InvestoQuest

A Comprehensive Study of Portfolio Management

Finance and Analytics Club, IIT Kanpur

Mentors:

- Harshvardhan Gaur
- Ayush Omer
- Nitin Maheshwari
- Atit Kumar Satsangi

Contents

Portfolio Management and Its Importance

1 What is Portfolio Management?

Portfolio Management is the art and science of managing a group of investments to achieve financial goals while balancing risk and return. It involves strategic decision-making on asset allocation, diversification, and risk control.

Key Concepts:

- **Diversification:** Spreading risk across various assets to reduce the impact of any single investment's poor performance.
- **Risk Management:** Tailoring investment strategies to suit an investor's risk appetite and financial profile.
- **Performance Monitoring:** Regular evaluation of investment performance to ensure alignment with goals.
- **Strategic Allocation:** Constructing a portfolio with the right mix of asset classes (equity, debt, commodities, etc.).

2 Types of Portfolio Management

1. Active Management

Frequent buying and selling of assets based on market timing, alpha signals, forecasts, or analysis such as:

- Technical indicators (momentum, RSI, moving averages)
- Fundamental data (P/E ratio, revenue growth)
- Machine Learning models trained on financial data

2. Passive Management

Aims to replicate the performance of a specific market index (e.g., S&P 500) using ETFs or index funds. Tools like Python are used to allocate weights and track index performance.

3. Quantitative Management

This strategy uses statistical models, optimization techniques, and algorithmic trading systems. Techniques include:

- Factor investing
- Risk parity
- Mean-variance optimization

Python scripts are employed to backtest and execute these models.

4. Discretionary Management

A portfolio manager makes all investment decisions on behalf of the investor based on agreed-upon objectives and risk tolerance.

5. Non-Discretionary Management

The advisor provides investment recommendations, but the investor retains decision-making authority.

3 Importance of Portfolio Management

1. Reduction of Risk

By allocating investments across uncorrelated assets, overall risk is reduced. Portfolio managers also control exposure by adjusting the proportion of capital assigned to riskier assets.

2. Dealing with Adverse Conditions

During market downturns or unfavorable conditions, funds can be withdrawn from poorly performing assets and reallocated to safer or more promising financial instruments.

3. Tax Planning

Portfolio management helps analyze an investor's financial situation from a tax-efficiency perspective. Investments are structured to minimize tax liability and maximize post-tax returns.

4. Efficient Allocation of Funds

Capital is allocated across various investment opportunities by evaluating market returns, investor preferences, and risk appetite to optimize performance.

4 Conclusion

Portfolio management plays a vital role in ensuring that financial goals are met while minimizing risks and maximizing returns. It empowers investors through diversification, data-driven strategies, and continuous performance evaluation, leading to better financial outcomes.

Integrating Fundamental and Technical Analysis in Portfolio Strategy

5 Fundamental Analysis

Fundamental Analysis (FA) focuses on evaluating a company's intrinsic value through its financial health, industry position, and qualitative factors like management efficiency. It is typically used by long-term investors who aim to identify undervalued opportunities in the market.

Key Components

- **Financial Strength**: Analyzing metrics such as earnings growth, revenue trends, debt levels, and profitability ratios.
- **Management Quality**: Assessing the company leadership's track record, transparency, and governance.
- **Valuation Tools**: Use of P/E ratio, Price-to-Book ratio, or discounted cash flow (DCF) models to estimate fair value.

Purpose

To identify sound businesses trading below their intrinsic value and hold them for long-term appreciation.

6 Technical Analysis

Technical Analysis (TA) evaluates securities by studying historical price patterns and market-generated data like volume. It assumes that all fundamental factors are already reflected in the price and focuses on market behavior and trends.

Key Techniques

- **Chart Patterns**: Identifying formations such as head-and-shoulders, double tops, or triangles to forecast direction.
- **Indicators**: Using tools like RSI, MACD, moving averages, Bollinger Bands to assess trend strength or momentum.
- **Volume Analysis**: Confirming trends and breakouts with corresponding changes in trading volume.

Purpose

To determine optimal buy/sell points and manage short- to medium-term market movements effectively.

7 Combining Fundamental and Technical Analysis

While FA ensures the selection of quality assets, TA aids in optimizing the timing of trades. When used together, they form a powerful decision-making framework that balances long-term value with

short-term opportunity.

How They Complement Each Other

Role of FA (What to Buy):

- Select companies with stable earnings, high return on equity (RoE), and sound balance sheets.
- Focus on industries poised for structural growth.
- Diversify across fundamentally strong sectors and geographies.

Role of TA (When to Buy or Sell):

- Use momentum indicators to avoid entering during overbought phases.
- Apply support/resistance levels and moving averages for precise timing.
- Adjust allocation using volatility-based signals and volume confirmation.

Benefits of Integration

- Enhances portfolio returns while reducing risk.
- Helps avoid emotional or impulse-driven trading.
- Allows for both conviction in holdings and agility in execution.

Conclusion

Blending FA and TA allows investors to build resilient portfolios grounded in economic value and guided by market behavior. This strategic combination supports better timing, risk control, and performance consistency over time.

Key Performance Metrics in Portfolio Evaluation

Introduction

Quantitative metrics are essential in evaluating investment strategies, balancing risk and return, and guiding portfolio decisions. The following sections group the metrics based on the aspects of portfolio performance they measure.

