Stock Portfolio Optimization using PyPortfolio

Minor Project-I Report

Submitted for the partial fulfillment of the degree of

Bachelor of Technology

In

Mathematics and Computing

Submitted By

Anupam Awasthi

0901MC221015

UNDER THE SUPERVISION AND GUIDANCE OF

Dr .Vikas Shinde

Professor

Department of Engineering Mathematics & Computing

माधव प्रौद्योगिकी एवं विज्ञान संस्थान, ग्वालियर (म.प्र.), भारत

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR (M.P.), INDIA

Deemed University

(Declared under Distinct Category by Ministry of Education, Government of India)

NAAC ACCREDITED WITH A++ GRADE

July-December 2024

DECLARATION BY THE CANDIDATE

I hereby declare that the work entitled "Stock Portfolio Optimization using PyPortfolio" is my work, conducted under the supervision of **Dr.Vikas Shinde, Professor,** during the session July-Dec 2024. The report submitted by me is a record of bonafide work carried out by me.

I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Anupam Awasthi

0901MC221015

Date:

Place: Gwalior

This is to certify that the above statement made by the candidates is correct to the best of my knowledge and belief.

Guided By:

Dr. Vikas Shinde
Professor
Department of Engineering Mathematics & Computing
MITS-DU, Gwalior

Departmental Project Coordinators

Dr. Atul Kumar Ray
Assistant Professor
Engineering Mathematics & Computing,MITS-DU, Gwalior

Dr. Minakshi Dahiya
Assistant Professor
Engineering Mathematics & Computing, MITS-DU, Gwalior

Approved by HoD

Dr. D. K. Jain
Professor and Head
Engineering Mathematics & Computing,MITS-DU, Gwalior

PLAGIARISM CHECK CERTIFICATE

This is to certify that I/we, a student of B.Tech. in **Mathematics & Computing** have checked my complete report entitled **"**Stock Portfolio Optimization using PyPortfolio**"** for similarity/plagiarism using the "Turnitin" software available in the institute.

This is to certify that the similarity in my report is found to be …… which is within the specified limit (30%).

The full plagiarism report along with the summary is enclosed.

Anupam Awasthi

0901MC221015

Checked & Approved By:

Dr. Barkha Tiwari
Assistant Professor
Engineering Mathematics & Computing
MITS-DU, Gwalior

ABSTRACT

Stock Portfolio Optimization, is the critical area of financial management which tries to maximize returns and minimize risks. A Python-focused library to purpose and optimize portfolios for investing, we utilize this project throughout. It is built around the principles of modern portfolio theory (MPT) using concepts like efficient frontier, risk-return trade-off analysis, and maximization of Sharpe ratio.

It starts with data capture from financial APIs pertaining to past stock rates, market status and economic resources. It is used to compute expected returns, covariance matrices, or risk profiles (PyPortfolioOpt 2023). Diversified portfolios according to risk appetite are built using optimization techniques, such as mean-variance optimization and risk parity allocation.

The results of the project showcase how quantitative capabilities can add value in making investment decisions through efficient portfolios that are optimized for returns over defined risk limits. This implementation is designed to be simple, reproducible, and easy to understand for real-world use cases which makes it useful for novices as well as experienced investors.

Key takeaways emphasize the necessity of diversification, the influence of risk appetite on portfolio construction, and the benefits of automated optimization methods. The research highlight the ability of pyportfolioopt to do higher portfolio management processes in a much simpler way and even as a tool for future research in financial analytics.

ACKNOWLEDGEMENT

I extend my sincere gratitude to the esteemed Vice Chancellor of the Deemed University**, Dr. R. K. Pandit**, and the respected Dean Faculty of Engineering & Technology, **Dr. Manjaree Pandit**, for their valuable support and guidance throughout this project.

I would sincerely like to thank my department, Department of Mathematics & Computing, for allowing me to explore this project & its domain. I humbly thank **Dr. D.K Jain** Professor & Head, Department of Mathematics & Computing for his continued support during this engagement which eventually eased the process & formalities involved.

Further, I thank my project Supervisor **Dr.Vikas shinde** from the Department of Engineering Mathematics & Computing. Their expert guidance, continuous support, and encouragement in my journey, help me to overcome challenges and achieve my target. l am also, deeply grateful to the esteemed faculties of the Department of Engineering Mathematics & Computing for their invaluable feedback, insightful suggestions, and continuous encouragement, which have played a pivotal role in shaping my project.

Anupam Awasthi

0901MC221015

CONTENT

Table of Contents

ACRONYMS

General Acronyms:

MPT: Modern Portfolio Theory

CAPM : Capital Asset Pricing Model

CML: Capital Market Line

SML: Security Market Line

ETF: Exchange-Traded Fund

Mathematical Acronyms: Statistical Acronyms:

VAR: Value at Risk

CVaR : Conditional Value at Risk.

SR: Helps to evaluate the performance as adjusted for risk

Sortino ratio: A measure of return relative to downside risk

Max Drawdown: The largest percentage drop from peak to trough of a portfolio

Code and Data Abbreviations:

Python: Popular programming language, also commonly used for data analysis and scientific computing

The pandas: Powerful, Flexible Open Source Data Analysis and Manipulation Tool Built on Top of the Python Programming Language

NumPy: A Python library for numerical computing

Matplotlib: Library for data visualization in Python

SciPy — Scientific library for Python

Yahoo Finance — Financial Data Provider and API

API : Application Programming Interface

NOMENCLATURE

Basic Terms

Portfolio: A set of investments.

Asset: A single investment in a portfolio.

Return: The gain or loss associated with an investment, typically expressed as a percentage.

Risk: The possibility of a loss in the value of an investment.

