegatia, and Rosario N. Mantegnac. 2008. "Cluster analysis for portfolio optimization." *Journal of Economic Dynamics & Control 32* 235–258.

Scikit Learn. 2022. *Kmeans Algorithm Documentation.* Accessed 10 2022. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.

Duarte, Flávio Gabriel, and Leandro Nunes De Castro. 2020. “A Framework to Perform Asset Allocation Based on Partitional Clustering.” *IEEE Access.*

Ponsich, Antonin, Antonio López Jaimes, and Carlos Coello. 2013. “A Survey on Multiobjective Evolutionary Algorithms for the Solution of the Portfolio Optimization Problem and Other Finance and Economics Applications.” *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council 17(3)* 321-344.

Grötschel, M., and Y. Wakabayashi. 1989. “A cutting plane algorithm for a clustering problem.” *Math. Program. vol. 45* 59-96.

Mantegna, R. N. 1999. "Hierarchical structure in financial markets." *Eur. Phys. J. B, vol. 11, no. 1* 193-197.

de Prado, Marcos López. 2016. "Building Diversified Portfolios that Outperform Out of Sample." *The Journal of Portfolio Management* 59-69.

Raffinot, Thomas. 2018. "Hierarchical Clustering-Based Asset Allocation." *The Journal of Portfolio Management* 89-99.

Marti, G., F. Nielsen, M. Binkowski, and P. Donnat. 2021. "A review of two decades of correlations, hierarchies, networks and clustering in financial markets." *Chapter in Progress in Information Geometry: Theory and Applications* 245-274.

de Prado, Marcos López. 2020. *Machine Learning for Asset Managers.* Cambridge: Cambridge University Press.

Winton Capital Management. 2003. "Assessing CTA Quality with the Omega Performance Measure."

Shrifan, Nawaf H.M.M., Muhammad F. Akbar, and Nor Ashidi Mat Isa. 2022. "An adaptive outlier removal aided k-means clustering algorithm." *Journal of King Saud University - Computer and Information Sciences, Volume 34, Issue 8, Part B* 6365-6376.

Yaskov, Pavel. 2016. "A short proof of the Marchenko–Pastur theorem." *Comptes Rendus Mathematique, Volume 354, Issue 3* 319-322.

Dubrovskiy, Stanislav. 2022. *Quasi-triangle inequality for absolute correlation distance.* arXiv.

Kamble, Vaishali H., and Manisha P. Dale. 2022. "Machine learning approach for longitudinal face recognition of children." *Machine Learning for Biometrics* (Machine Learning for Biometrics).

Vandeginste, B.G.M., D.L. Massart, L.M.C. Buydens, S. De Jong, P.J. Lewi, and J. Smeyers-Verbeke. 1998. "Handbook of Chemometrics and Qualimetrics: Part B, Chapter 31." *Data Handling in Science and Technology, Volume 20, Part 2,* 87-160.

Sehgal, Ruchika, and Aparna Mehra. 2021. "Robust reward–risk ratio portfolio optimization." *International Transactions in Operational Research* 2169–2190.

Bernard, Carole, Steven Vanduffel, and Jiang Ye. 2019. "Optimal strategies under Omega ratio." *European Journal of Operational Research* 755-767.

Keating, Con, and William F. Shadwick. 2002. "An Introduction to Omega." *The Finance Development Centre.*

Scikit YB. 2019. *Elbow Method.* Accessed 2022. https://www.scikit-yb.org/en/latest/api/cluster/elbow.html.

Towards Datascience. 2019. *Silhouette Coefficient.* Accessed 2022. https://towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10c.

Davies, David L., and Donald W. Bouldin. 1979. "A Cluster Separation Measure." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 224-227. Accessed 2022. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.davies_bouldin_score.html.