1. Risk Metrics

Volatility

Definition: Standard deviation of returns, indicating variability.
Formula: $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(R_i - \bar{R})^2}$
Importance: Higher volatility implies greater uncertainty.

Drawdowns

Max Drawdown: Largest peak-to-trough loss.
$MaxDD = \frac{\text{Peak}-\text{Trough}}{\text{Peak}}$
Average Drawdown: Average of all drawdowns.
Importance: Evaluates loss severity and recovery requirements.

Value at Risk (VaR)

Definition: Maximum loss at a confidence level.
Formula: $VaR_\alpha = \inf\{x|P(R \leq x) \geq \alpha\}$
Importance: Quantifies loss potential.

Conditional Value at Risk (CVaR)

Definition: Average loss beyond VaR.
Importance: Captures tail risks better than VaR.

Time to Recovery

Definition: Time taken to recover from drawdown.
Importance: Reflects recovery speed.

2. Risk-Adjusted Performance Metrics

Sharpe Ratio

Definition: Excess return per unit of total risk.
Formula: $Sharpe = \frac{R_p - R_f}{\sigma_p}$
Importance: Allows risk-adjusted return comparisons.

Sortino Ratio

Definition: Excess return per unit of downside risk.
Formula: $Sortino = \frac{R_p - R_f}{\sigma_d}$
Importance: Penalizes harmful volatility only.

Profit Factor

Definition: Ratio of gross profits to losses.
Formula: $Profit\ Factor = \frac{\text{Total Profit}}{\text{Total Loss}}$
Importance: > 1 indicates profitability.

Recovery Factor

Definition: Return relative to max drawdown.
Formula: $Recovery\ Factor = \frac{\text{Net Profit}}{MaxDD}$
Importance: Measures resilience.

Alpha

Definition: Excess return unexplained by Beta.
Formula: $Alpha = R_p - [R_f + \beta(R_m - R_f)]$
Importance: Indicates strategy edge.

3. Trading Quality Metrics

Win Rate

Definition: Proportion of winning trades.
Formula: $Win\ Rate = \frac{\text{Winning Trades}}{\text{Total Trades}}$
Importance: Indicates consistency.

4. Market Sensitivity Metrics

Beta

Definition: Sensitivity to market movements.
Formula: $Beta = \frac{\text{Cov}(R_p, R_m)}{\text{Var}(R_m)}$
Importance: Beta > 1 implies more volatility than market.

Conclusion

These metrics together offer a comprehensive evaluation of portfolio performance across risk, return, efficiency, consistency, and market sensitivity dimensions.

8 Modern Portfolio Theory: A Mathematical and Computational Overview

8.1 Introduction

Modern Portfolio Theory (MPT), introduced by Harry Markowitz in 1952, is a quantitative framework that helps investors construct portfolios to optimize expected return based on a given level of market risk. It emphasizes the importance of diversification, showing that a well-diversified portfolio can reduce risk without sacrificing returns.

8.2 Advantages of MPT

- Encourages diversification to reduce unsystematic risk.
- Provides a clear mathematical framework for portfolio optimization.
- Helps in identifying the efficient frontier and optimal portfolios.
- Useful for asset allocation and strategic decision-making.

8.3 Usage of MPT

- **Efficient Frontier Construction:** Helps identify portfolios with the best return-to-risk ratio.
- **Portfolio Optimization:** Allocating weights to minimize risk or maximize Sharpe ratio.
- **Foundation for CAPM (Capital Asset Pricing Model):** CAPM builds on MPT to price individual assets.

8.4 Assumptions of MPT

- **Investors are rational and risk-averse**: They seek to maximize returns for a given risk
- **Markets are efficient**: Prices fully reflect all available information.
- Returns are normally distributed.
- **Markets are efficient:** Prices fully reflect all available information.

8.5 Mathematical Foundation

8.5.1 Expected Return of a Portfolio

Let R_i be the expected return of asset i, and w_i its weight in the portfolio.

$$E[R_p] = \sum_{i=1}^{n} w_i R_i$$

8.5.2 Portfolio Variance

Let Σ be the covariance matrix of asset returns.

$$\sigma_p^2 = \mathbf{w}^T \Sigma \mathbf{w}$$

where $\mathbf{w} = [w_1, w_2, \ldots, w_n]^T$ is the weight vector.

8.5.3 Efficient Frontier

An efficient frontier represents the set of portfolios offering the highest expected return for a defined level of risk.

8.5.4 Optimization Problem

To find the minimum variance portfolio for a given return:

$$\min_{\mathbf{w}} \quad \mathbf{w}^T \Sigma \mathbf{w} \quad \text{subject to} \quad \mathbf{w}^T \mathbf{1} = 1, \quad \mathbf{w}^T \mathbf{R} = R_t$$

where $\mathbf{1}$ is a vector of ones, and R_t is the target return.

Solving these equations gives the optimal weights.