Risk and Return Metrics

Expected Return: The return you can expect to receive from a particular investment.

Volatility — the extent to which returns differ from their mean/expected return

Standard Deviation: A measure that is used to quantify the amount of variation or dispersion in a data set.

Covariance: The extent to which two variables change together.

Correlation : A normalized assessment of the association between two variables.

Sharpe Ratio: Risk-adjusted return, excess return over risk-free for unit to total volatility

Sortino Ratio: A measure of return adjusted for downside risk

Maximum Drawdown: The largest decline of the portfolio from its peak to trough.

Methods for optimizing a portfolio

Mean-Variance Optimization: An approach that tries to achieve the highest returns for a level of risk or least risk for a level of return

Modern Portfolio Theory (MPT): A statistical theory for quantifying risk and return of a diversified investment portfolio.

Capital Asset Pricing Model(CAPM): A model that establishes the relationship between asset systematic risk and expected return

Black-Litterman Model: A method that integrates market equilibrium returns with investor opinions to obtain improved portfolio weights.

Risk Parity: A style of portfolio allocation designed to allocate weights based on risk contribution instead of expected return.

Other Terms

Efficient FrontierScheme of optimal portfolios providing the maximum expected return for a given level of risk

Risk Aversion: The amount of risk that an investor is willing to trade for greater returns.

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Portfolio Optimization in Financial Management (Overview)

Portfolio optimization[6] is a mainstay of financial management that seeks to develop investment portfolios that generate the most desirable level of returns for a given amount of risk. Portfolio optimization is a process of choosing the combinations of assets that will minimize risk and maximize returns with roots in Modern Portfolio Theory (MPT). Today's complex financial landscape requires this approach, where investment animals dwell in volatile markets, pools of investment instruments with different risk profiles, and the trajectory of the economic conditions. Portfolio optimization uses systematic and data driven ways to assist investors in making their investment decisions rationalized using the financial goals and risk tolerance.

1.2 The significance of risk return balance.

Investment strategy is all about choosing a risk versus return trade off. Strangely, the rewards come with the threat of major losses. However, low risk investments are stable while they deliver low returns. Growth between these extremes is key to sustainable wealth growth. Through portfolio optimization, we offer a structured framework for making this trade off in the sense that portfolios will be customized to the investor's preferences and the given market dynamics. Investing in different asset classes and industries allows investors to balance the risk associated with market fluctuations, while still keeping in growth potential.

1.3 Quantitative Tools and Libraries as a Tool of Investment Strategies

In the era of big data and high powered computing, quantitative tools have become indispensible in financial decision making. In its simplest form, these tools can simplify the task of analyzing a large dataset, identifying underlying patterns, and then implementing sophisticated optimization techniques. PyPortfolioOpt libraries make portfolio construction process easier by providing ready use functions to calculate the expected return, risk analysis, and portfolio allocation optimization. It spans theoretical finance and practical implementation; it helps you focus on strategy rather than on complicated calculations.

1.4 The PyPortfolioOpt and its introduction.

We develop PyPortfolioOpt, a Python library for efficient portfolio optimization. It provides an easy to adopt mechanism of advanced financial concepts such as efficient frontier, risk modeling or mean variance optimization. PyPortfolioOpt features user friendly interface as well as a rich set of functionalities which makes it facility for both beginners and experienced financial analysts. It is a versatile tool in the portfolio manager's toolkit, supporting such a range of optimization objectives as maximizing returns or minimizing volatility. With PyPortfolioOpt, not only can investors build data driven portfolios that are tailored to their goals, but it also boosts their decision making in a highly competitive investment environment.

CHAPTER 2: LITERATURE SURVEY

2.1 Modern Portfolio Theory (MPT): Markowitz's Approach

Since 1952, that revolution was in the creation of Modern Portfolio Theory (MPT) by Harry Markowitz. One important thing that MPT offers is a mathematical way to construct portfolios that achieve highest expected return for a fixed amount of risk, or lower risk for a specific expected return. This theory revolves around the idea of diversification, investing in multiple assets to cut down risk, in total portfolio. According to MPT the expected returns, variances and covariances of different assets will be analyzed to arrive at the best possible portfolios from the optimal combinations of investments, the efficient frontier, which is a boundary that represents the best possible portfolios.

Key Concepts

2.1.1 Risk(Standard Deviation, Variance)

Uncertainty around the returns of an investment is known as Risk. It is quantified by:

- Standard Deviation: It measures the dispersion of returns for an asset around its mean. Greater standard deviation means greater risk.

- Variance: A measure of risk which includes the variability in returns squared.

Lowering risk is as worthy a goal in portfolio optimization as one would expect to find in maximizing returns.

2.1.2 Return (Expected Return)

In the above, Expected Return is the expected average return from an asset or portfolio over a specify period. It is calculated as:

$$Expected\ Return = \sum_{i=1}^{n} (P_i . R_i)$$

Expected return in the portfolio is a weighted average of the returns on individual assets linked to their allocation in the portfolio in portfolio context.

2.1.3 Correlation / Covariance

- The strength and direction of the relationship between the returns of two assets, called correlation, is measured.

-We range from negative 1 (perfect negative correlation) to 1 (perfect positive).

How two assets returns move together is quantified by covariance.

$$\text{Covariance}(A, B) = E[(R_A - \mu_A)(R_B - \mu_B)]$$

Where μ_A and μ_B are the mean returns of assets A and B.

- Combining assets with low or negative correlations means diversification benefits: your risk is lowered as a whole.

2.2 The Importance of Efficient Frontier

Efficient frontier is a graphical portrayal of Portfolio optimization which provides the maximum return for every level of risk. Those portfolios on the frontier are the best risk-return trade offs whereas all other portfolios lying below the frontier are considered sub-optimal. The frontier shape depicts the decreasing return to increasing risk, and hence the importance of an efficient allocation. One of the most basic tasks of portfolio building is finding the efficient frontier.