7 Appendix

Equation 1: Minkowski Distance

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

Equation 2: Omega Ratio

$$\Omega = \frac{\int_r^b \big(1 - F(x)\big) dx}{\int_a^r F(x) dx}$$

Figure 1: Correlation Matrix Eigenvalues Original vs. Denoised

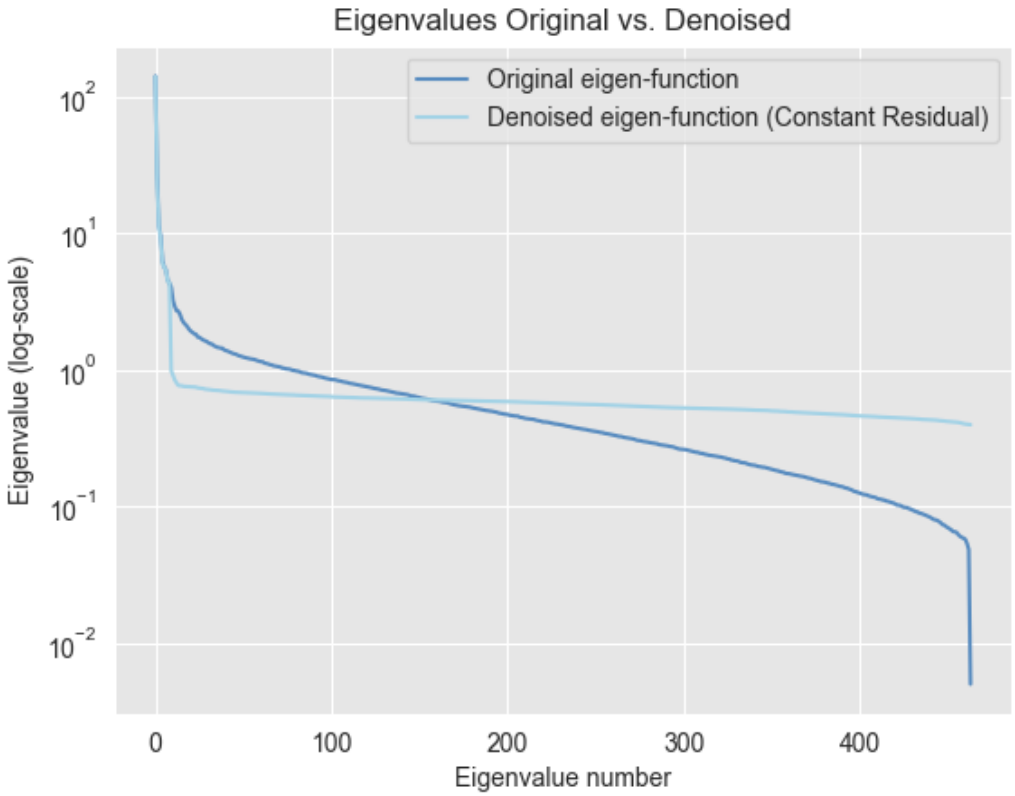

Table 1: Cross-sectional Regression for the Minkowski P-Parameters

	r2	beta	p-value beta
ARet	0.0000	-0.0055	0.9734
AVol	0.0000	-0.0009	0.9875
Alpha	0.0000	-0.0000	0.9813
p Alpha	0.0027	0.0349	0.6794
Beta	0.0000	-0.0014	0.9793
Sharpe	0.0002	0.0438	0.9202
Sortino	0.0005	0.1474	0.8518
IR	0.0000	-0.0061	0.9895
MDD	0.0000	-0.0000	0.9979