8.6 Special Portfolios

8.6.1 Global Minimum Variance Portfolio (GMVP)

The portfolio with the lowest possible variance:

$$\min_{\mathbf{w}} \quad \mathbf{w}^T \Sigma \mathbf{w} \quad \text{subject to} \quad \mathbf{w}^T \mathbf{1} = 1$$

8.6.2 Tangency Portfolio (Maximum Sharpe Ratio)

Assuming a risk-free rate R_f:

$$\max_{\mathbf{w}} \quad \frac{\mathbf{w}^T (\mathbf{R} - R_f \mathbf{1})}{\sqrt{\mathbf{w}^T \Sigma \mathbf{w}}} \quad \text{subject to} \quad \mathbf{w}^T \mathbf{1} = 1$$

8.7 Our Implementation in Python

The following Python code demonstrates how to visualize the Efficient Frontier and identify the tangency portfolio (maximum Sharpe ratio) when the risk-free rate is assumed to be zero. The portfolios are first sorted by volatility, and the upper envelope of efficient portfolios is constructed by filtering those with increasing returns. Sharpe ratios are calculated from the origin (representing the Capital Market Line), and the portfolio with the maximum Sharpe ratio is identified as the tangent point. The plot displays the full set of portfolios, the efficient frontier, the tangent line from the origin, and highlights the tangency portfolio on the graph.

```
# Sort portfolios by volatility
sorted_indices = np.argsort(portfolio_volatility)
sorted_volatility = portfolio_volatility[sorted_indices]
sorted_returns = expected_returns[sorted_indices]
```

```
# Construct upper envelope (efficient frontier)
efficient_vol = []
efficient_ret = []
max_return = -np.inf

for vol, ret in zip(sorted_volatility, sorted_returns):
    if ret > max_return:
        efficient_vol.append(vol)
        efficient_ret.append(ret)
        max_return = ret

efficient_vol = np.array(efficient_vol)
efficient_ret = np.array(efficient_ret)

# Compute Sharpe ratios from origin (risk-free rate = 0)
origin_sharpe_ratios = efficient_ret / efficient_vol
max_sharpe_idx = np.argmax(origin_sharpe_ratios)

# Coordinates of the tangent point
tangent_vol = efficient_vol[max_sharpe_idx]
tangent_ret = efficient_ret[max_sharpe_idx]

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(efficient_vol, efficient_ret, color='blue', linewidth=2, label='Efficient
    Frontier')
plt.scatter(portfolio_volatility, expected_returns, c=sharpe_ratios, cmap='viridis
    ', alpha=0.5)
plt.colorbar(label='Sharpe Ratio')

# Draw the tangent line from origin
plt.plot([0, tangent_vol], [0, tangent_ret], color='red', linestyle='--',
    linewidth=2, label='Capital Allocation Line (Tangent)')

# Highlight tangent point
plt.scatter([tangent_vol], [tangent_ret], color='red', zorder=5)
plt.text(tangent_vol, tangent_ret, f"  Max Sharpe\n  ({tangent_vol:.2f}, {
    tangent_ret:.2f})", verticalalignment='bottom')

# Labels and title
plt.xlabel('Volatility (Risk)')
plt.ylabel('Expected Return')
plt.title('Efficient Frontier with Tangent from Origin')
plt.legend()
plt.grid(True)
plt.show()
```

