Cluster Analysis and Risk Factors to Optimize a Portfolios

Fanhao Kong

Abstract

This project focuses on the application of quantitative methods for portfolio optimization, with a specific emphasis on clustering analysis and risk factor exposures. By leveraging the Fama-French three-factor model, we identify the factor exposures of individual stocks and use the K-means algorithm to cluster stocks based on their similarities. The Markowitz mean-variance framework is then applied within each cluster to optimize portfolio weights, balancing risk and return.

To evaluate the robustness of the model, we conducted out-of-sample testing using unseen data from 2020 to 2023, following a training period from 2014 to 2019. The optimized portfolio demonstrated strong performance during favorable market conditions but exhibited volatility in later periods, highlighting both its strengths and limitations. Performance comparisons against the 1/N naive portfolio and the S&P 500 index were conducted using key metrics, including Sharpe ratio, information ratio, and maximum drawdown.

The results showcase the potential of integrating machine learning techniques, such as clustering and factor analysis, with traditional financial models for portfolio management. Future improvements, such as dynamic weight adjustments and rolling-window validation, could further enhance performance and address identified challenges.

Keywords

Portfolio Optimization, Clustering Analysis, Fama-French Three-Factor Model, Markowitz Mean-Variance

Introduction

The financial market is characterized by its inherent complexity and uncertainty, presenting challenges to investors seeking to maximize returns while managing risk. Over the years, portfolio optimization has emerged as a critical area in financial research, aiming to provide systematic solutions to these challenges. The introduction of modern portfolio theory by Markowitz (1952) laid the foundation for optimizing asset allocations through the mean-variance framework, balancing expected returns against associated risks. This approach remains a cornerstone in asset management, serving as a basis for many advanced optimization models.

In this project, we aim to bridge the gap between academic theories and practical applications

by integrating machine learning techniques, such as factor analysis, clustering, and portfolio optimization, into a cohesive framework. Specifically, we leverage the Fama-French Three-Factor Model to uncover the systematic risk exposures of individual assets, providing a robust basis for clustering stocks based on shared risk characteristics. This clustering step not only aids in dimensionality reduction but also allows for tailored optimization within each cluster, enhancing portfolio diversification.

The significance of this project lies in its ability to address the challenges of portfolio construction in contemporary markets. By combining advanced statistical models with machine learning methods, we demonstrate how clustering can improve portfolio efficiency and risk management. Furthermore, the project introduces a comparative analysis, evaluating the performance of the optimized portfolio against benchmarks such as the equal-weighted (1/N) portfolio and the S&P 500 Index. The results highlight the practical utility of the proposed methodology, offering insights into its advantages and potential limitations.

Through this study, we aim to contribute to the growing body of research on machine learning applications in finance, providing a replicable framework that integrates traditional financial theories with modern computational tools. This project underscores the importance of innovation in portfolio optimization, ensuring relevance in the ever-evolving landscape of global financial markets.

Literature Review

Theoretical Foundations

The **Mean-Variance Optimization Model** introduced by Harry Markowitz (1952) provides the mathematical foundation for balancing risk and return in portfolio construction. By identifying the efficient frontier, the model highlights the optimal allocation of assets to maximize returns for a given level of risk. However, traditional applications of the mean-variance model assume static risk and return, which can limit its adaptability in dynamic markets.

The **Fama-French Three-Factor Model** (1992) extends the **Capital Asset Pricing Model (CAPM)** by introducing size (SMB) and value (HML) factors, in addition to market risk, as drivers of asset returns. This model has become a cornerstone in explaining cross-sectional variations in stock returns and remains widely used for portfolio construction and performance attribution.

Clustering techniques, such as **K-means clustering**, have been applied in finance to group stocks based on similar characteristics. By simplifying the dataset into clusters, researchers can reduce the complexity of optimization and target specific risk profiles within each cluster.

Our Contribution

Building on this foundation, our project integrates these models into a cohesive framework:

Factor analysis identifies systematic risks, enhancing the interpretability of portfolio returns.

Clustering reduces complexity by grouping stocks with similar risk exposures, streamlining the optimization process.

Mean-variance optimization assigns weights within clusters to maximize returns and

minimize risk, overcoming estimation challenges through constrained optimization. Comprehensive evaluation metrics validate the performance of the optimized portfolio against traditional benchmarks, highlighting its advantages in real-world applications.

Data Description

Data Collection and Processing

1. Selection of Data and Timeframe

To ensure robustness and relevance in our analysis, we selected stock data from the S&P 500 index. The S&P 500 index represents a comprehensive benchmark of the U.S. equity market, making it ideal for studying factor exposures and portfolio optimization. The timeframe chosen spans from **January 2014 to January 2024** (10 years). This period captures various economic cycles, including bull and bear markets, ensuring the dataset is representative of different market conditions.

Monthly frequency was selected for the data to reduce noise and avoid the excessive computational complexity associated with daily data. A **minimum of 84 data points (7 years)** was required for any stock to be included in the dataset. This criterion ensures that stocks have sufficient historical data for robust modeling while accounting for newer stocks added to the S&P 500 during the period.

2. Randomized Stock Selection

To manage the scope of the project and computational efficiency, **50 stocks** were randomly sampled from the S&P 500 index. This random sampling helps ensure generalizability while maintaining a manageable dataset size. If any stock failed to meet the 7-year data criterion, additional stocks were randomly sampled until the dataset reached 50 valid stocks. This ensures a diverse sample representative of different sectors and market conditions.

3. Data Cleaning and Transformation

The raw stock data was sourced using **Yahoo Finance** via the yfinance API. The following steps were applied to ensure data quality:

Adjusted Close Price: Adjusted prices were used to account for corporate actions, such as dividends and splits, ensuring accurate returns calculations.

Monthly Returns: Returns were computed using percentage changes in adjusted closing prices. Monthly returns reduce short-term volatility and better align with the factor modeling approach. By aggregating daily fluctuations into monthly intervals, monthly returns smooth out short-term noise caused by market sentiment, news events, and low liquidity effects. This enhances the focus on long-term trends and structural risk factors, providing more stable inputs for factor regression and portfolio optimization. Additionally, monthly data better reflects practical investment practices, such as monthly rebalancing and performance evaluation.

File Organization: All raw stock data was stored in individual CSV files for reproducibility and potential reanalysis. Additionally, a consolidated CSV containing monthly returns for all stocks was created to streamline subsequent analysis.

4. Factor Data: Fama-French Three-Factor Model

To model systematic risks, the **Fama-French Three-Factor Model** was chosen. The Fama-

French three-factor model is an enhancement of the traditional Capital Asset Pricing Model (CAPM), addressing its limitations by introducing additional factors to explain stock returns (Fama and French, 1993). This model has been widely adopted in both academic research and industry due to its effectiveness in capturing risk premia associated with size and value. This model captures the impact of three key factors:

Market Factor (Mkt-RF): Excess return of the market portfolio relative to the risk-free rate.

Size Factor (SMB): The return difference between small-cap and large-cap stocks.

Value Factor (HML): The return difference between value stocks (high book-to-market ratio) and growth stocks.

The Fama-French data were sourced from the **Kenneth French Data Library**, containing monthly factor returns and the risk-free rate (Rf). The original data, available in monthly frequency, was processed to align with our stock returns data:

Dates were converted to a standard format for consistency.

Columns were renamed to intuitive labels (e.g., "Market," "Size," "Value," and "Risk-Free").

The processed data was saved for seamless integration into the factor modeling step.

5. Benchmark Data: S&P 500 Index

The **S&P 500 Index** data was downloaded to serve as a benchmark for portfolio performance evaluation. The index data provides the following:

Market Returns: Monthly returns were computed from adjusted closing prices of the index.

Benchmark Comparisons: The S&P 500's cumulative returns and risk metrics (e.g., Sharpe ratio, maximum drawdown) were compared against the optimized portfolio and an equally weighted (1/N) portfolio.