2.3 Sharpe Ratio: Measuring Risk Adjusted Returns

A widely used metric to evaluate an investment's performance, that adjusts for risk, is the Sharpe Ratio. It measures the excess return per unit of risk:

$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma_p}$$

Where

R_p: Portfolio return

R_f: Risk − free rate of return

σ_p: Portfolio standard deviation

This means that a good Sharpe Ratio shows better risk adjusted performance. In portfolio optimization, maximizing the Sharpe Ratio is often what gets you the best risk and return compromise.

CHAPTER 3: SETTING UP THE ENVIRONMENT

This section outlines, in the order needed, to have pyportfolioopt plus necessary libraries and data sources set up in environment.

3.1 PyPortfolioOpt Installation and Setup

A Python[7] library which simplifies the tasks of portfolio optimization is PyPortfolioOpt. To get started, you need to install the library using pip:

```
pip install PyPortfolioOpt
```

3.2 Required Libraries

However, if you want to use Python for portfolio optimization, you'll need extra libraries for data handling, analysis and visualization.It involves working on organizing stock data with pandas Data manipulation and analysis (such as data manipulation and analysis).In other words, Numpy is a numerical computing adding that comes in handy while working with arrays and matrices.Plotting stock data, risk metrics and efficient frontier plots in matplotlib.Historical stock price data from Yahoo Finance is picked up by yfinanceely, you'll need additional Python libraries for data manipulation, analysis, and visualization. Below is a list of the key libraries and their purposes:

Library	Purpose
Pandas	Data manipulation and analysis (e.g., organizing stock data).
NumPy	Numerical computing (e.g., working with arrays and matrices).
matplotlib	Visualization of stock data, risk metrics, and efficient frontier plots.
yfinance	Fetching historical stock price data from Yahoo Finance.

3.3 Stock Price and Financial Data Data Sources

In order for you to optimize a portfolio, you'll need historical price data of the stocks or any assets that you might work with. Reliable data sources include:

Yahoo Finance: Using yfinance library, it is easily accessible.

Alpha Vantage: It is dependent on an API key to provide the data.

Quandl: Provide financial, economic and alternative data.

4. Example: Fetching Data and Set Up Enviroment

Here's a step-by-step example of setting up the environment and retrieving historical stock price data using yfinance:

3.3.1 We import the needed libraries.

```
import matplotlib.pyplot as plt
import yfinance as yf
import numpy as np
import pandas as pd
import seaborn as sns
```

3.3.2 Fetch Historical Stock Data

Use yfinance to download adjusted closing prices for selected stocks:

```
tickers = ["TCS.NS", "RELIANCE.NS", "INFY.NS", "HDFCBANK.NS", "ICICIBANK.NS"]

# Fetch data for each ticker
data = {}
for ticker in tickers:
    stock_data = yf.download(ticker, start="2023-01-01", end="2023-12-31")
    data[ticker] = stock_data

# Combine data into a single DataFrame
stock_df = pd.concat(data, axis=1)

# Display the first few rows
print(stock_df.head())
```

3.3.3 Visualize the Data

Plot the stock price trends to gain initial insights:

```
plt.figure(figsize=(14, 7))
sns.set(style='whitegrid')
sns.lineplot(data=stock_data, x='Date', y='Adj Close', hue='Ticker', marker='o')
```

CHAPTER 4: CORE FEATURES OF PYPORTFOLIOOPT

4.1 PyPortfolioOpt's core features

Based on Python library PyPortfolioOpt, it has been developed to be a powerful tool for financial portfolio optimization. A full suite of tools for data preparation, metric calculation and portfolio optimization techniques is provided. Here's a breakdown of its key features:

Data Preparation:

- Data Cleaning and Organization:Functions in PyPortfolioOpt clean and organize financial data so it can be used for analysis. It includes how to handle missing values, outliers and inconsistencies.

- Data Integration:I support all kinds of data sources, such as CSV, Excel, and even databases, which means no additional work is needed to feed your data into Pig.

Metric Calculation:

- Expected Returns:Expected returns of individual assets are essential to estimated portfolio returns, and it calculates expected returns for individual assets.

- Covariance Matrix:It estimates the covariance matrix, which reflects the exposure of asset returns to portfolio risk.

- Risk Models:They provide a number of risk models to analyze risk as it pertains to investments, namely:

  - Historical Volatility: Estimating volatility uses historical price data.

  - Exponentially Weighted Moving Average (EWMA): And gives more weight to recent data points to follow changing patterns of volatility.

  - Generalized Autoregressive Conditional Heteroskedasticity (GARCH): Models time variation volatility, based on previous volatility and other factors.

4.2 Portfolio Optimization Techniques:

Mean-Variance Optimization: A standard procedure, expected return vs risk (measured as variance). It seeks to identify an optimal portfolio for a particular return ambition in terms of risk, or an optimal risk for a given return ambition.

Risk Parity Allocation: It allocates weights assets according to their marginal contribution in reducing the portfolio risk. The aim of this approach is to spread risk evenly between all the assets, to dampen impact of extreme events.

Black-Litterman Model: It combines market expectations with the investors' views to make portfolio weights of each asset. By enabling investors to 'bake in' their own view of asset returns, within the market consensus, it is better for forecasting and long-term decisions.

4.3 Additional Features:

Customizable Objective Functions: With PyPortfolioOpt you can create custom objective functions to optimise your investment for speficic investment goals.

Constraints: Second, you can impose different constraints on portfolio weights for example minimum/maximum weight limits, budget constraints, and leverage limits.