Figure 2: NCO Algorithm Flowchart

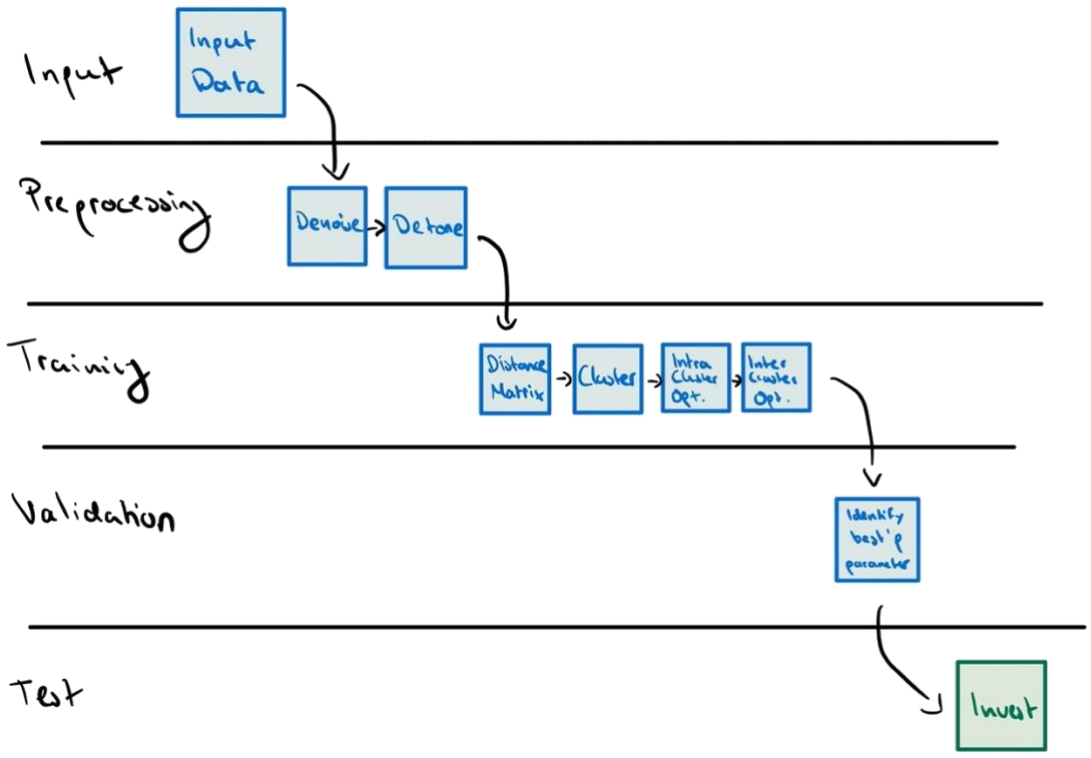

Figure 3: Strategy Implementation

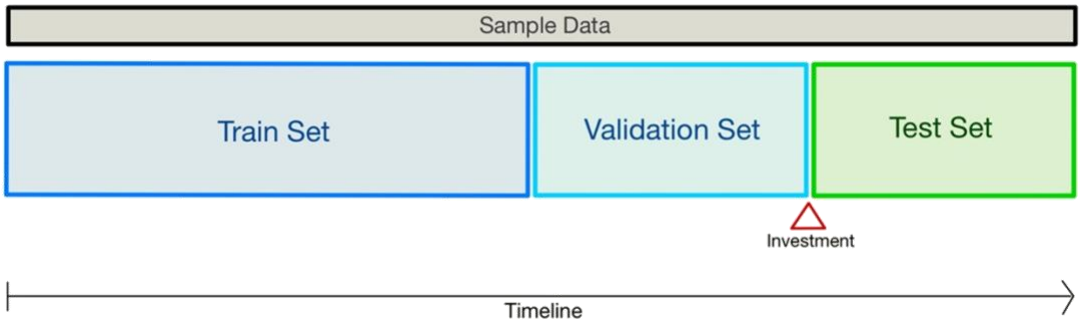

Figure 4: Benchmarking NCO vs. Classic Portfolios

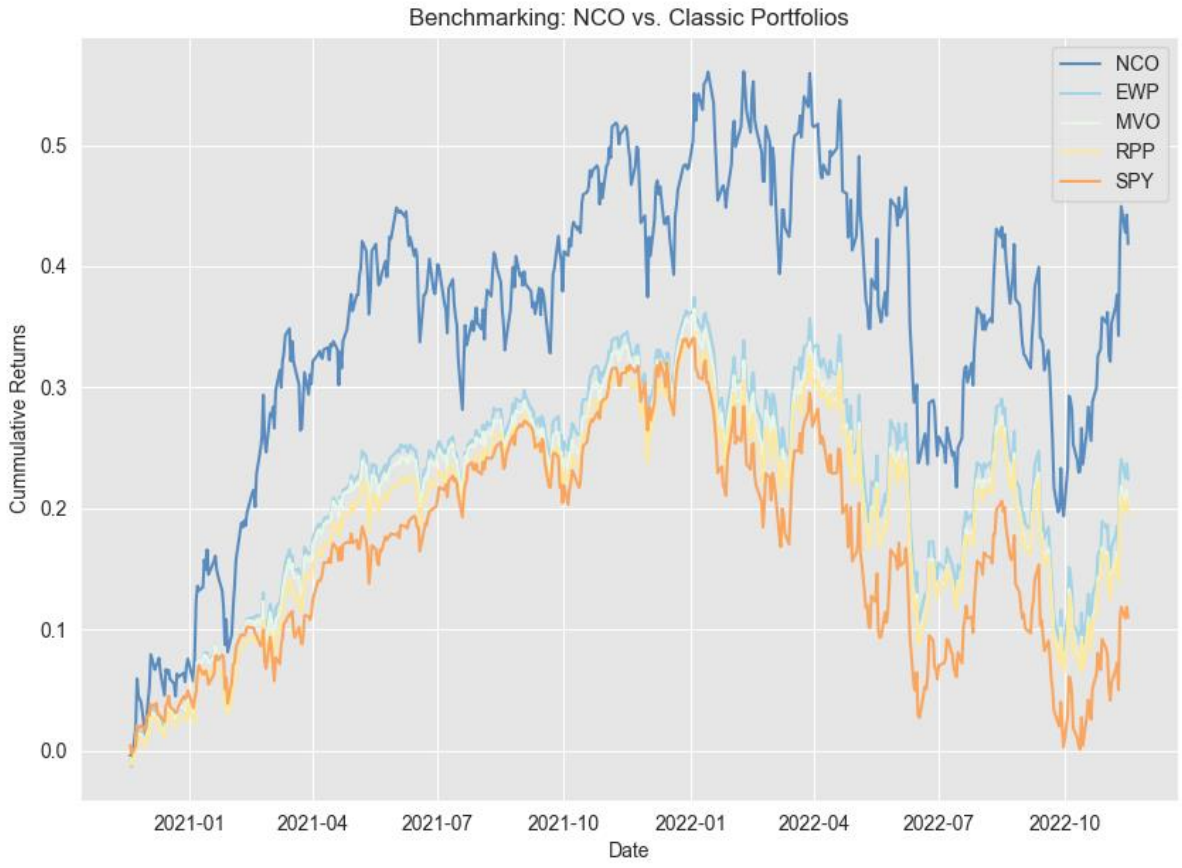

Table 2: Benchmarking NCO vs. Classic Portfolios

	ARet	AVol	Alpha	p Alpha	Beta	Sharpe	Sortino	IR	MDD
SPY	0.070600	0.190900	0.000000	0.961700	1.000000	0.369800	0.524500	0.000000	-0.075300
NCO	0.206200	0.248800	0.000500	0.169100	1.077500	0.828500	1.356000	1.006900	-0.090400