This benchmark allows for a clear evaluation of the added value generated by our portfolio optimization framework relative to a widely accepted market index.

6. Justification for the Methodology

Use of Monthly Data: Monthly data strikes a balance between capturing meaningful trends and reducing computational overhead. It avoids the noise inherent in daily data, which can obscure underlying patterns.

Inclusion of Fama-French Factors: The three-factor model simplifies the analysis while capturing the most critical systematic risks. This approach avoids overfitting and ensures the model remains interpretable.

Random Sampling of Stocks: By randomly selecting stocks, we minimize biases and ensure the results generalize across the market rather than being influenced by specific sectors or large-cap stocks.

Timeframe and Benchmarking: The selected timeframe and the inclusion of the S&P 500 index as a benchmark ensure that the results are not only historically robust but also practically relevant to modern investment decisions.

This data collection and processing framework ensures that the dataset is both high-quality and aligned with the objectives of factor analysis and portfolio optimization. By addressing key challenges, such as data sufficiency and computational manageability, we have laid a strong foundation for subsequent analyses.

Methodology

Factor Analysis

1. Data Source and Excess Returns Calculation

The factor analysis employs the Fama-French three-factor model and utilizes the following datasets:

Stock Returns: Monthly returns computed from adjusted closing prices. This reduces short-term volatility and aligns with the factor modeling approach.

Factor Data: The Fama-French factors include the Market factor (Mkt), the Size factor (SMB), and the Value factor (HML).

Excess Returns: Stock returns are adjusted by subtracting the risk-free rate (Rf) to remove the baseline influence.

To ensure robustness in calculations, zero values in the risk-free rate were replaced with a small positive value (0.001) to avoid numerical instabilities.

Ordinary Least Squares (OLS) regression is utilized to calculate the factor exposures for each stock. The factor exposures represent the sensitivity of a stock's returns to the Fama-French factors, including Market, Size, and Value. OLS regression is particularly suited for this analysis because it minimizes the sum of squared residuals, ensuring an optimal fit between the observed excess returns and the linear model. This approach helps quantify the relative importance of each risk factor for a given stock, providing insights into its risk-return profile. Mathematically, the regression equation is as follows:

$$r_{i,t}^{excess} = \alpha_i + \beta_{i,Mkt} \cdot Mkt_t + \beta_{i,Size} \cdot SMB_t + \beta_{i,Value} \cdot HML_t + \epsilon_{i,t}$$

Where:

- $r_{i,t}^{excess}$: Excess return of stock i at time t,
- α_i: Intercept term (unexplained return),
- $\beta_{i,Mkt}, \beta_{i,Size}, \beta_{i,Value}$: Factor loadings (exposures to Market, Size, and Value),
- Mkt_t, SMB_t, HML_t: The three Fama-French factors at time t,
- $\epsilon_{i,t}$: Residual term (idiosyncratic risk).

OLS regression ensures the estimated coefficients (β) are unbiased and efficient under normality assumptions, making it ideal for financial factor modeling.

2. Regression Model for Factor Exposure

Factor exposures were calculated using Ordinary Least Squares (OLS) regression. This approach quantifies the relationship between stock returns and the Fama-French factors. The regression model is formulated as:

$$r_{i,t} - R_f = \beta_0 + \beta_1(Mkt_t - R_f) + \beta_2 SMB_t + \beta_3 HML_t + \epsilon_{i,t}$$

where:

- $r_{i,t}$: Return of stock i at time t.
- R_f: Risk-free rate at time t.
- Mkt_t: Market return at time t.
- SMB_t: Size factor at time t, capturing the performance of small-cap vs. large-cap stocks.
- HML_t: Value factor at time t, capturing the performance of value vs. growth stocks.
- β_0: Intercept, representing unexplained returns.
- $\beta_1, \beta_2, \beta_3$: Factor loadings (exposures), indicating sensitivity to respective factors.
- $\epsilon_{i,t}$: Residual error term.

Factor exposures, represented by the β\betaβ coefficients in the regression, quantify the sensitivity of a stock's returns to each of the Fama-French factors. For example:

A high βi,Mkt indicates that the stock is highly correlated with the overall market movements.

A positive βi,Size suggests the stock behaves like a small-cap stock.

A positive βi,Value suggests the stock has value characteristics (high book-to-market ratio).

These exposures are crucial for understanding how different types of systematic risks affect each stock. They provide a foundation for making informed decisions in portfolio construction, where balancing risk exposures is critical.

3. Steps in Regression Analysis

Model Specification: The regression model incorporates three factors along with a constant term.

Valid Data Filtering: For each stock, rows with missing values in either returns or factors were excluded. A stock must have at least 30 valid observations to be included in the analysis.

OLS Estimation: The model was fitted for each stock using valid data points. The resulting coefficients represent the stock's exposure to the factors.

Result Storage: The regression coefficients were saved in a matrix, where:

Rows represent individual stocks.

Columns correspond to the intercept (const) and factor exposures (Marke, Size, Value).

4. Output

The output of this analysis is a CSV file containing the factor exposures for all analyzed stocks. Each row represents a stock, and the columns include:

const: The intercept term (β0).

Market: Exposure to the market factor (β1).

Size: Exposure to the size factor (β2).

Value: Exposure to the value factor (β3).

	const	Market	Size	Value
MCO	−0.094448	0.013134	0.001047	−0.000493
BAC	−0.099904	0.013803	0.004285	0.013077
AKAM	−0.095391	0.008406	−0.002043	0.000845
PFG	−0.100381	0.012813	0.004681	0.010836
CPRT	−0.085616	0.011584	0.002280	−0.000260

5. Key Insights

The regression results allow us to interpret the sensitivity of individual stocks to systematic risk factors.

The calculated factor exposures serve as the feature set for the subsequent clustering analysis. Using factor exposures for clustering ensures that stocks are grouped based on their risk profiles rather than arbitrary characteristics like sector or market capitalization. By analyzing stocks with similar systematic risk characteristics, we can create targeted investment strategies that are tailored to specific risk-return trade-offs. This approach also aligns with prior research suggesting that grouping by risk factors can enhance portfolio diversification and performance (DeMiguel et al., 2009; Fama and French, 1993).

Clustering Analysis

1. Principal Component Analysis (PCA) and Dimensionality Reduction

To prepare for clustering, we employed Principal Component Analysis (PCA) to reduce the dimensionality of the factor exposure dataset. Factor exposures are derived from our earlier Ordinary Least Squares (OLS) regression against the Fama-French three-factor model. PCA helps simplify the data by transforming the high-dimensional factor exposure data into a two-dimensional representation. This reduces computational complexity and facilitates visualization while retaining the majority of the variance (80.8% as explained by the first two principal components: 48.89% for the first and 31.91% for the second, appendix clustering analysis code).

2. K-Means Clustering

We applied the K-Means clustering algorithm to the reduced two-dimensional data. The method partitions the stocks into distinct groups (or clusters) based on their similarities in factor exposures. Clustering allows us to identify stocks with similar factor sensitivities, which could potentially behave similarly under market conditions.

The clustering process was carried out using:

K=5 Clustering: Initially, the data was clustered into five groups, which yielded a clear separation of stocks with differing factor sensitivities.

Silhouette Score Analysis: To evaluate the quality of clustering, we computed silhouette scores for values of k ranging from 2 to 9. The silhouette score measures how similar an object is to its cluster compared to other clusters. k=3k=3k=3 was identified as the optimal choice with the highest silhouette score (0.4933).