Backtesting: Tools are provided by the library to backtest optimized portfolios on historical performance.

Visualization: It provides visualization for plotting efficient frontiers, risk return scatter plots, and other plots relevant to the problem category.

CHAPTER 5: BUILDING AN OPTIMIZED PORTFOLIO

Building an Optimized Portfolio: A Step-by-Step Guide

5.1 Data Preparation

* Gather Historical Data: Start by collecting historical price data for the assets you're considering working with. It can be taken from financial data providers or from platforms like Yahoo Finance, Google Finance or Bloomberg.

* Clean and Organize Data: Remove outliers or data that's an outlier compared to the remainder of the dataset. Deal with missing values and outliers meaningly.

5.2 Calculate Key Metrics

* Expected Returns: Find out the expected returns of each asset. That could be done with historical average returns or, more sophisticatedly, with CAPM or Fama French models.

* Covariance Matrix:## From the covariance matrix calculate the relationship between asset returns. And this matrix will be indispensable for our portfolio optimization.

5.3 Portfolio Optimization

* Mean-Variance Optimization:In this classic approach we attempt to find the optimal portfolio that gives maximum return for a given amount of risk or minimum risk given a maximum amount of return.

* Risk Parity: This strategy determines the asset weight allocations that minimize the portfolio risk contributions. The aim is to spread risk equally on all assets.

* Black-Litterman Model: This model combines market expectations and investor opinions to better determine portfolio weights. It means that the investors can add their own sources of believed return on assets.

5.4 Evaluate and Visualize

* Efficient Frontier: Visualize the trade off between risk and return while plotting the efficient frontier. This will then enable you to know the precise portfolio that is best for your risk tolerance.

* Performance Metrics: You calculate the Sharpe ratio, the Sortino ratio, and the maximum drawdown to compare the portfolio performance.

* Backtesting: Test the portfolio's performance (against a few alternative implementations) in back test mode by using historical data so that we can be confident in its robustness and reliability.

CHAPTER 6: VISUALIZATION AND INSIGHTS

Visualizing the end result is a key step in the whole process of portfolio optimization as it enables the investors to understand the risk return trade off, diversification effects and impacts from their risk tolerance. With matplotlib it provides tools for plotting and analysis of optimized portfolios — **PyPortfolioOpt.**

6.1 Working on Visualising the Efficient Frontier Using Matplotlib

Efficient frontier is the collection of portfolios which provide the optimal trade off between highest expected return and given risk. With PyPortfolioOpt, we can plot this frontier for a better understanding.

Code to Plot the Efficient Frontier:

```
# Plot the efficient frontier
plt.figure(figsize=(14, 7))
plt.scatter(portfolio_results['Volatility'], portfolio_results['Return'], c=portfolio_results['Sharpe Ratio'], cmap='viridis', marker='o', s=10, alpha=0.3)
plt.colorbar(label='Sharpe Ratio')

# Mark the maximum Sharpe ratio portfolio
plt.scatter(max_sharpe_volatility, max_sharpe_return, color='red', marker='*', s=200, label='Max Sharpe Ratio')

# Mark the minimum volatility portfolio
min_vol_idx = portfolio_results['Volatility'].idxmin()
min_vol_volatility = portfolio_results.iloc[min_vol_idx]['Volatility']
min_vol_return = portfolio_results.iloc[min_vol_idx]['Return']
plt.scatter(min_vol_volatility, min_vol_return, color='blue', marker='*', s=200, label='Min Volatility')

# Labels and title
plt.title('Efficient Frontier for Random Portfolios', fontsize=16)
plt.xlabel('Volatility (Risk)', fontsize=14)
plt.ylabel('Return', fontsize=14)
plt.legend(loc='upper left')
plt.grid(True)
plt.tight_layout()
plt.show()

# Print the details of the portfolio with the maximum Sharpe ratio
print(f"Portfolio with Maximum Sharpe Ratio:")
print(f"Expected Return: {max_sharpe_return:.4f}")
print(f"Volatility: {max_sharpe_volatility:.4f}")
print(f"Sharpe Ratio: {max_sharpe_sharpe_ratio:.4f}")
print(f"Weights: {dict(zip(tickers, max_sharpe_weights))}")
```

6.2 Interpretation of Portfolio Metrics

Risk vs. Return

Risk (Volatility): On the efficient frontier plot, portfolio risk represents on the x axis and it incrementally increases with increasing portfolio returns.

Expected Return: Depicted on the y-axis, depicting the fact potential portfolio will perform.

The curve demonstrates the trade-off: However, the sources of higher returns are normally accompanied by higher risk.

Key Insight: The efficient frontier portfolios are optimal and portfolios below are suboptimal.

Diversification Effects

Dividing investments among different assets reduces the amount of unsystematic risk.

Focusing on the efficient frontier plot, as you vary the correlation between the assets the return you can get for a given level of risk tends to decrease.

Key Insight: Than others, a less diversified portfolio is closer to the efficient frontier.

Different versions of these trade-offs for different investor risk profiles

Risk-Averse Investors: Minimize volatility rather than prefer portfolios on the lower end of the efficient frontier.

Risk-Tolerant Investors: Maximize returns and risk at the same time, favoring higher end portfolios.

Balanced Investors: Go for portfolios at the 'knee' of the curve (where the tradeoff between risk and return is the best).

Key Insight: Portfolios can be customized according to an individual risk tolerance, then aligned with financial goals.

CHAPTER 7: CASE STUDY: A REAL-WORLD PORTFOLIO

This case study demonstrates how to use **PyPortfolioOpt** to optimize a stock portfolio, analyze its performance, and derive actionable insights. We will compare the optimized portfolio against a market index to assess its effectiveness.