EWP	0.118600	0.186400	0.000200	0.138300	0.931900	0.636300	0.942800	0.948400	-0.071400
MVO	0.109300	0.184700	0.000200	0.168100	0.930500	0.592000	0.871800	0.861000	-0.070300
RPP	0.105600	0.178500	0.000200	0.186400	0.896100	0.591300	0.864300	0.752000	-0.069500

Figure 5: Cluster Persistence Static (Legend shows cluster number)

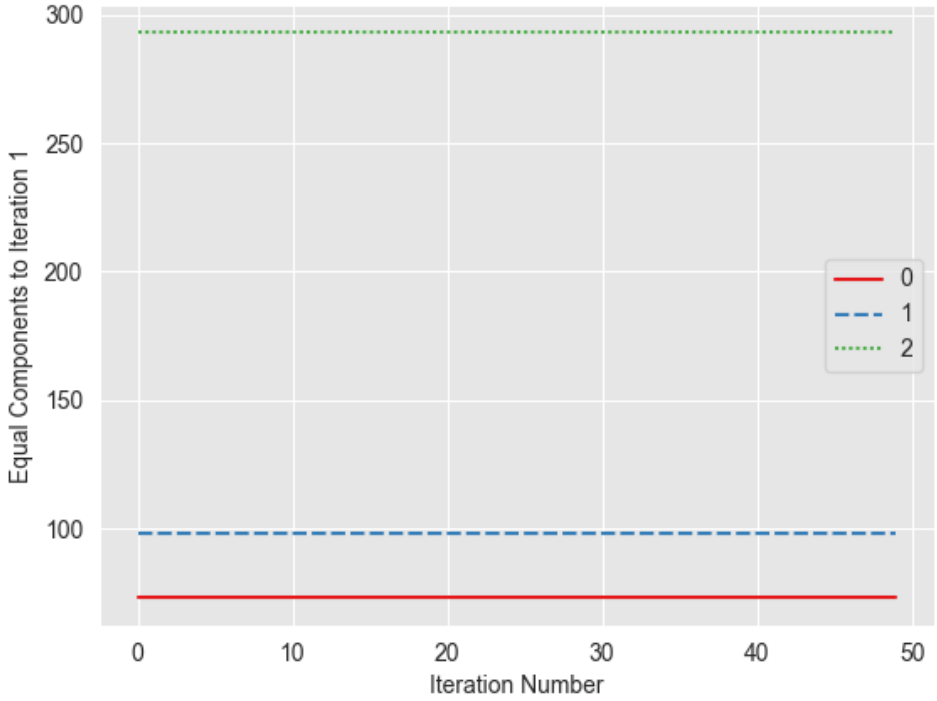

Figure 6: Cluster Persistence over Time (Legend shows cluster number)