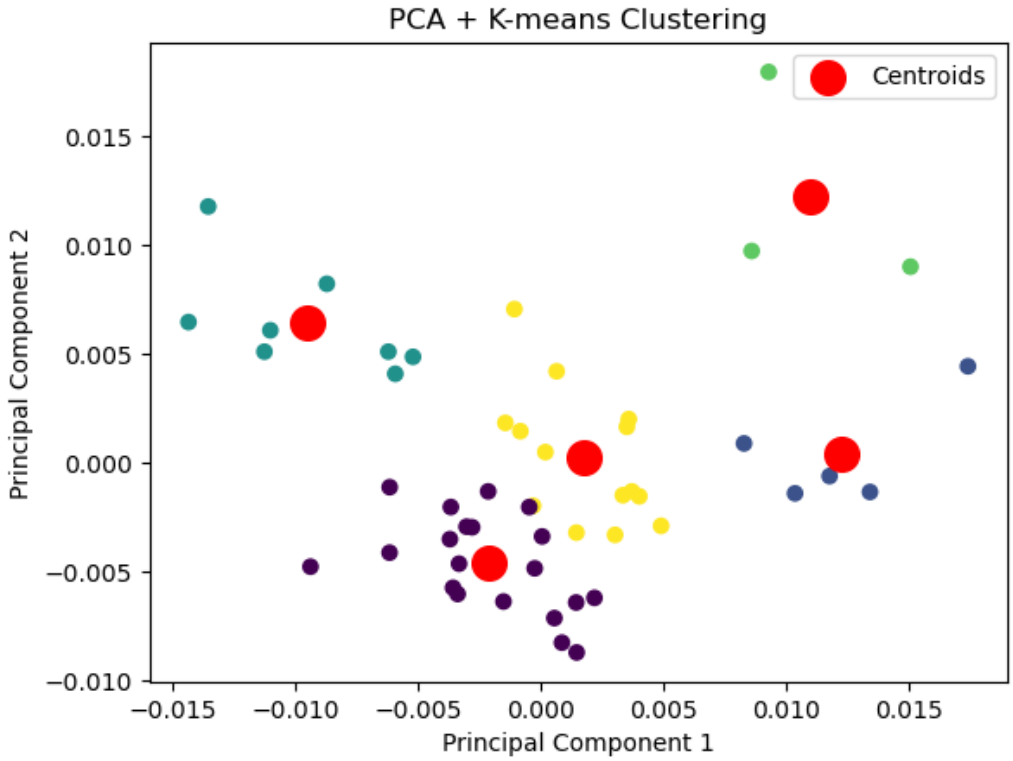
K=3 Clustering: Based on the silhouette score, we re-clustered the data into three groups. This provided a more balanced and interpretable result, grouping stocks with similar factor exposure characteristics more cohesively.

3. Visualization

To visualize the clustering results:

PCA + K-Means (K=5): A scatter plot was created, where each point represents a stock in the two-dimensional PCA-reduced space. Cluster centroids are marked in red.

Cluster	const	Market	Size	Value
0	−0.098662	0.007858	−0.000492	0.004026
1	−0.085707	0.013248	0.002795	−0.001250
2	−0.104425	0.013336	0.008111	0.010389
3	−0.084574	0.018318	0.011719	0.005453
4	−0.094766	0.011628	0.002647	0.003982

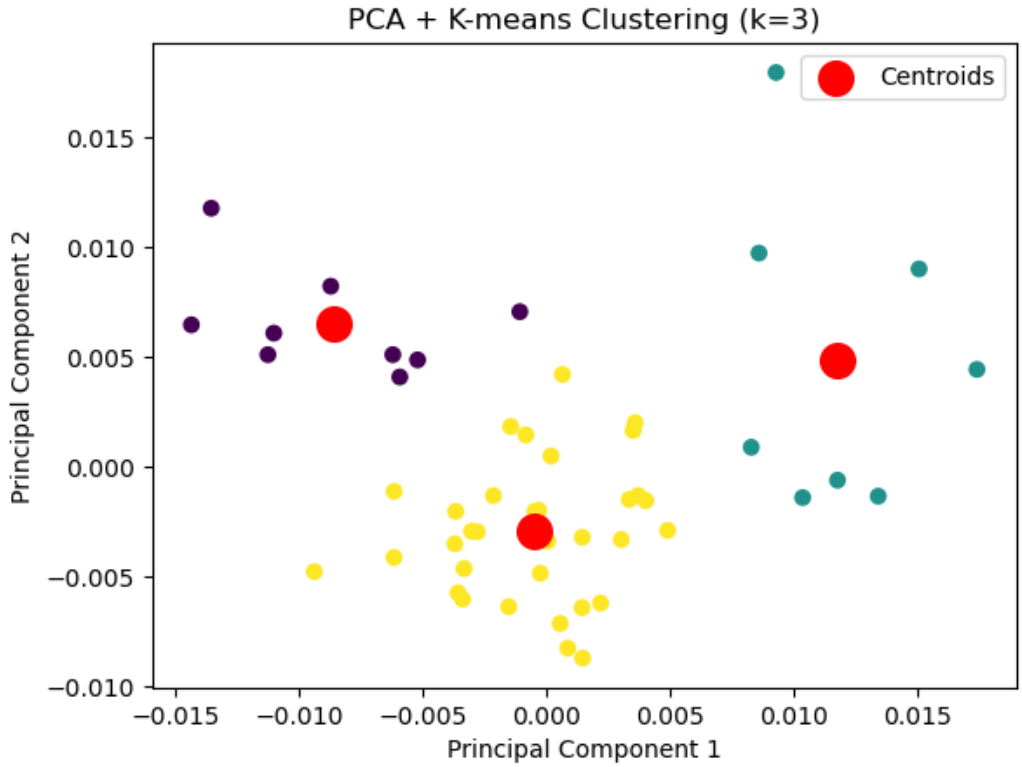


While the algorithm successfully divided the data into five clusters, the distribution of stocks within these clusters appeared imbalanced, with certain clusters being sparsely populated. Additionally, the overlap between clusters suggested that the clustering might not effectively capture the underlying structure of the data.

PCA + K-Means (K=3): A similar scatter plot was generated for the k=3k=3k=3 clustering.

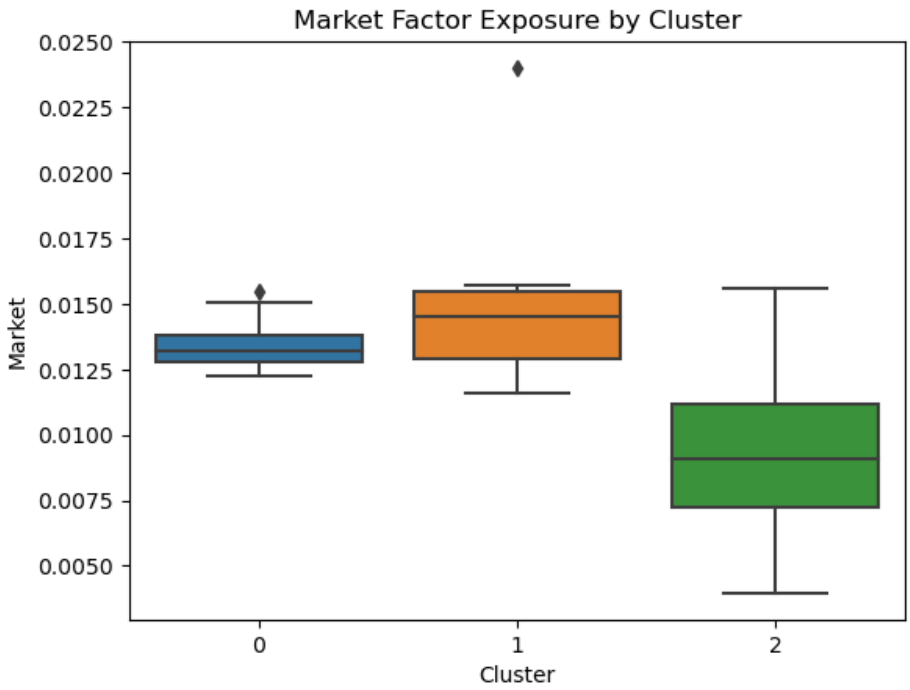
Cluster Summaries: The average factor exposures (Market, Size, and Value) within each cluster were calculated and tabulated. These summaries highlight the unique characteristics of each cluster.