7.1 Selected Stock Dataset Application for PyPortfolioOpt

Step 1: Data Collection

Get a set of stocks for the portfolio and using yfinance fetch historical price data of the stocks.

```
import yfinance as yf
import pandas as pd

# Define the stock tickers for some popular Indian companies
# Tickers use the NSE suffix for Yahoo Finance, e.g., 'TCS.NS' for TCS on NSE
tickers = ["TCS.NS", "RELIANCE.NS", "INFY.NS", "HDFCBANK.NS", "ICICIBANK.NS"]

# Fetch data for each ticker
data = {}
for ticker in tickers:
    stock_data = yf.download(ticker, start="2023-01-01", end="2023-12-31")
    data[ticker] = stock_data

# Combine data into a single DataFrame
stock_df = pd.concat(data, axis=1)

# Display the first few rows
print(stock_df.head())
```

Step 2: Optimize with Calculate Metrics

First, we estimate expected returns and the covariance matrix.

```
import matplotlib.pyplot as plt
import yfinance as yf
import pandas as pd
import seaborn as sns

# Define tickers and download the required data
tickers = ["TCS.NS", "RELIANCE.NS", "INFY.NS", "HDFCBANK.NS", "ICICIBANK.NS"]
stock_data = yf.download(tickers, start="2023-01-01", end="2023-12-31")['Adj Close']

# Calculate daily returns for each stock
daily_returns = stock_data.pct_change().dropna()  # pct_change() gives daily returns, dropna removes NaNs

# Compute the correlation matrix of daily returns
correlation_matrix = daily_returns.corr()

# Plot the correlation matrix as a heatmap
plt.figure(figsize=(10, 7))
sns.set(style="whitegrid")
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5, vmin=-1, vmax=1)

# Set plot labels and title
plt.title('Correlation Matrix of Daily Returns for Selected Stocks', fontsize=16)
plt.tight_layout()
plt.show()
```

Step 3: Optimize the Portfolio

For maximum Sharpe ratio, we use the PyPortfolioOpt.

```
tickers = ["TCS.NS", "RELIANCE.NS", "INFY.NS", "HDFCBANK.NS", "ICICIBANK.NS"]
stock_data = yf.download(tickers, start="2023-01-01", end="2023-12-31")['Adj Close']

# Calculate daily returns for each stock
daily_returns = stock_data.pct_change().dropna()

# Calculate the expected return (mean of daily returns) and covariance matrix of returns
expected_returns = daily_returns.mean() * 252  # Annualized expected returns (252 trading days)
cov_matrix = daily_returns.cov() * 252  # Annualized covariance matrix

# Number of random portfolios to simulate
num_portfolios = 10000

# Arrays to store results
portfolio_returns = []
portfolio_volatilities = []
portfolio_weights = []

# Generate random portfolios
for _ in range(num_portfolios):
    weights = np.random.random(len(tickers))
    weights /= np.sum(weights)  # Normalize weights so that they sum to 1

    # Calculate portfolio return and volatility
    portfolio_return = np.sum(weights * expected_returns)
    portfolio_volatility = np.sqrt(np.dot(weights.T, np.dot(cov_matrix, weights)))

    portfolio_returns.append(portfolio_return)
    portfolio_volatilities.append(portfolio_volatility)
    portfolio_weights.append(weights)
```

7.2 Results Analysis and Market Indices Comparison.

Step 1: Portfolio Allocation Visualization

To understand diversification display allocation weights.

```
# Plot the efficient frontier
plt.figure(figsize=(14, 7))
plt.scatter(portfolio_results['Volatility'], portfolio_results['Return'], c=portfolio_results['Sharpe Ratio'], cmap='viridis', marker='o', s=10, alpha=0.3)
plt.colorbar(label='Sharpe Ratio')

# Mark the maximum Sharpe ratio portfolio
plt.scatter(max_sharpe_volatility, max_sharpe_return, color='red', marker='*', s=200, label='Max Sharpe Ratio')

# Mark the minimum volatility portfolio
min_vol_idx = portfolio_results['Volatility'].idxmin()
min_vol_volatility = portfolio_results.iloc[min_vol_idx]['Volatility']
min_vol_return = portfolio_results.iloc[min_vol_idx]['Return']
plt.scatter(min_vol_volatility, min_vol_return, color='blue', marker='*', s=200, label='Min Volatility')

# Labels and title
plt.title('Efficient Frontier for Random Portfolios', fontsize=16)
plt.xlabel('Volatility (Risk)', fontsize=14)
plt.ylabel('Return', fontsize=14)
plt.legend(loc='upper left')
plt.grid(True)
plt.tight_layout()
plt.show()

# Print the details of the portfolio with the maximum Sharpe ratio
print(f"Portfolio with Maximum Sharpe Ratio:")
print(f"Expected Return: {max_sharpe_return:.4f}")
print(f"Volatility: {max_sharpe_volatility:.4f}")
print(f"Sharpe Ratio: {max_sharpe_sharpe_ratio:.4f}")
print(f"Weights: {dict(zip(tickers, max_sharpe_weights))}")
```

Step 2: Market Index Benchmarking

See how the portfolio does as compared to a benchmark, say the S&P 500.