Table 3: NCO Robustness over Time

	ARet	AVol	Alpha	p Alpha	Beta	Sharpe	Sortino	IR	MDD
NCO Period 1	0.1888	0.1078	-0.0001	0.5486	1.1304	1.7510	2.2673	-0.3549	-0.0362
SPY Period 1	0.1847	0.0906	-0.0000	0.0797	1.0000	2.0373	2.5936	0.0000	-0.0318
NCO Period 2	-0.1111	0.2755	0.0002	0.8417	1.2683	-0.4031	-0.6951	0.1060	-0.0671
SPY Period 2	-0.1269	0.1865	-0.0000	0.3665	1.0000	-0.6806	-1.0973	0.0000	-0.0444
NCO Period 3	0.2934	0.1003	0.0003	0.4684	1.3700	2.9259	3.6533	2.2194	-0.0348
SPY Period 3	0.1642	0.0619	0.0000	0.0458	1.0000	2.6526	2.8387	0.0000	-0.0255
NCO Period 4	0.1412	0.1574	-0.0004	0.3091	1.2767	0.8975	1.1600	-0.7968	-0.0548
SPY Period 4	0.1946	0.1142	-0.0000	0.0000	1.0000	1.7034	2.0934	0.0000	-0.0370
NCO Period 5	1.1311	0.3079	0.0026	0.1535	1.1504	3.6736	9.8748	2.8107	-0.1395
SPY Period 5	0.4116	0.1500	0.0000	0.0000	1.0000	2.7437	4.0880	0.0000	-0.0468
NCO Period 6	0.2803	0.3333	0.0003	0.6121	1.2681	0.8408	1.5691	1.0881	-0.0957
SPY Period 6	0.1628	0.2544	0.0000	0.1342	1.0000	0.6398	1.1635	0.0000	-0.0705

Table 4: NCO Robustness over Train-Test Splits

	ARet	AVol	Alpha	p Alpha	Beta	Sharpe	Sortino	IR	MDD
NCO Split [0.1, 0.9]	0.1591	0.2091	0.0002	0.0395	1.1128	0.7612	0.9305	0.8325	-0.2176
SPY Split [0.1, 0.9]	0.1041	0.1791	0.0000	0.0000	1.0000	0.5810	0.6917	0.0000	-0.1949
NCO Split [0.25, 0.75]	0.1523	0.2494	0.0001	0.4062	1.1845	0.6108	0.7726	0.4521	-0.2401
SPY Split [0.25, 0.75]	0.1007	0.1895	-0.0000	0.0000	1.0000	0.5315	0.6301	0.0000	-0.1949
NCO Split [0.4, 0.6]	0.1568	0.2583	0.0001	0.7194	1.1785	0.6071	0.7542	0.3219	-0.2408
SPY Split [0.4, 0.6]	0.1192	0.1977	0.0000	0.1480	1.0000	0.6027	0.7016	0.0000	-0.1949
NCO Split [0.5, 0.5]	0.1232	0.3125	-0.0000	0.8924	1.2323	0.3942	0.5211	0.0861	-0.2857
SPY Split [0.5, 0.5]	0.1082	0.2143	-0.0000	0.5097	1.0000	0.5048	0.6029	0.0000	-0.1949
NCO Split [0.6, 0.4]	0.2419	0.2859	0.0004	0.0222	1.1789	0.8462	1.0812	1.2680	-0.2074
SPY Split [0.6, 0.4]	0.1228	0.2284	0.0000	0.0025	1.0000	0.5375	0.6478	0.0000	-0.1949
NCO Split [0.75, 0.25]	0.3114	0.2748	0.0006	0.1247	1.1530	1.1335	1.7813	1.0841	-0.1487
SPY Split [0.75, 0.25]	0.1335	0.1928	-0.0000	0.0966	1.0000	0.6922	0.9435	0.0000	-0.0833
NCO Split [0.9, 0.1]	-0.0258	0.2808	0.0005	0.3207	1.0493	-0.0919	-0.1484	0.9384	-0.0868
SPY Split [0.9, 0.1]	-0.1422	0.2403	-0.0000	0.4636	1.0000	-0.5918	-0.9454	0.0000	-0.0753