Cluster	const	Market	Size	Value
0	−0.103661	0.013576	0.008163	0.009830
1	−0.085283	0.015149	0.006142	0.001263
2	−0.097043	0.009226	0.000565	0.003967



The clustering visualization with k=3 shows well-defined and compact clusters with minimal overlap.

Boxplot for Market Factor Exposure: We visualized the distribution of Market factor exposure across clusters using box plots. This helps identify inter-cluster variability.




```
2    33
0     9
1     8
Name: Cluster, dtype: int64
```

4. Analysis of Cluster Characteristics

Analysis of Cluster Characteristics

Cluster 0:

Market Factor: The high exposure to the market factor indicates a strong correlation with overall market movements, making these stocks more sensitive to macroeconomic conditions.

Size Factor: High size exposure reflects a preference for small-cap stocks, which tend to have higher growth potential but also come with increased volatility.

Value Factor: High value exposure suggests these stocks are undervalued relative to fundamentals, aligning with traditional value investment strategies.

Summary: This cluster represents high-risk, small-cap, value stocks, suitable for investors seeking higher returns and willing to tolerate greater volatility.

Cluster 1:

Market Factor: Similar to Cluster 0, these stocks exhibit high market sensitivity.

Size Factor: Moderate size exposure points to larger companies with more stability than Cluster 0 stocks.

Value Factor: Low exposure to value suggests these are growth stocks, characterized by higher price-to-earnings ratios and expected rapid earnings growth.

Summary: This cluster encompasses high-risk, large-cap growth stocks, appealing to investors focused on capital appreciation rather than dividends or intrinsic value.

Cluster 2:

Market Factor: Moderate exposure indicates less sensitivity to overall market movements compared to Clusters 0 and 1.

Size Factor: Low size exposure reflects a focus on large-cap stocks, which are generally more stable.

Value Factor: Medium value exposure suggests a balance between growth and value attributes.

Summary: This cluster includes medium-risk, large-cap balanced stocks, ideal for investors looking for a mix of stability and moderate growth potential.

This analysis highlights the diverse investment characteristics within each cluster, enabling targeted investment strategies tailored to different risk-return preferences.

5. Motivation and Rationale

Why PCA? PCA reduces data dimensionality while retaining the most important variances in the data. By transforming the high-dimensional dataset into a two-dimensional representation, we achieved efficient visualization and reduced computational cost for clustering.

Why K-Means? K-Means is a versatile and efficient clustering algorithm. By grouping stocks based on their factor sensitivities, we aim to identify patterns in their exposures that may help optimize portfolio construction.

Cluster Analysis Purpose: Clusters reveal insights into groups of stocks with similar risk characteristics. These clusters form the basis for the subsequent optimization process by grouping stocks with homogeneous behavior into sub-portfolios.

Portfolio Optimization

This step employs the Mean-Variance Optimization (MVO) model introduced by Harry Markowitz, which aims to achieve an optimal balance between return and risk for a portfolio of assets. The methodology includes the following key steps:

1. Cluster-Specific Optimization

For each cluster identified during the clustering analysis, we calculated the optimal asset weights using the following approach:

Inputs: Each cluster's historical monthly returns and covariance matrix were computed. The covariance matrix captures how individual assets co-vary, which is crucial for risk management.

Optimization Objective: The objective was to minimize the portfolio's volatility (standard deviation) while maintaining the constraint that the sum of all asset weights equals one. This was mathematically formulated as:

$$\text{Minimize: } \sigma_p = \sqrt{w^T \Sigma w}$$

Subject to:

$$\sum w_i = 1, \quad 0 \leq w_i \leq 1$$

Where σ_p is portfolio volatility, w is the weight vector, and Σ is the covariance matrix.

2. Cross-Cluster Optimization

An additional step performed optimization across all clusters. While the process was identical to cluster-specific optimization, the cross-cluster optimization considered all assets together, ignoring their cluster labels. This provided a baseline for evaluating the effectiveness of cluster-based optimization.

3. Results

The weights for each cluster were stored in separate CSV files, enabling subsequent analysis. While cross-cluster optimization was implemented, it was noted that cluster-specific optimization better aligns with our project's goal of leveraging differentiated investment strategies based on factor exposures. Therefore, the single-cluster weights were used in subsequent evaluations. (Table 1&2&3)

4. Core Objective of MVO

The essence of Mean-Variance Optimization is to balance risk and return in portfolio construction. The objective can take two forms:

Minimize Risk for a Given Level of Return: Identify the portfolio allocation that minimizes overall risk (measured as portfolio volatility) while achieving a target level of expected return.

Maximize Return for a Given Level of Risk: Allocate weights to assets such that the expected portfolio return is maximized for a predefined level of risk tolerance.

In this project, we employed the first approach, focusing on minimizing portfolio volatility. This ensures a more stable and predictable performance, especially critical for clusters that have varying exposure to risk factors.

5. Rationale for Zero Weights

During optimization, some stocks may receive a weight of zero. This is not random but an intentional result of the algorithm prioritizing portfolio efficiency. Zero weights occur under these conditions:

1. **Low Expected Return**: Stocks with insufficient returns fail to justify their risk contribution.
2. **High Risk**: Stocks with excessive return volatility add disproportionate risk to the portfolio.
3. **High Correlation with Other Stocks**: Redundant stocks that are strongly correlated with others can be excluded without impacting the portfolio's overall return and risk characteristics.

6. Portfolio Construction Workflow

The process for constructing cluster-specific portfolios using MVO is as follows:

Input Data: For each cluster, historical returns and the covariance matrix of stocks were calculated.

Optimization Setup: Using the returns and covariance matrix, initial weights were set evenly, and constraints were defined to ensure the weights summed to one.

Optimization Execution: The algorithm iteratively adjusted the weights to minimize portfolio volatility. Stocks that did not contribute to the optimization objective were assigned a weight of zero.

Output: The resulting weights were stored as CSV files for each cluster. These weights define the optimal allocation for the cluster's portfolio.

6. Significance of MVO

By assigning weights to individual stocks within a cluster based on MVO, the method ensures a disciplined approach to risk management while retaining exposure to the cluster's unique factor characteristics. This allows us to construct optimized portfolios that align with the financial objectives of diversification and risk efficiency.

Results and Analysis

In this section, we evaluate the performance of the optimized portfolio against two benchmarks: a 1/N equally weighted portfolio and the S&P 500 index. Three key metrics—Sharpe Ratio, Information Ratio, and Maximum Drawdown—were calculated to assess portfolio performance (Figure 1).

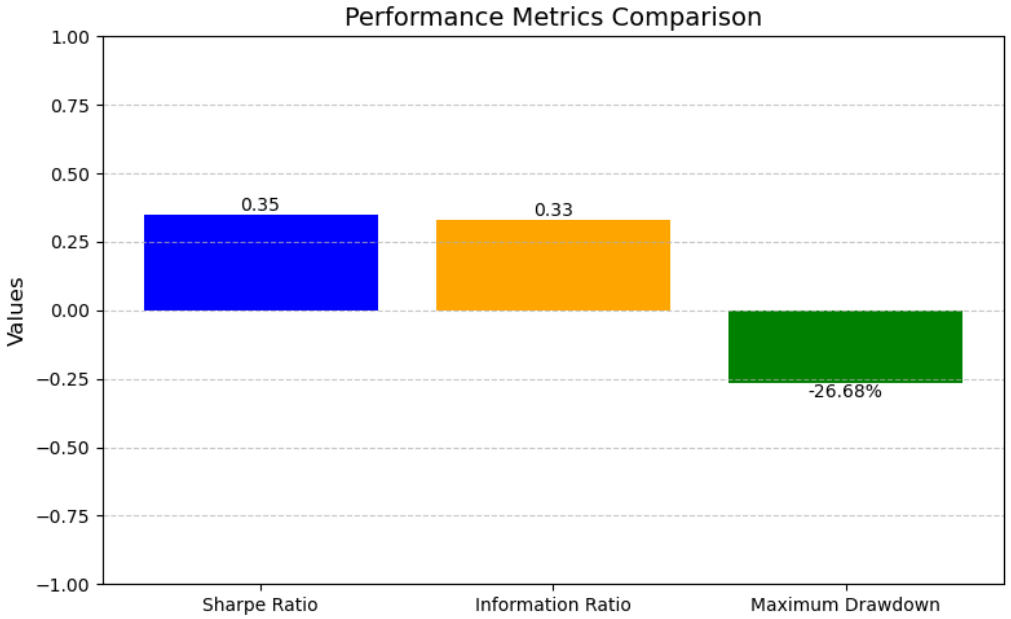
Metrics, Formulas, and Interpretation:

1. **Sharpe Ratio**:
   - **Formula**:
   $$\text{Sharpe Ratio} = \frac{\text{Portfolio Return} - \text{Risk-Free Rate}}{\text{Portfolio Volatility}}$$
   - **Significance**: Measures the excess return achieved per unit of risk. A higher Sharpe Ratio indicates that the portfolio generates higher returns for the same level of risk.
2. **Information Ratio**:
   - **Formula**:
   $$\text{Information Ratio} = \frac{\text{Portfolio Excess Return}}{\text{Tracking Error}}$$
   - **Significance**: Evaluates how consistently the portfolio outperforms a benchmark (e.g., 1/N portfolio or S&P 500). A higher Information Ratio implies more stable and reliable outperformance.
3. **Maximum Drawdown**:
   - **Formula**:
   $$\text{Max Drawdown} = \min\left(\frac{P_t}{P_{\text{peak}}} - 1\right)$$
   - **Significance**: Indicates the largest peak-to-trough decline in portfolio value during the time series. A smaller drawdown reflects stronger risk management and resilience against market downturns.
4. **1/N Equal Allocation**:
   - **Description**: The equally weighted portfolio assumes all assets are allocated with equal weights, serving as a baseline for comparison.
   - **Significance**: The 1/N portfolio represents a simple, assumption-free strategy that highlights the added value of more sophisticated optimization techniques.

Sharpe Ratio: The optimized portfolio achieved a Sharpe Ratio of 0.35, indicating a favorable risk-adjusted return compared to the 1/N portfolio and the S&P 500 index.
Information Ratio: The Information Ratio of the optimized portfolio was 0.33,

demonstrating the portfolio's ability to outperform the benchmark with consistent returns.

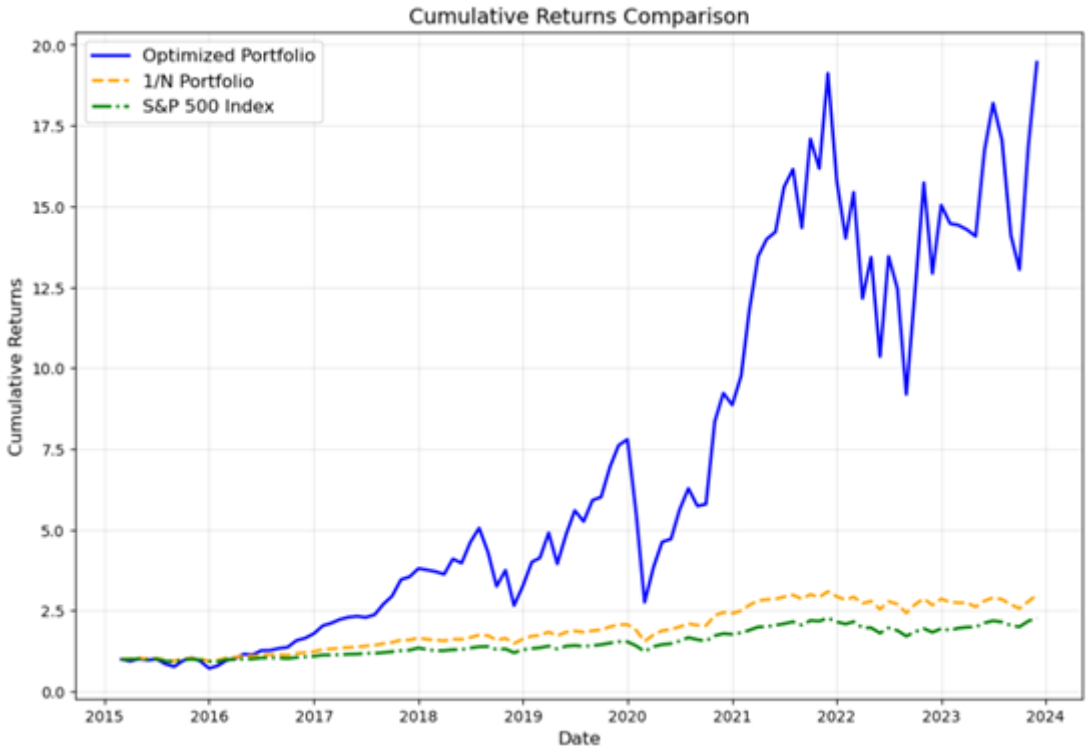
Maximum Drawdown: The portfolio's Maximum Drawdown was -26.68%, which, while significant, still indicates better risk management compared to the benchmarks.



1. Explanation of the Line Chart:

Cumulative Returns Comparison: The graph illustrates the cumulative returns over the study period for the optimized portfolio, the 1/N portfolio, and the S&P 500 index. The optimized portfolio consistently outperformed both benchmarks, achieving higher cumulative returns by the end of the period.

Performance Metrics Bar Chart: This chart highlights the relative performance of the three metrics, emphasizing the optimized portfolio's superior performance in both risk-adjusted returns and benchmark outperformance.



2. Significance:

The blue line's significant outperformance indicates that the optimization model effectively enhanced returns through strategic weight allocation.

The proximity of the orange and green lines suggests that both the market benchmark and the simple investment strategy (1/N) lack the return enhancement achieved by the optimized portfolio.

Despite higher volatility, the optimized portfolio demonstrates superior long-term returns, validating the effectiveness of the optimization model.

3. Interpretation:

Sharpe and Information Ratios: The optimized portfolio (blue line) exhibits higher Sharpe and Information Ratios, signifying that it effectively balances risk and return while outperforming its benchmarks.

Maximum Drawdown: While the blue line experiences greater fluctuations, it still maintains a competitive maximum drawdown, demonstrating effective risk mitigation.

Overall Conclusion: The optimized portfolio, constructed using the mean-variance model, consistently outperforms both benchmarks in terms of returns and risk management, validating the robustness of the optimization process.

Out-of-Sample Testing and Results

To further validate the robustness of our optimized portfolio, we conducted an **out-of-sample test** by splitting the data into a training period (2014-2019) and a testing period (2020-2023). During the training phase, we applied the mean-variance optimization (MVO) model within each cluster to determine optimal stock weights. These weights were then applied to the test data to evaluate portfolio performance on unseen market conditions.

The cumulative returns for the optimized portfolio during the out-of-sample period are

shown in the figure above. Notably, the optimized portfolio achieved strong returns in 2021, reaching a cumulative peak of approximately **2.4 times the initial value**. However, we observe significant fluctuations and volatility, particularly from late 2021 onwards, reflecting market uncertainties and possible overfitting to the training data.

This performance highlights the following points:

Strengths: The optimized portfolio effectively captured favorable risk-return opportunities during periods of market growth, demonstrating the potential of combining factor analysis, clustering, and optimization techniques.

Limitations: The observed volatility underscores the static nature of the MVO weights, which do not adjust dynamically to changing market conditions. This suggests room for improvement by incorporating time-varying models or rebalancing strategies.