Fetch Index Data:

```
data = {}
for ticker in tickers:
    stock_data = yf.download(ticker, start="2023-01-01", end="2023-12-31")
    data[ticker] = stock_data


# Combine data into a single DataFrame
```

Calculate Portfolio Returns:

```
# Define tickers and download the required data
tickers = ["TCS.NS", "RELIANCE.NS", "INFY.NS", "HDFCBANK.NS", "ICICIBANK.NS"]
stock_data = yf.download(tickers, start="2023-01-01", end="2023-12-31")['Adj Close']

# Calculate daily returns for each stock
daily_returns = stock_data.pct_change().dropna()

# Calculate the expected return (mean of daily returns) and covariance matrix of returns
expected_returns = daily_returns.mean() * 252  # Annualized expected returns (252 trading days)
cov_matrix = daily_returns.cov() * 252  # Annualized covariance matrix

# Number of random portfolios to simulate
num_portfolios = 10000

# Arrays to store results
portfolio_returns = []
portfolio_volatilities = []
portfolio_weights = []
```

Plot Performance:

```
tickers = ["TCS.NS", "RELIANCE.NS", "INFY.NS", "HDFCBANK.NS", "ICICIBANK.NS"]
stock_data = yf.download(tickers, start="2023-01-01", end="2023-12-31")['Adj Close']

# Calculate daily returns for each stock
daily_returns = stock_data.pct_change().dropna()  # pct_change() gives daily returns, dropna removes NaNs

# Plot the distribution of daily returns
plt.figure(figsize=(14, 7))
sns.set(style="whitegrid")
for ticker in tickers:
    sns.histplot(daily_returns[ticker], kde=True, label=ticker, bins=30, stat="density", common_norm=False)

plt.title('Distribution of Daily Returns for Selected Stocks', fontsize=16)
plt.xlabel('Daily Return', fontsize=14)
plt.ylabel('Density', fontsize=14)
plt.legend(title='Ticker', title_fontsize='13', fontsize='11')
plt.grid(True)
plt.tight_layout()
plt.show()
```

7.3 Insights and Learnings

Observations

Optimized Allocation: Portfolio weights after optimization are diversified by risk return metrics. Stable high return stocks are given higher weights and volatile stocks are given lower weights.

Performance:

The Sharpe ratio of the optimized portfolio may be better risk adjusted than the market index. Diversifying benefits may help the portfolio outperform the market index in times of instability.

Learnings

Risk Reduction: Combining assets with low correlation value, the diversification reduces unsystematic risk.

Importance of Data Quality: We need reliable historical data for an accurate optimization.

Customization: The fact that PyPortfolioOpt can customize for different risk profiles means that it suits different investor types.

CHAPTER 8: CONCLUSION

Portfolio optimization is a critical concept in the current financial management since it enables most strategies to achieve investment objectives while controlling risk. To this end, PyPortfolioOpt has developed strong and easy-to-use tools that can support investors in making the best decision.

8.1 In a nutshell, the functions of PyPortfolioOpt have the following benefits:

Ease of Use:

PyPortfolioOpt hides the hassle of what can be a complicated matter with its clean and well-developed API and coming standard functions for required finance expansions.

Diverse Techniques:

It supports multiple optimization approaches, such as mean variance optimization, risk parity and black-litterman.

Has flexibility to allow investor to take highest level of risks or lowest level of risks.

Customizable Inputs:

Lets users to set up risk models, expected returns and investors' point of view for optimization according to the particular objectives.

Integration with Python Ecosystem:

Is fully compatible with such libraries as Pandas, NumPy, Matplotlib, which provides the ability to operate with data intensively.

Visualization Capabilities:

The plausibility of mapping efficient frontier, portfolio allocation and evaluation metrics assists in understanding outcomes.

8.2 Limitations and Challenges

Data Quality:

Historical data which is integrated into the computation of expected returns and risks can be misleading and give wrong indications of the future outcomes.

Currently, inaccurate combinations stem from omitted or variable data that translates to unsound optimizations.

Assumptions of Market Models:

Modern Portfolio Theory considers markets efficient and return distributions to be normally distributed that is not entirely accurate in practice.

Optimization solutions are dependent on inputs, including expected returns, as well as the covariance matrices, which can be difficult to estimate.

Overfitting Risks:

Methods that are optimized strictly according to the statistics are likely to over fit, bad showing a out of sample performance.

Computational Complexity:

High dimensionality, the situation relating to large datasets or more assets managed in a portfolio, are a computational burden.

8.3 Future Scope

Advanced Optimization Techniques:

Robust Optimization: How to increase a model's reliability by admitting uncertainty in the parameters used in the process.

Stochastic Programming: By applying the scenario analysis to the multi-period portfolio optimization.

Integration with Machine Learning:

Concluding, the presented examples of applying ML models: for predicting the returns on assets or identifying the tendencies in financial data.

Outsourcing strategies that involve constantly changing in response to the current market conditions in real time.

Sustainability and ESG Factors:

The process of incorporating social, environmental and governance factors in works for the purpose of asserting ethical investment objectives.

Global Asset Allocation:

Thus, the appendix might include examples of multiple-currency portfolios and International investments hedging.