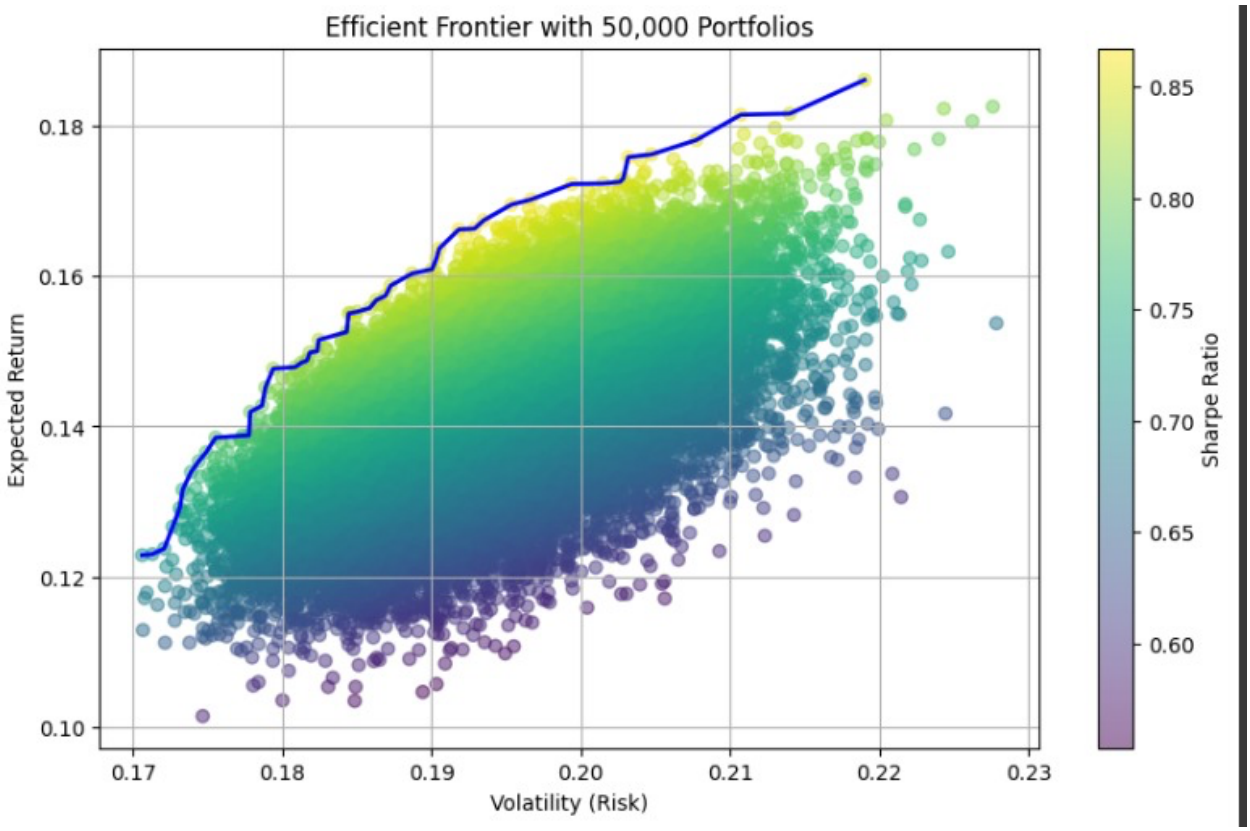


Figure 1: Efficient Frontier with 50,000 Portfolios. Color indicates the Sharpe ratio. The blue curve represents the upper envelope of optimal portfolios.

9 Hierarchical Portfolio Optimization using HRP and HERC

9.1 Introduction

Portfolio optimization involves selecting the proportion of various assets in a portfolio to achieve desired objectives such as maximizing return or minimizing risk. Traditional approaches like Modern Portfolio Theory (MPT) rely heavily on the estimation and inversion of the covariance matrix, which can lead to unstable solutions in the presence of noisy or limited data.

To address these challenges, we explore two modern, hierarchical alternatives:

- Hierarchical Risk Parity (HRP)
- Hierarchical Equal Risk Contribution (HERC)

Both approaches utilize clustering techniques to improve portfolio robustness and risk allocation.

9.2 Clustering Workflow

The foundation of both HRP and HERC is a clustering process that organizes assets based on their historical correlations. The main steps include:

9.2.1 Correlation-Based Distance

A pairwise correlation matrix is computed using the log returns of asset prices. This is converted to a distance matrix using the formula:

$$D(i,j) = \sqrt{0.5 \times (1 - \rho(i,j))}$$

where $\rho(i,j)$ is the correlation between assets i and j.

9.2.2 Hierarchical Clustering

The distance matrix is passed through a hierarchical clustering algorithm. We use the *average linkage* method, where the distance between two clusters is the average pairwise distance between elements of the two clusters.

9.2.3 Dendrogram and Quasi-Diagonalization

The output of the clustering is a dendrogram. Assets are reordered based on this tree structure so that similar assets appear adjacent in the matrix. This process, known as *quasi-diagonalization*, results in a reordered covariance matrix with higher correlations near the diagonal, improving the stability of further computations.

9.3 Hierarchical Risk Parity (HRP)

HRP constructs a portfolio by recursively splitting the asset tree (obtained from clustering) and allocating weights in inverse proportion to the risk (volatility) of subclusters.

Key Properties:

- Weights are derived without inverting the covariance matrix.
- Allocation is hierarchical and risk-aware.
- Robust to estimation errors in input data.

At each split in the hierarchy, the weight assigned to a branch is proportional to the inverse of its total risk. The final asset weight is the product of allocations at each level of the tree.

9.4 Hierarchical Equal Risk Contribution (HERC)

HERC enhances HRP by enforcing equal risk contribution from each subcluster at every node in the hierarchy. Instead of just allocating inversely to volatility, HERC computes weights such that each group contributes equally to overall portfolio risk.

Key Properties:

- Ensures risk contributions are evenly distributed across the tree.
- Prevents overly volatile assets from dominating the portfolio.
- Especially useful in highly volatile or unbalanced datasets.

9.5 Advantages Over Traditional Methods

HRP and HERC offer significant advantages over classical optimization frameworks like Mean-Variance Optimization:

- **No matrix inversion:** Makes them more numerically stable.
- **Robust to noise:** Performance remains reliable even with noisy or sparse data.
- **Improved diversification:** Naturally groups similar assets, reducing concentration risk.
- **Hierarchy-aware:** Provides an intuitive structure to asset relationships.

9.6 Conclusion

Hierarchical methods like HRP and HERC provide powerful alternatives to traditional portfolio construction techniques. By leveraging correlation structures and recursive logic, they allow for more stable, interpretable, and risk-balanced portfolios. These methods are particularly well-suited for modern financial markets where data quality and stability are major concerns.

10 Transaction Costs and Dynamic Weight Switching

Transaction Costs (TC)

Transaction costs are the hidden costs associated with trading. These include:

- Brokerage and commission fees charged by platforms for each trade
- Taxes and statutory charges that vary across regions and asset classes
- Slippage, which occurs when there is a difference between the expected price of a trade and the actual execution price due to market movement

These costs may seem small individually but can add up significantly, especially for strategies that involve frequent trading.

Why transaction costs matter:

- A strategy with high returns might underperform once these costs are accounted for, especially if it trades too often.
- Rebalancing the portfolio too frequently increases costs, while rebalancing too infrequently may reduce responsiveness to market changes.
- Efficient strategies aim to find a balance — rebalancing just enough to stay aligned with the market while minimizing unnecessary trading.

Dynamic Weight Switching

In a changing market, keeping portfolio weights fixed is often suboptimal. Dynamic switching involves adjusting asset weights over time to adapt to current conditions.

Why is it necessary?

- Market conditions change: Asset volatilities and correlations vary over time. During crises, assets tend to move together, reducing the benefits of diversification.
- Performance of fixed strategies can decay: A strategy tuned for a bull market might underperform in a bearish or sideways market.
- Better risk management: When an asset becomes more volatile or illiquid, its weight can be reduced to maintain overall portfolio stability.