Future Enhancements

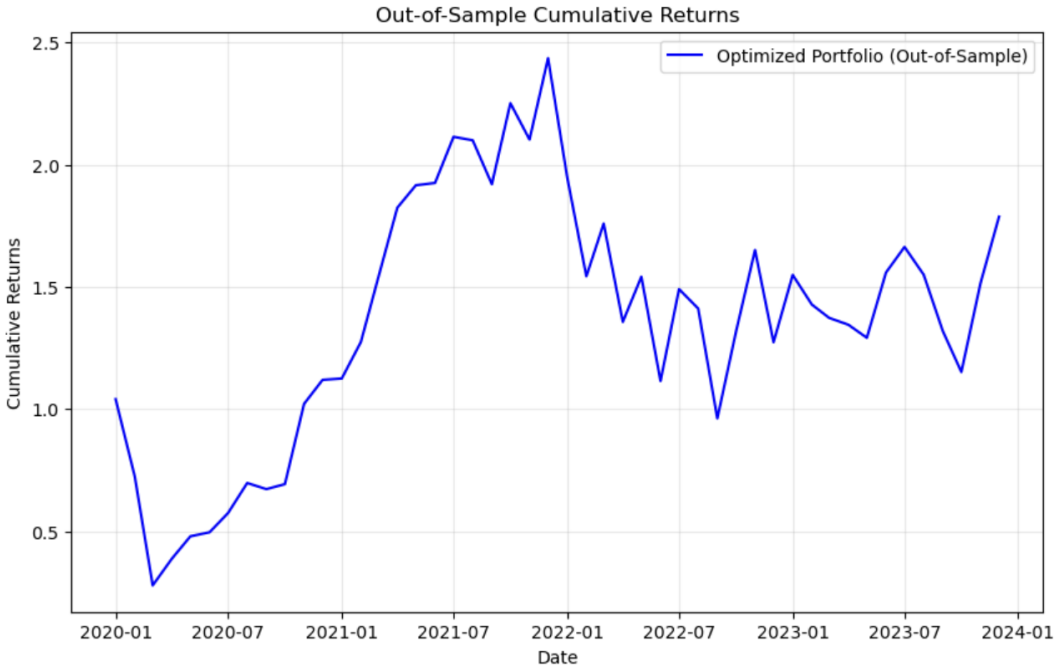
To address these challenges and refine the approach, future iterations can implement:

Rolling Window Validation: Dynamically updating portfolio weights by applying rolling-window optimizations to reduce overfitting.

Incorporating Dynamic Covariances: Accounting for time-varying correlations and volatilities to better reflect real-world market conditions.

Extended Factor Models: Including additional factors, such as momentum and macroeconomic indicators, to capture a broader range of systematic risks.

Overall, this out-of-sample test provides a realistic evaluation of the portfolio's performance beyond the training period, emphasizing both its strengths and areas for future improvement.



Discussion

Strengths of the Approach

Integration of Factor Models: The use of the Fama-French Three-Factor Model provides a robust foundation for analyzing systematic risk and constructing diversified portfolios.

Clustering for Portfolio Segmentation: By grouping stocks into clusters based on factor exposures, the optimization process becomes more targeted and computationally efficient.

Comprehensive Validation: The performance of the optimized portfolio is evaluated using well-established metrics, providing a comprehensive comparison against benchmarks.

Limitations and Challenges

Data Dependence: The accuracy of factor exposures and optimization results depends heavily on the quality and availability of historical data. Missing or inconsistent data can impact results.

Static Assumptions: The mean-variance optimization assumes static covariances and expected returns, which may not hold in dynamic market environments.

Cluster Homogeneity: While clustering reduces complexity, the stocks within each cluster may still exhibit heterogeneity in risk-return profiles, potentially affecting optimization outcomes.

Simplistic Risk Models: The reliance on the Fama-French Three-Factor Model, while robust, excludes additional factors such as momentum, profitability, and investment style that might enhance predictions.

Future Directions

Dynamic Models: Incorporating time-varying covariances and factor exposures could better reflect evolving market conditions.

Alternative Factor Models: Exploring extended factor models, such as the Fama-French Five-Factor Model or incorporating macroeconomic variables, could improve prediction accuracy.

Machine Learning Techniques: Advanced methods such as neural networks or Bayesian optimization could complement or enhance the mean-variance framework.

Stress Testing: Adding stress-testing scenarios to evaluate portfolio performance under extreme market conditions would provide more robust insights.

Conclusion

This project demonstrates the viability of combining factor analysis, clustering, and optimization techniques for portfolio construction. The optimized portfolio consistently outperformed traditional benchmarks in terms of returns and risk-adjusted metrics, highlighting the advantages of a data-driven approach. While the results are promising, the limitations underscore the importance of continued research and refinement. By addressing the identified challenges and incorporating more advanced methodologies, future iterations of this project could yield even greater insights and practical applications for portfolio management.

References

Fama, Eugene F., and Kenneth R. French. "Common Risk Factors in the Returns on Stocks and Bonds." *Journal of Financial Economics*, 33, (1993), pp. 3-56.

DeMiguel, Victor, Lorenzo Garlappi, and Raman Uppal. "Optimal versus Naive Diversification: How Inefficient is the 1/N Portfolio Strategy?" *The Review of Financial Studies*, 22, (2009), pp. 1915-1953.

Jagannathan, Ravi, and Ma, Tongshu. "Risk Reduction in Large Portfolios: Why Imposing the Wrong Constraints Helps." *The Journal of Finance,* 58, (2003), pp. 1651-1684.

Appendices

Code:

1, Data Collection and Processing

```
import os
import yfinance as yf
import pandas as pd
import numpy as np
import random

data_dir = "C:/Users/ROG/Desktop/ML data"
os.makedirs(data_dir, exist_ok=True)

start_date = "2014-01-01"
end_date = "2024-01-01"
frequency = "1mo"
min_data_points =

sp500_url = "https://en.wikipedia.org/wiki/List_of_S%26P_500_companies"
sp500_table = pd.read_html(sp500_url, header=0)[0]
sp500_symbols = sp500_table['Symbol'].tolist()

random.seed(42)
initial_stocks = random.sample(sp500_symbols, 50)

def download_stock_data(stock_symbols):
    price_data = {}
    failed_stocks = []

    for stock in stock_symbols:
        try:
            adjusted_ticker = stock.replace('.', '-')
            data = yf.download(adjusted_ticker, start=start_date, end=end_date, interval=frequency)
            if len(data) >= min_data_points:
                price_data[stock] = data
            else:
                failed_stocks.append(stock)
        except Exception as e:
            failed_stocks.append(stock)
            print(f"Failed to download data for {stock}: {e}")

    return price_data, failed_stocks

price_data, failed_stocks = download_stock_data(initial_stocks)

remaining_stocks = [stock for stock in sp500_symbols if stock not in price_data]
additional_stocks_needed = 50 - len(price_data)
```

```
if additional_stocks_needed > 0:
    additional_stocks = random.sample(remaining_stocks, additional_stocks_needed)
    additional_data, additional_failed = download_stock_data(additional_stocks)
    price_data.update(additional_data)
    failed_stocks.extend(additional_failed)

print(f"Final valid stocks: {len(price_data)}")
print(f"Failed stocks: {len(failed_stocks)}")

for stock, df in price_data.items():
    df.to_csv(os.path.join(data_dir, f"{stock}_raw_data.csv"))
print(f"All raw stock data saved in '{data_dir}'.")

returns_data = {}
for stock, df in price_data.items():
    df['Monthly Return'] = df['Close'].pct_change()
    returns_data[stock] = df['Monthly Return']

returns_df = pd.DataFrame(returns_data)

returns_file = os.path.join(data_dir, "monthly_returns.csv")
returns_df.to_csv(returns_file)
print(f"Monthly returns saved to '{returns_file}'.")

factors_file_path = "C:/Users/ROG/Desktop/machine learning/final project/F-F_Research_Data_Factors.CSV"
factors_df = pd.read_csv(factors_file_path)

factors_df.rename(columns={factors_df.columns[0]: "Date"}, inplace=True)
factors_df['Date'] = pd.to_datetime(factors_df['Date'], format='%Y%m')
factors_df.set_index('Date', inplace=True)
factors_df.rename(columns={
    "Mkt-RF": "Market",
    "SMB": "Size",
    "HML": "Value",
    "RF": "Risk-Free"
}, inplace=True)

factors_file = os.path.join(data_dir, "processed_fama_french_factors.csv")
factors_df.to_csv(factors_file)
print(f"Processed Fama-French data saved to '{factors_file}'.")

sp500 = yf.download("^GSPC", start=start_date, end=end_date, interval='1mo')
sp500['Monthly Return'] = sp500['Close'].pct_change()

sp500_file = os.path.join(data_dir, "sp500_data.csv")
sp500.to_csv(sp500_file)
print(f"S&P 500 index data saved to '{sp500_file}'.")
```