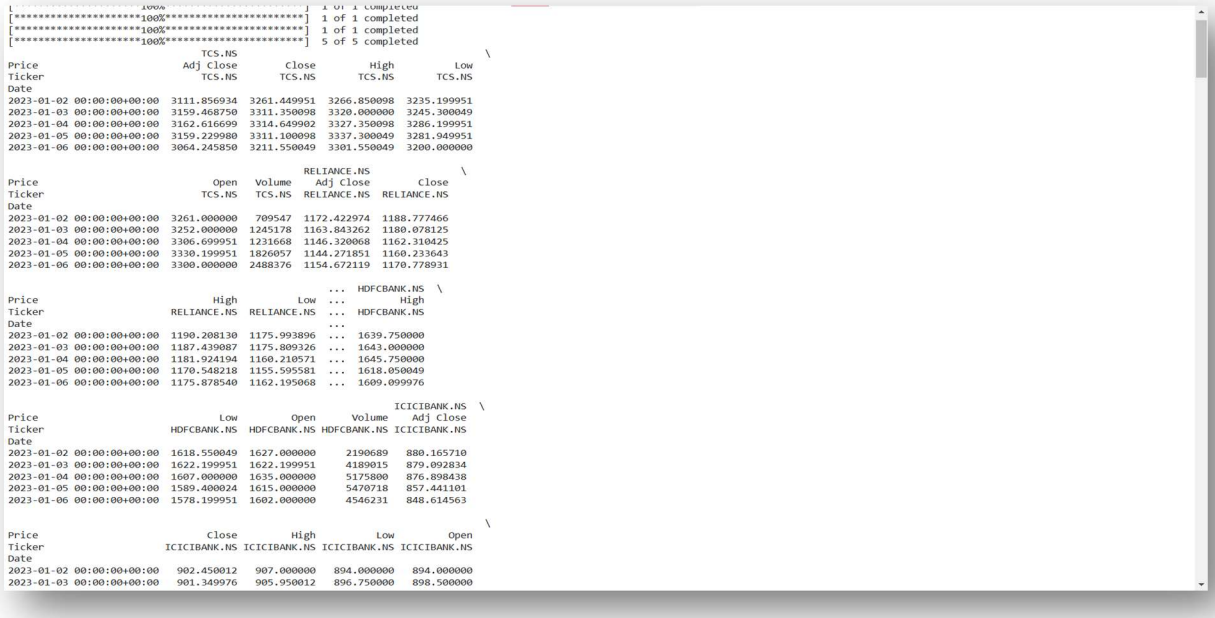
Real-Time Applications:

Indeed, incorporating portfolio optimization into adaptable algorithms for real-time portfolio rebalancing.

Closing Thoughts

With tools to design data driven, optimized portfolios, PyPortfolioOpt offers investors a powerful tool. Limitations apart, it acts as a starting point for developing new techniques based on more advanced approaches, such as machine learning and alternate data sources. With financial markets becoming more evolved with time, tools like PyPortfolioOpt will be of a immense help in defining the modern investment strategies.

UI INTERFACE



```
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  5 of 5 completed
                                TCS.NS                                    \
Price                        Adj Close       Close        High         Low
Ticker                         TCS.NS       TCS.NS      TCS.NS      TCS.NS
Date
2023-01-02 00:00:00+00:00  3111.856934  3261.449951  3266.850098  3235.199951
2023-01-03 00:00:00+00:00  3159.468750  3311.350098  3320.000000  3245.300049
2023-01-04 00:00:00+00:00  3162.616699  3314.649902  3327.350098  3286.199951
2023-01-05 00:00:00+00:00  3159.229980  3311.100098  3337.300049  3281.949951
2023-01-06 00:00:00+00:00  3064.245850  3211.550049  3301.550049  3200.000000

                                      RELIANCE.NS                          \
Price                           Open    Volume    Adj Close        Close
Ticker                         TCS.NS    TCS.NS  RELIANCE.NS  RELIANCE.NS
Date
2023-01-02 00:00:00+00:00  3261.000000   709547  1172.422974  1188.777466
2023-01-03 00:00:00+00:00  3252.000000  1245178  1163.843262  1180.078125
2023-01-04 00:00:00+00:00  3306.699951  1231668  1146.320068  1162.310425
2023-01-05 00:00:00+00:00  3330.199951  1826057  1144.271851  1160.233643
2023-01-06 00:00:00+00:00  3300.000000  2488376  1154.672119  1170.778931

                                                ...  HDFCBANK.NS  \
Price                           High          Low  ...         High
Ticker                   RELIANCE.NS  RELIANCE.NS  ...  HDFCBANK.NS
Date                                              ...
2023-01-02 00:00:00+00:00  1190.208130  1175.993896  ...  1639.750000
2023-01-03 00:00:00+00:00  1187.439087  1175.809326  ...  1643.000000
2023-01-04 00:00:00+00:00  1181.924194  1160.210571  ...  1645.750000
2023-01-05 00:00:00+00:00  1170.548218  1155.595581  ...  1618.050049
2023-01-06 00:00:00+00:00  1175.878540  1162.195068  ...  1609.099976

                                                                ICICIBANK.NS  \
Price                           Low         Open      Volume      Adj Close
Ticker                   HDFCBANK.NS  HDFCBANK.NS HDFCBANK.NS ICICIBANK.NS
Date
2023-01-02 00:00:00+00:00  1618.550049  1627.000000     2190689   880.165710
2023-01-03 00:00:00+00:00  1622.199951  1622.199951     4189015   879.092834
2023-01-04 00:00:00+00:00  1607.000000  1635.000000     5175800   876.898438
2023-01-05 00:00:00+00:00  1589.400024  1615.000000     5470718   857.441101
2023-01-06 00:00:00+00:00  1578.199951  1602.000000     4546231   848.614563

                                                                          \
Price                           Close        High         Low         Open
Ticker                   ICICIBANK.NS ICICIBANK.NS ICICIBANK.NS ICICIBANK.NS
Date
2023-01-02 00:00:00+00:00   902.450012   907.000000   894.000000   894.000000
2023-01-03 00:00:00+00:00   901.349976   905.950012   896.750000   898.500000
```

Fig8.1: Chart Performance of Indian Companies Stock Market

```
Date
2023-01-02 00:00:00+00:00      4315403
2023-01-03 00:00:00+00:00      5846855
2023-01-04 00:00:00+00:00     12454582
2023-01-05 00:00:00+00:00     16547196
2023-01-06 00:00:00+00:00     14804613

[5 rows x 30 columns]
```



```
[*********************100%***********************]  5 of 5 completed
```