Figure 7: NCO Static vs. Rebalanced

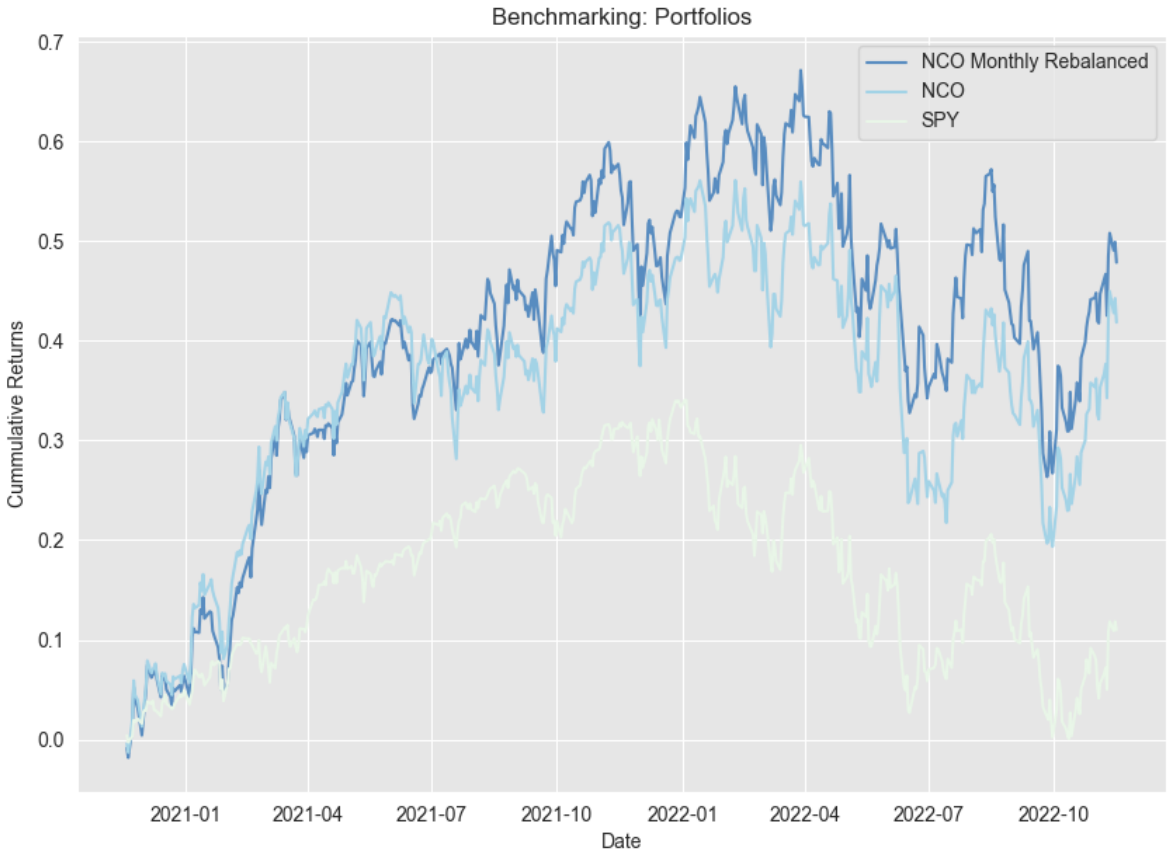

Table 5: NCO Static vs. Rebalanced

	ARet	AVol	Alpha	p Alpha	Beta	Sharpe	Sortino	IR	MDD
SPY	0.070600	0.190900	0.000000	0.961700	1.000000	0.369800	0.524500	0.000000	-0.075300
NCO Monthly Rebalanced	0.225300	0.242100	0.000600	0.124000	1.009100	0.930500	1.506200	1.097000	-0.088800
NCO	0.206200	0.248800	0.000500	0.169100	1.077500	0.828500	1.356000	1.006900	-0.090400