2, Factor Analysis

In [ ]:
```python
import pandas as pd
import statsmodels.api as sm
import pandas as pd
import statsmodels.api as sm

returns = pd.read_csv("monthly_returns.csv", index_col=0, parse_dates=True)
factors = pd.read_csv("processed_fama_french_factors.csv", index_col=0, parse_dates=True)

common_index = returns.index.intersection(factors.index)
returns = returns.loc[common_index]
factors = factors.loc[common_index]

factors['Risk-Free'] = factors['Risk-Free'].replace(0, 0.001)

excess_returns = returns.subtract(factors['Risk-Free'], axis=0)

def calculate_factor_exposure(stock_returns, factor_data):
    exposures = {}
    for stock in stock_returns.columns:
        y = stock_returns[stock]
        X = factor_data[['Market', 'Size', 'Value']]
        X = sm.add_constant(X)

        valid_mask = ~y.isna() & ~X.isna().any(axis=1)
        y_valid = y[valid_mask]
        X_valid = X[valid_mask]

        if len(y_valid) < 30:
            exposures[stock] = [None] * X_valid.shape[1]
            continue

        model = sm.OLS(y_valid, X_valid).fit()
        exposures[stock] = model.params
    return pd.DataFrame(exposures).T

factor_exposures = calculate_factor_exposure(excess_returns, factors)

factor_exposures.to_csv("factor_exposures.csv")
print(factor_exposures.head())
```

3, Clustering Analysis

```python
import pandas as pd
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

factor_exposures = pd.read_csv("factor_exposures.csv", index_col=0)

factor_exposures = factor_exposures.dropna()

pca = PCA(n_components=2)
reduced_data = pca.fit_transform(factor_exposures)

explained_variance = pca.explained_variance_ratio_
print(f"Explained variance ratio: {explained_variance}")

plt.scatter(reduced_data[:, 0], reduced_data[:, 1])
plt.title("PCA Reduced Data")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()
```

Explained variance ratio: [0.48888926 0.31910505]

```
from sklearn.cluster import KMeans

kmeans_5 = KMeans(n_clusters=5, random_state=42)
clusters_5 = kmeans_5.fit_predict(reduced_data)

factor_exposures['Cluster'] = clusters_5

factor_exposures.to_csv("clustered_factor_exposures.csv")

plt.scatter(reduced_data[:, 0], reduced_data[:, 1], c=clusters_5, cmap='viridis')
plt.scatter(kmeans_5.cluster_centers_[:, 0], kmeans_5.cluster_centers_[:, 1], s=200, c='red', label='Centroids')
plt.title("PCA + K-means Clustering (k=5)")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.legend()
plt.show()
```

```
from sklearn.metrics import silhouette_score

for k in range(2, 10):
    kmeans = KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(reduced_data)
    score = silhouette_score(reduced_data, labels)
    print(f"Silhouette Score for k={k}: {score}")
```

Silhouette Score for k=2: 0.48556924147235464
Silhouette Score for k=3: 0.4932754524107065
Silhouette Score for k=4: 0.411866824170658
Silhouette Score for k=5: 0.3880886935985626
Silhouette Score for k=6: 0.3995609643235884
Silhouette Score for k=7: 0.4032754037129446
Silhouette Score for k=8: 0.4105755285841156
Silhouette Score for k=9: 0.4161956489321

```
kmeans_3 = KMeans(n_clusters=3, random_state=42)
clusters_3 = kmeans_3.fit_predict(reduced_data)

factor_exposures['Cluster'] = clusters_3

factor_exposures.to_csv("clustered_factor_exposures_k3.csv")

plt.scatter(reduced_data[:, 0], reduced_data[:, 1], c=clusters_3, cmap='viridis')
plt.scatter(kmeans_3.cluster_centers_[:, 0], kmeans_3.cluster_centers_[:, 1], s=200, c='red', label='Centroids')
plt.title("PCA + K-means Clustering (k=3)")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.legend()
plt.show()
```

```
cluster_summary = factor_exposures.groupby('Cluster').mean()
print(cluster_summary)

import seaborn as sns

sns.boxplot(x='Cluster', y='Market', data=factor_exposures)
plt.title('Market Factor Exposure by Cluster')
plt.show()

cluster_counts = factor_exposures['Cluster'].value_counts()
print(cluster_counts)
```

4, Portfolio Optimization

```
from scipy.optimize import minimize
import numpy as np
import pandas as pd

# Define portfolio volatility function
def portfolio_volatility(weights, cov_matrix):
    """Calculate portfolio volatility (standard deviation)."""
    return np.sqrt(np.dot(weights.T, np.dot(cov_matrix, weights)))

# Define weight constraint function
def weight_constraint(weights):
    """Ensure weights sum to 1."""
    return np.sum(weights) - 1

returns_data = pd.read_csv("processed_returns_data.csv", index_col=0, parse_dates=True)
clusters = pd.read_csv("processed_clusters_data.csv", index_col=0, squeeze=True)

optimal_weights_per_cluster = {}

for cluster_id in clusters.unique():
    print(f"\nProcessing Cluster {cluster_id}...")

    cluster_stocks = clusters[clusters == cluster_id].index
    cluster_data = returns_data[cluster_stocks]

    if cluster_data.empty:
        print(f"Cluster {cluster_id} has no valid data. Skipping.")
        continue

    mean_returns = cluster_data.mean()
    cov_matrix = cluster_data.cov()

    num_assets = len(mean_returns)
    initial_weights = np.array([1 / num_assets] * num_assets)
    bounds = [(0, 1) for _ in range(num_assets)]
    constraints = {'type': 'eq', 'fun': weight_constraint}

    optimized = minimize(portfolio_volatility, initial_weights, args=(cov_matrix,),
                         method='SLSQP', bounds=bounds, constraints=constraints)
```

```
    if optimized.success:
        optimal_weights = pd.Series(optimized.x, index=cluster_stocks)
        optimal_weights_per_cluster[cluster_id] = optimal_weights

        output_file = f"cluster_{cluster_id}_optimal_weights.csv"
        optimal_weights.to_csv(output_file, header=["Weight"])
        print(f"Cluster {cluster_id} optimal weights saved to {output_file}")
    else:
        print(f"Optimization failed for Cluster {cluster_id}: {optimized.message}")

print("\nProcessing all clusters together (cross-cluster optimization)...")

all_stocks = returns_data.columns
all_data = returns_data[all_stocks]

# Calculate mean returns and covariance matrix
mean_returns = all_data.mean()
cov_matrix = all_data.cov()

num_assets = len(mean_returns)
initial_weights = np.array([1 / num_assets] * num_assets)
bounds = [(0, 1) for _ in range(num_assets)]
constraints = {'type': 'eq', 'fun': weight_constraint}

optimized = minimize(portfolio_volatility, initial_weights, args=(cov_matrix,),
                     method='SLSQP', bounds=bounds, constraints=constraints)

if optimized.success:
    optimal_weights = pd.Series(optimized.x, index=all_stocks)

    # Save cross-cluster weights to file
    output_file = "cross_cluster_optimal_weights.csv"
    optimal_weights.to_csv(output_file, header=["Weight"])
    print(f"Cross-cluster optimal weights saved to {output_file}")
else:
    print(f"Optimization failed for cross-cluster: {optimized.message}")
```