Common Dynamic Allocation Techniques:

- Volatility-Based Allocation: Assets with higher recent volatility are given lower weights to control risk.
- Risk Parity: Allocates weights such that each asset contributes equally to overall portfolio risk.
- Model-Based Allocation: Techniques like HRP (Hierarchical Risk Parity) and HERC (Hierarchical Equal Risk Contribution) group assets by similarity and allocate weights in a risk-efficient manner.

Methods of Rebalancing

Time-Based Rebalancing: Rebalance at fixed intervals — monthly or quarterly. **Threshold-Based Rebalancing:** Rebalance if any asset's weight deviates beyond a threshold. **Correlation-Based Rebalancing:** Compare portfolio returns over the most recent 3 months with those in the previous 3 months. If the correlation breaks down, it signals a structural shift and triggers rebalancing. Clustering may also be used to regroup assets based on updated behavior.

Our Implementation in Project

- Transaction costs are included in portfolio performance calculations.
- We applied correlation-based dynamic switching by comparing recent return windows.
- We used clustering to group similar stocks for smarter weight adjustment.
- Allocation was performed using volatility-based and model-driven (HRP, HERC) techniques.

This combination produced a robust, adaptable, and cost-aware portfolio strategy.

Code: Correlation-Based Rebalancing with Transaction Costs

```
def correlation_rebalancing_with_tc(returns, initial_weights, lookback_months=3,
    corr_threshold=0.82, cost_rate=0.003):
    from riskfolio import HCPortfolio
    lookback_days = 21 * lookback_months
    current_weights = initial_weights.to_frame().T
    portfolio_values = [1.0]
    portfolio_values_with_cost = [1.0]
    first_date = returns.index[0]
    current_weights['Date'] = first_date

    for i in range(lookback_days, len(returns), 21):
        current_date = returns.index[i]
        recent = returns.iloc[i-lookback_days:i]
        previous = returns.iloc[i-2*lookback_days:i-lookback_days]
        if recent.isna().any().any() or previous.isna().any().any():
            continue
        corr_diff = (recent.corr() - previous.corr()).abs().mean().mean()
        rebalance = False
        if corr_diff > (1 - corr_threshold):
            rebalance = True
            hrp = HCPortfolio(returns=recent)
            new_weights = hrp.optimization(model='HRP', max_k=10, leaf_order=True)
                .T
        period_returns = returns.iloc[i-21:i]
        weighted_return = (current_weights.drop('Date', axis=1).values * (1 +
            period_returns)).sum(axis=1)
        new_value = portfolio_values[-1] * weighted_return.prod()
        portfolio_values.append(new_value)
        if rebalance:
            tc = sum([
                max(0, current_weights[col].values[0] - new_weights[col].values
                    [0]) * cost_rate
                for col in current_weights.columns if col != 'Date' and col in
                    new_weights.columns
            ])
```

```
            new_value_with_cost = portfolio_values_with_cost[-1] * (1 - tc) *
                weighted_return.prod()
            current_weights = new_weights.copy()
        else:
            new_value_with_cost = portfolio_values_with_cost[-1] * weighted_return
                .prod()
        portfolio_values_with_cost.append(new_value_with_cost)
        current_weights['Date'] = current_date

    return pd.DataFrame({
        'Date': [first_date] + [returns.index[i] for i in range(lookback_days, len
            (returns), 21)],
        'Without Cost': portfolio_values,
        'With Cost': portfolio_values_with_cost
    })
```

11 Web Deployment

Web deployment refers to the process of making a web application available to users over the internet. It involves preparing the application for production, configuring servers, and ensuring scalability, security, and maintainability.

11.1 Django

Django is a powerful, high-level web framework built in Python that emphasizes rapid development and clean, pragmatic design. It adopts the Model–View–Template (MVT) architecture, which separates data handling, business logic, and presentation for better maintainability.

For our project, Django provided:

- A built-in admin interface for efficient data management.
- URL routing for clean navigation across the application.
- High reusability and modularity, reducing development effort and ensuring scalability.

11.2 Implementation in Django

models.py The `models.py` file in our Django application contains a custom function `risk_parity()` that performs hierarchical risk parity analysis on financial data. The process involves:

- **Correlation & Distance Calculation** – Computes the correlation matrix between assets and converts it into a distance matrix.
- **Hierarchical Clustering** – Groups assets using hierarchical clustering to identify similar behavior patterns.
- **Quasi-Diagonalization** – Reorders the correlation matrix for better visualization of asset clusters.
- **Visualization Generation** – Creates a dendrogram and a heatmap, encodes them in Base64 for direct embedding into HTML templates.
- **Portfolio Weight Calculation** – Assigns equal weights to all assets (a placeholder for future HRP logic).
- **HTML Output** – Converts results into an HTML table for rendering in the Django front end.

admin.py This code bridges data analysis with web integration by dynamically generating financial analytics and visualizations that are directly displayed in our web application.

forms.py

- Defines `PortfolioOptimizerForm` for collecting user inputs.
- Includes a file upload field for a returns CSV file.
- Provides a dropdown menu to choose the optimization model (Mean–Variance, Black–Litterman, or Risk Parity).
- Validates that the uploaded file is in CSV format.