Figure 8: NCO Environmental Robustness

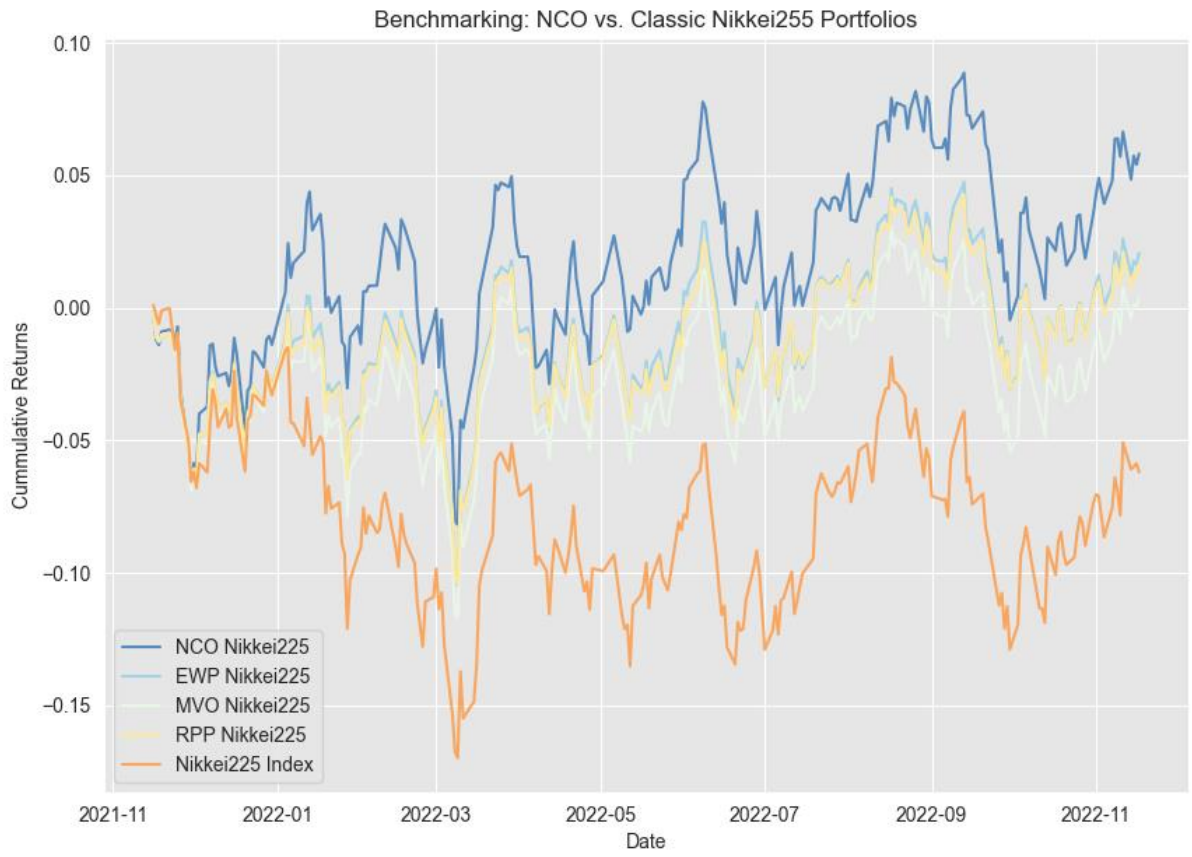

Table 6: NCO Environmental Robustness

	ARet	AVol	Alpha	p Alpha	Beta	Sharpe	Sortino	IR	MDD
NCO Nikkei225	0.076300	0.191900	0.000400	0.275900	0.785600	0.397800	0.633400	1.098600	-0.079100
EWP Nikkei225	0.036900	0.178200	0.000300	0.326500	0.782900	0.206900	0.341600	0.955700	-0.070400
MVO Nikkei225	0.020400	0.181800	0.000200	0.364800	0.820500	0.112200	0.186400	0.896300	-0.070000
RPP Nikkei225	0.031800	0.171900	0.000300	0.337100	0.759800	0.184900	0.308600	0.907300	-0.067600
Nikkei225 Index	-0.043800	0.208600	-0.000000	0.512700	1.000000	-0.209900	-0.348100	0.000000	-0.069400