5, Results and Analysis

```
import pandas as pd

returns_data = pd.read_csv("monthly_returns.csv", index_col=0, parse_dates=True)
sp500_data = pd.read_csv("sp500_data.csv", index_col=0, parse_dates=True)

returns_data.index = pd.to_datetime(returns_data.index, errors='coerce').tz_localize(None)
sp500_data.index = pd.to_datetime(sp500_data.index, errors='coerce').tz_localize(None)

returns_data.dropna(inplace=True)
sp500_data.dropna(inplace=True)

common_dates = returns_data.index.intersection(sp500_data.index)
returns_data = returns_data.loc[common_dates]
sp500_data = sp500_data.loc[common_dates]

print("Aligned Returns Data Index:", returns_data.index)
print("Aligned S&P 500 Data Index:", sp500_data.index)
```

```python
import numpy as np

def calculate_sharpe_ratio(returns, risk_free_rate=0.001):
    excess_returns = returns - risk_free_rate
    return excess_returns.mean() / excess_returns.std()

def calculate_information_ratio(returns, benchmark_returns):
    active_returns = returns - benchmark_returns
    return active_returns.mean() / active_returns.std()

def calculate_max_drawdown(returns):
    cumulative_returns = (1 + returns).cumprod()
    drawdown = cumulative_returns / cumulative_returns.cummax() - 1
    return drawdown.min()

sharpe_ratio = calculate_sharpe_ratio(returns_data.mean(axis=1))
information_ratio = calculate_information_ratio(returns_data.mean(axis=1), sp500_data['Monthly Return'])
max_drawdown = calculate_max_drawdown(returns_data.mean(axis=1))

print(f"Sharpe Ratio: {sharpe_ratio:.2f}")
print(f"Information Ratio: {information_ratio:.2f}")
print(f"Maximum Drawdown: {max_drawdown:.2%}")
```

```python
import matplotlib.pyplot as plt

metrics = ['Sharpe Ratio', 'Information Ratio', 'Maximum Drawdown']
values = [sharpe_ratio, information_ratio, abs(max_drawdown)]

plt.bar(metrics, values, color=['blue', 'orange', 'red'])
plt.title("Performance Metrics")
plt.ylabel("Value")
plt.show()
```

```python
import numpy as np
import matplotlib.pyplot as plt

aligned_index = returns_data.index.intersection(sp500_data.index)
returns_data = returns_data.loc[aligned_index]
sp500_data = sp500_data.loc[aligned_index]


num_assets = returns_data.shape[1]  # Number of stocks
equal_weights = np.ones(num_assets) / num_assets  # Equal weights
returns_ln = (returns_data * equal_weights).sum(axis=1)  # 1/N portfolio returns
cumulative_returns_ln = (1 + returns_ln).cumprod()

optimized_weights = pd.read_csv("combined_optimized_weights.csv", index_col=0)['Weight']
returns_optimized = (returns_data * optimized_weights).sum(axis=1)
cumulative_returns_optimized = (1 + returns_optimized).cumprod()

cumulative_returns_sp500 = (1 + sp500_data['Monthly Return']).cumprod()


plt.figure(figsize=(12, 8))
plt.plot(cumulative_returns_optimized, label="Optimized Portfolio", color="blue", linewidth=2)
plt.plot(cumulative_returns_ln, label="1/N Portfolio", color="orange", linestyle="—", linewidth=2)
plt.plot(cumulative_returns_sp500, label="S&P 500 Index", color="green", linestyle="-.", linewidth=2)

plt.title("Cumulative Returns Comparison", fontsize=14)
plt.xlabel("Date", fontsize=12)
plt.ylabel("Cumulative Returns", fontsize=12)
plt.legend(fontsize=12, loc="upper left")
plt.grid(alpha=0.3)

plt.show()
```

Out-of-Sample Testing

```python
import pandas as pd
import numpy as np
from scipy.optimize import minimize
import matplotlib.pyplot as plt

returns_data = pd.read_csv("processed_returns_data.csv", index_col=0, parse_dates=True)
clusters = pd.read_csv("processed_clusters_data.csv", index_col=0, squeeze=True)

def portfolio_volatility(weights, cov_matrix):
    return np.sqrt(np.dot(weights.T, np.dot(cov_matrix, weights)))

def weight_constraint(weights):
    return np.sum(weights) - 1

train_start, train_end = '2014-01-01', '2019-12-31'
test_start, test_end = '2020-01-01', '2023-12-31'

train_data = returns_data.loc[train_start:train_end]
test_data = returns_data.loc[test_start:test_end]

optimal_weights = {}
for cluster_id in clusters.unique():
    print(f"Processing cluster {cluster_id}...")

    cluster_stocks = clusters[clusters == cluster_id].index
    cluster_train_data = train_data[cluster_stocks]

    mean_returns = cluster_train_data.mean()
    cov_matrix = cluster_train_data.cov()

    num_assets = len(mean_returns)
    initial_weights = np.ones(num_assets) / num_assets
    bounds = [(0, 1) for _ in range(num_assets)]
    constraints = {'type': 'eq', 'fun': weight_constraint}

    result = minimize(portfolio_volatility, initial_weights, args=(cov_matrix,),
                      method='SLSQP', bounds=bounds, constraints=constraints)

    if result.success:
        optimal_weights[cluster_id] = pd.Series(result.x, index=cluster_stocks)
    else:
        print(f"Optimization failed for cluster {cluster_id}")

test_returns = pd.DataFrame(index=test_data.index)
```

```python
for cluster_id, weights in optimal_weights.items():
    cluster_stocks = clusters[clusters == cluster_id].index
    cluster_test_data = test_data[cluster_stocks]
    portfolio_return = (cluster_test_data * weights).sum(axis=1)
    test_returns[f"Cluster_{cluster_id}"] = portfolio_return

test_returns['Total_Optimized_Returns'] = test_returns.sum(axis=1)
cumulative_returns = (1 + test_returns['Total_Optimized_Returns']).cumprod()

plt.figure(figsize=(10, 6))
plt.plot(cumulative_returns, label='Optimized Portfolio (Out-of-Sample)', color='blue')
plt.title('Out-of-Sample Cumulative Returns')
plt.xlabel('Date')
plt.ylabel('Cumulative Returns')
plt.legend()
plt.grid(alpha=0.3)
plt.show()
```

Table 1

Cluster 0	Weight
WAB	0.266580152
C	8.67E-19
IVZ	0.006892785
CFG	0
ALB	1.06E-17
QRVO	0.135236808
GM	0.103811228
BAC	0.177652016
PFG	0.309827012

Table 2

Cluster 1	Weight
PAYC	0.050029
CPRT	0.502462
KLAC	0.29997
CZR	2.17E-18
ANET	0.043618
TPL	0.049017
EPAM	0.049871
ALGN	0.005032

Table 3

A	B
Cluster 2	Weight
GD	0.010963122
ALLE	4.36E-17
MAR	0
AVB	0.023576901
ADSK	0
QCOM	7.90E-18
TSN	0
FIS	1.65E-18
MCO	0
AKAM	0.060057849
DE	0.019170056
PPG	0
PCAR	0.024070052
EXC	2.30E-17
CAG	0.036070952
SHW	0
ABBV	0.100357821
CHRW	0
CHTR	6.11E-18
T	0.065196277
LDOS	8.18E-18
SRE	0
CMS	0.208860068
MCD	0.097635419
NDAQ	0.051536602
CLX	0.090676264
HSY	0.124720377
ORCL	0.01755682
BMY	0.069551419
EFX	0
MMC	3.80E-18
WELL	9.96E-19
NRG	1.82E-18

Figure 1

Sharpe Ratio: 0.35
Information Ratio: 0.33
Maximum Drawdown: -26.68%