- Uses custom styling for the dropdown to match the dark-themed UI.

views.py

- Uses `@login_required` to restrict access to logged-in users only.
- `welcome_view` – Displays the main welcome page with navigation buttons.
- `know_the_models_view` – Shows an explanation of different portfolio optimization models.
- `portfolio_optimizer_view` –
  - Handles file upload and model selection from the form.
  - Reads and cleans the uploaded CSV file using pandas.
  - Runs the selected model (Mean–Variance or Risk Parity) and returns results in HTML format.
  - Displays optimization results or error messages on the same page.
- `about_us_view` – Renders the "About Us" page.
- `contact_us_view` – Renders the "Contact Us" page.

urls.py Defines all URL routes for the application, including:

- Authentication routes for login and logout.
- Mapping URLs to their respective views:
  - `/` or `/welcome/` → Welcome page
  - `/know-the-models/` → Information about optimization models
  - `/portfolio-optimizer/` → Portfolio optimization tool
  - `/about-us/` → About Us page
  - `/contact-us/` → Contact Us page

11.3 Template Rendering

In the development of dynamic web applications, a fundamental architectural pattern involves the use of templates to separate content from presentation. This paradigm, commonly referred to as **template rendering**, is central to creating scalable and maintainable web systems. It allows for the systematic construction of web pages by combining a static structure with dynamic data. The final output is a complete HTML document, often accompanied by CSS for styling, which defines the visual appearance and layout of the page.

The process begins when a web server receives a client request. A designated function, or "view," is invoked to handle this request. The view does not directly generate raw HTML; instead, it orchestrates data retrieval and rendering. It selects an appropriate template, typically an HTML file containing static markup and special placeholders (template tags) where dynamic content should be inserted.

A **context** is then constructed — a data structure that encapsulates all the variables required to populate the template's placeholders. This context may include data from a database, user session

information, or other application-specific variables. The view passes both the selected template and the prepared context to a **template engine**.

The template engine parses the template file, identifies placeholders, and substitutes them with the corresponding values from the context. The result is a complete HTML document, which is then sent to the client's browser for rendering.

For example, in our application, the `contact(request)` function serves as the view for the Contact page. Instead of manually constructing HTML, it uses the `render()` function, passing the request object and the path to the template file (`'app1/contact.html'`). This approach encapsulates the rendering logic and promotes a clear separation between business logic and presentation, which is a cornerstone of modern software engineering.

12 Webpage Functionality Demonstration

12.1 Login Page

This page serves as the gateway to the platform, ensuring secure and authenticated access for users.

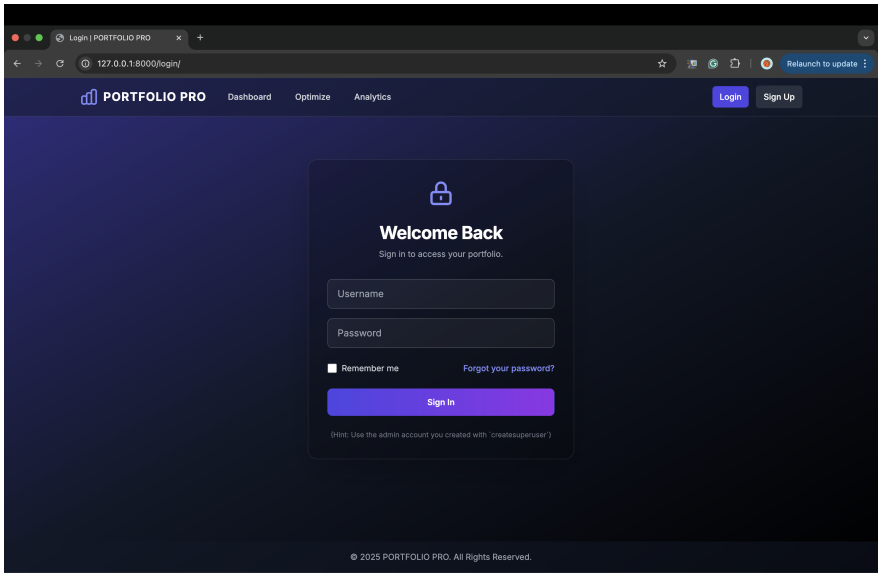


Figure 2: Login Page Interface

12.2 Dashboard

The dashboard provides an overview of the platform's key features and analytics in a single glance.

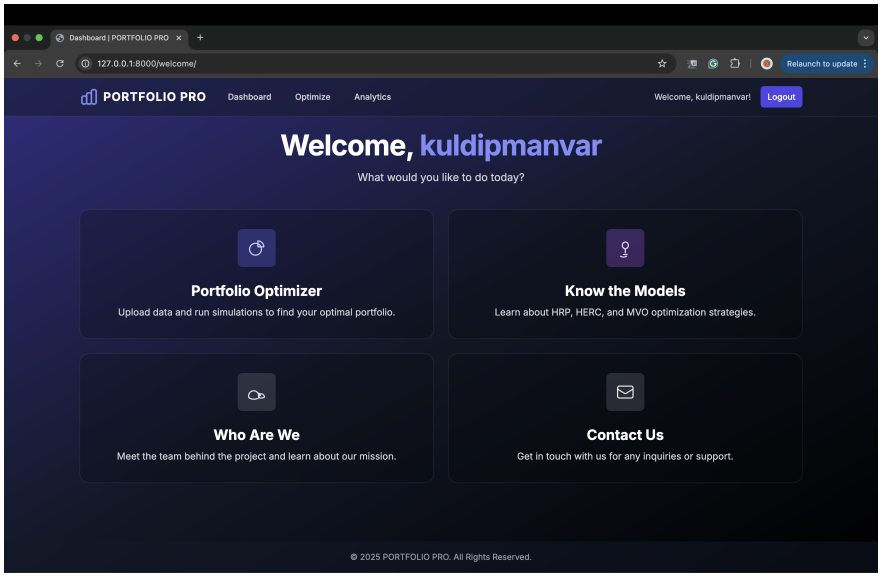


Figure 3: Dashboard Overview

12.3 Portfolio Optimizer

This section allows users to perform portfolio analysis and optimization using historical market data.
For example, we have uploaded a CSV file containing the closing price data of 48 NSE-listed stocks, dating from 1/1/2020 to 30/7/2025.

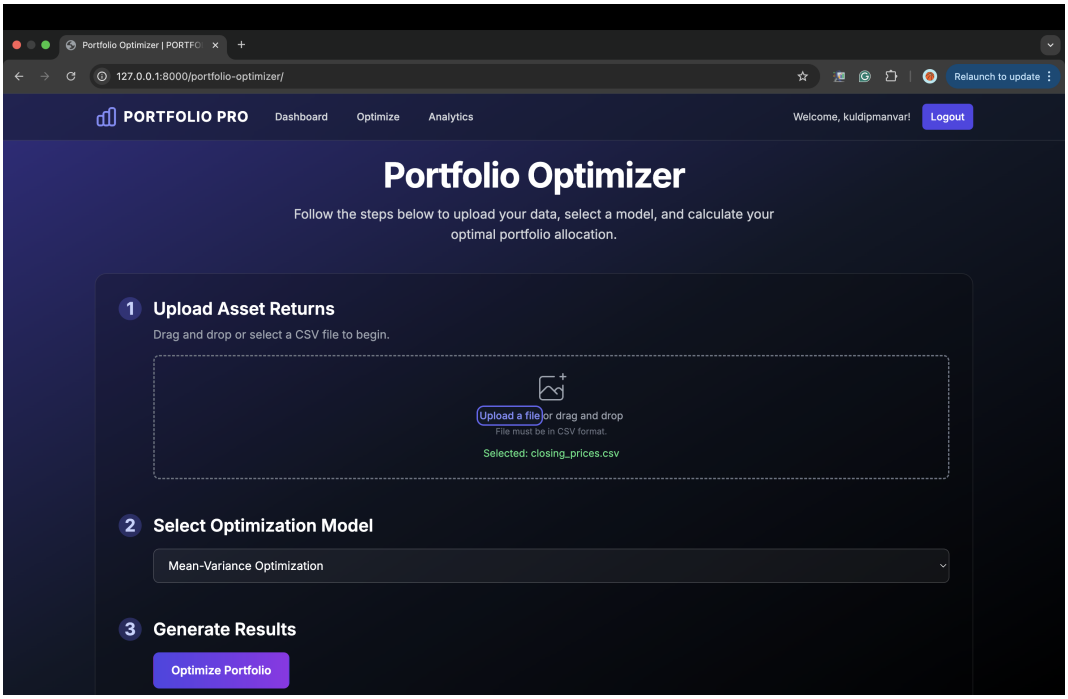
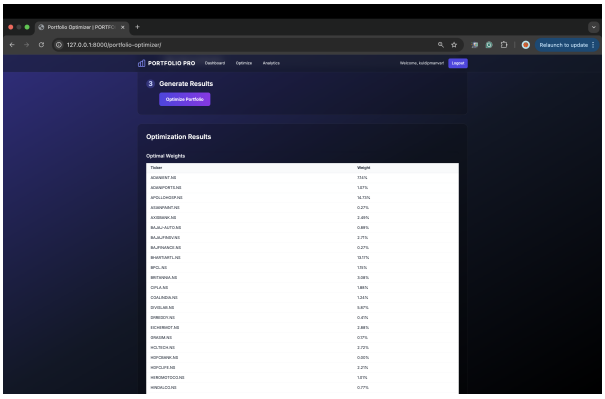
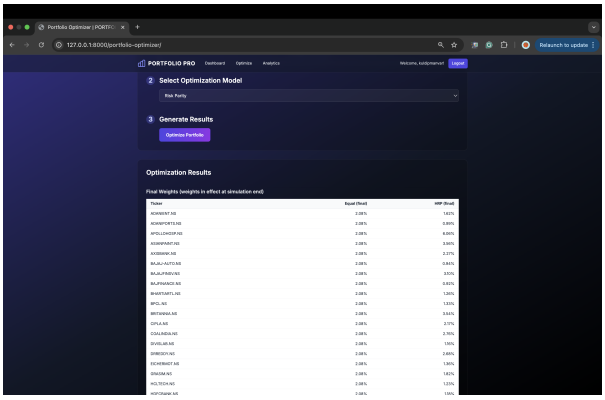


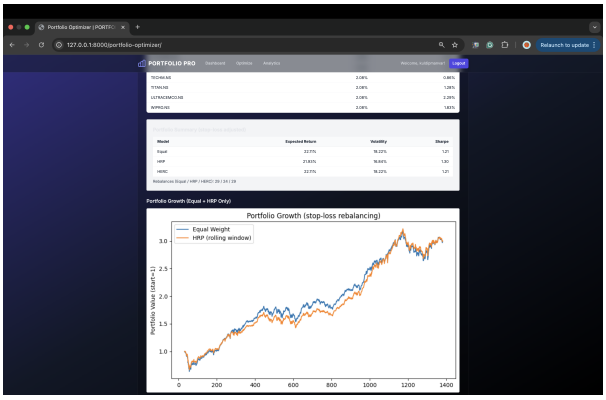
Figure 4: Portfolio Optimizer Interface

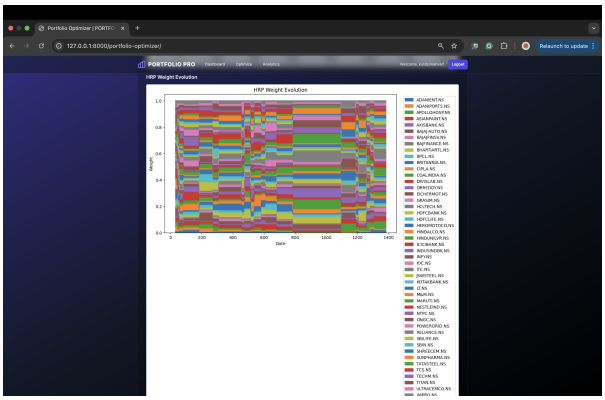
12.3.1 Mean Variance Optimization

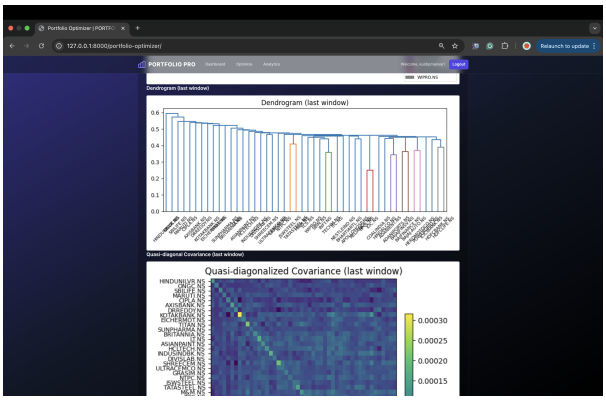


12.3.2 Risk Parity



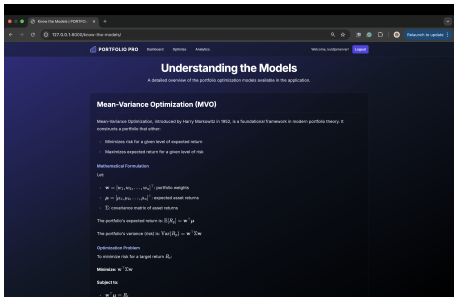


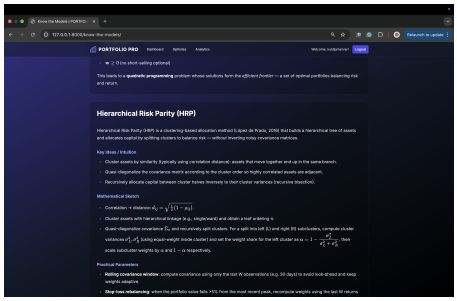


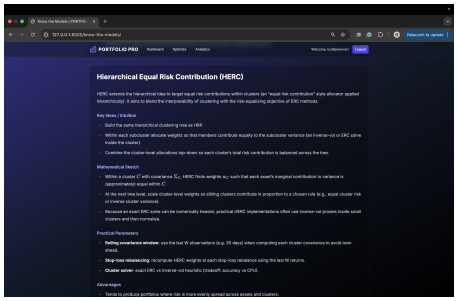


12.4 Understanding the Models

Provides detailed explanations of the mathematical models and assumptions behind portfolio strategies.







12.5 About Us

Introduces the team and vision behind the platform.

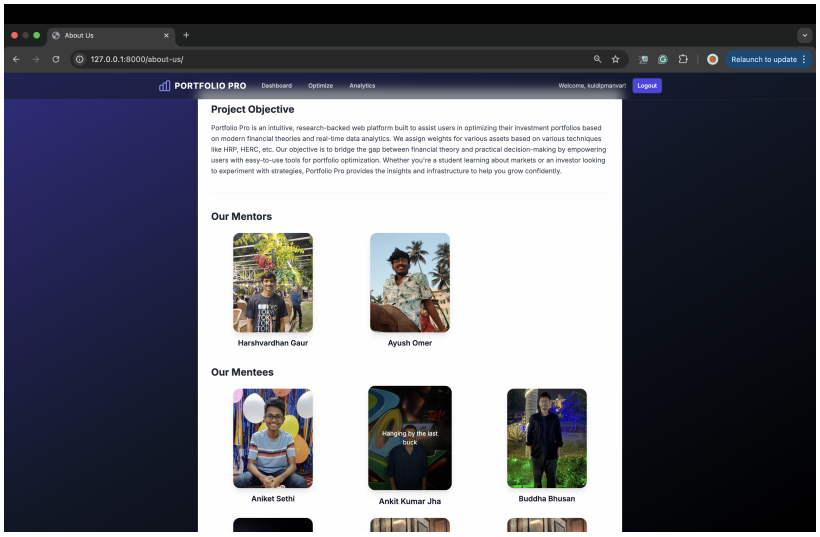


Figure 8: About Us Page