TCS.NS - Adjusted Close and Moving Averages

Fig8.2: Adjusted Close Price and Moving Averages

Fig8.3: Adjusted Close Price and Moving Averages of Companies

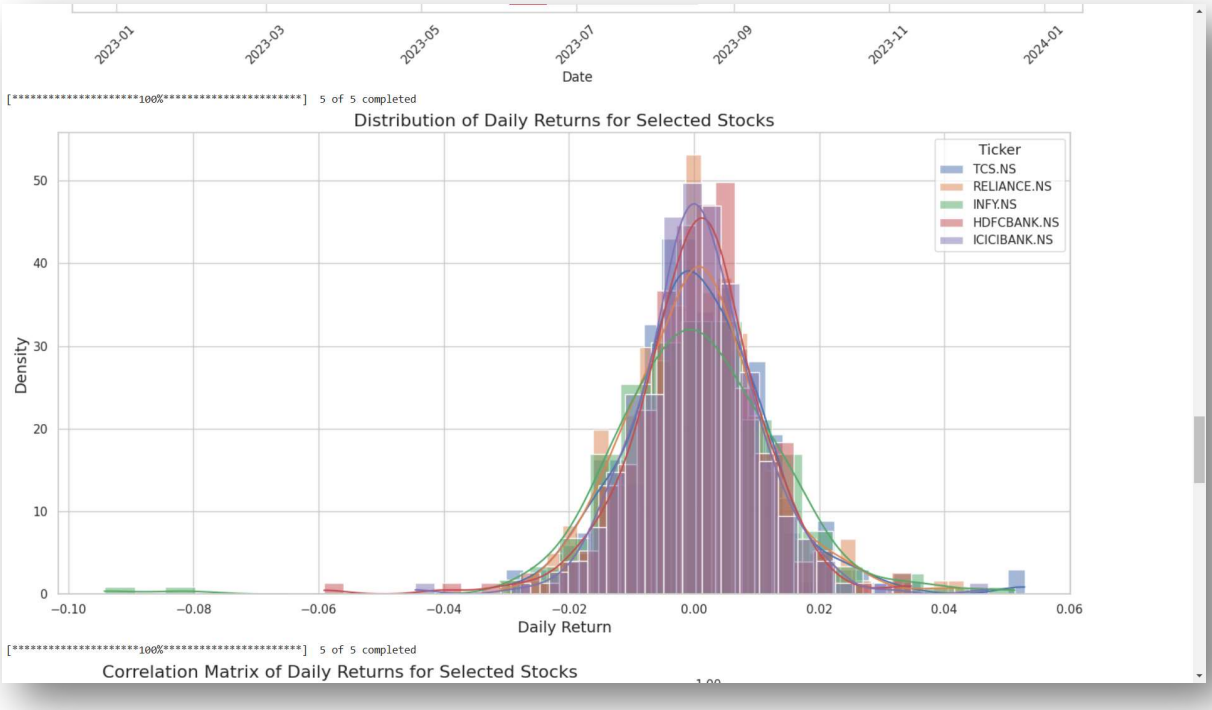


Fig8.4: Daily Returns Distribution

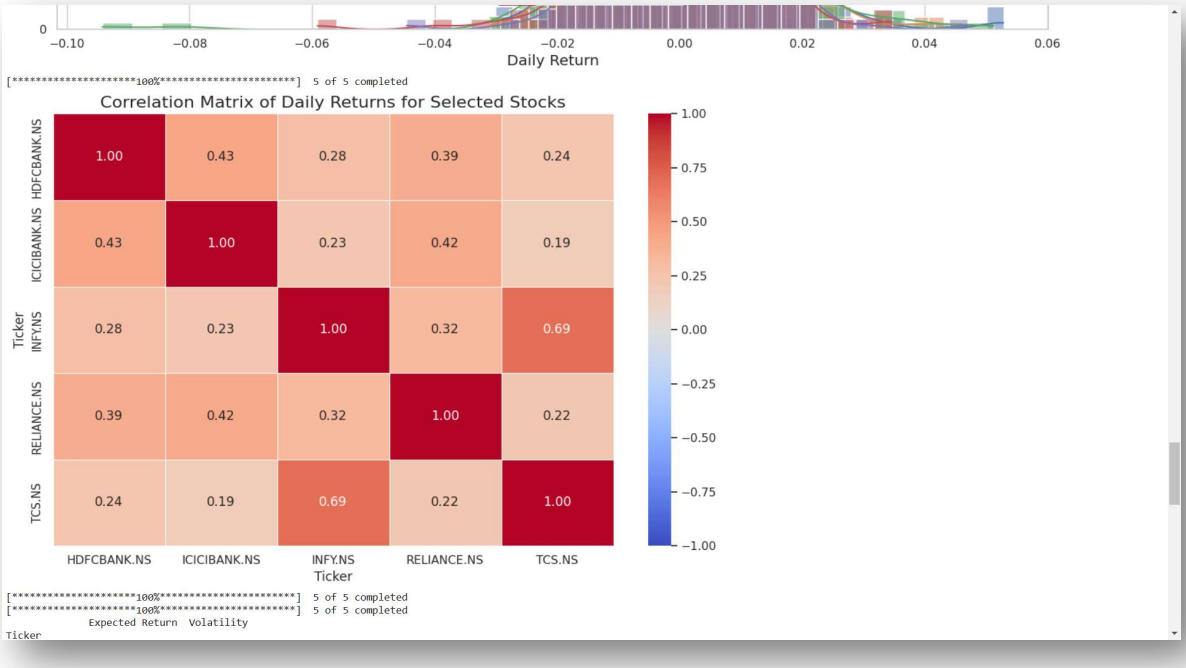


Fig8.5: Heatmap of Daily Returns Correlation Matrix

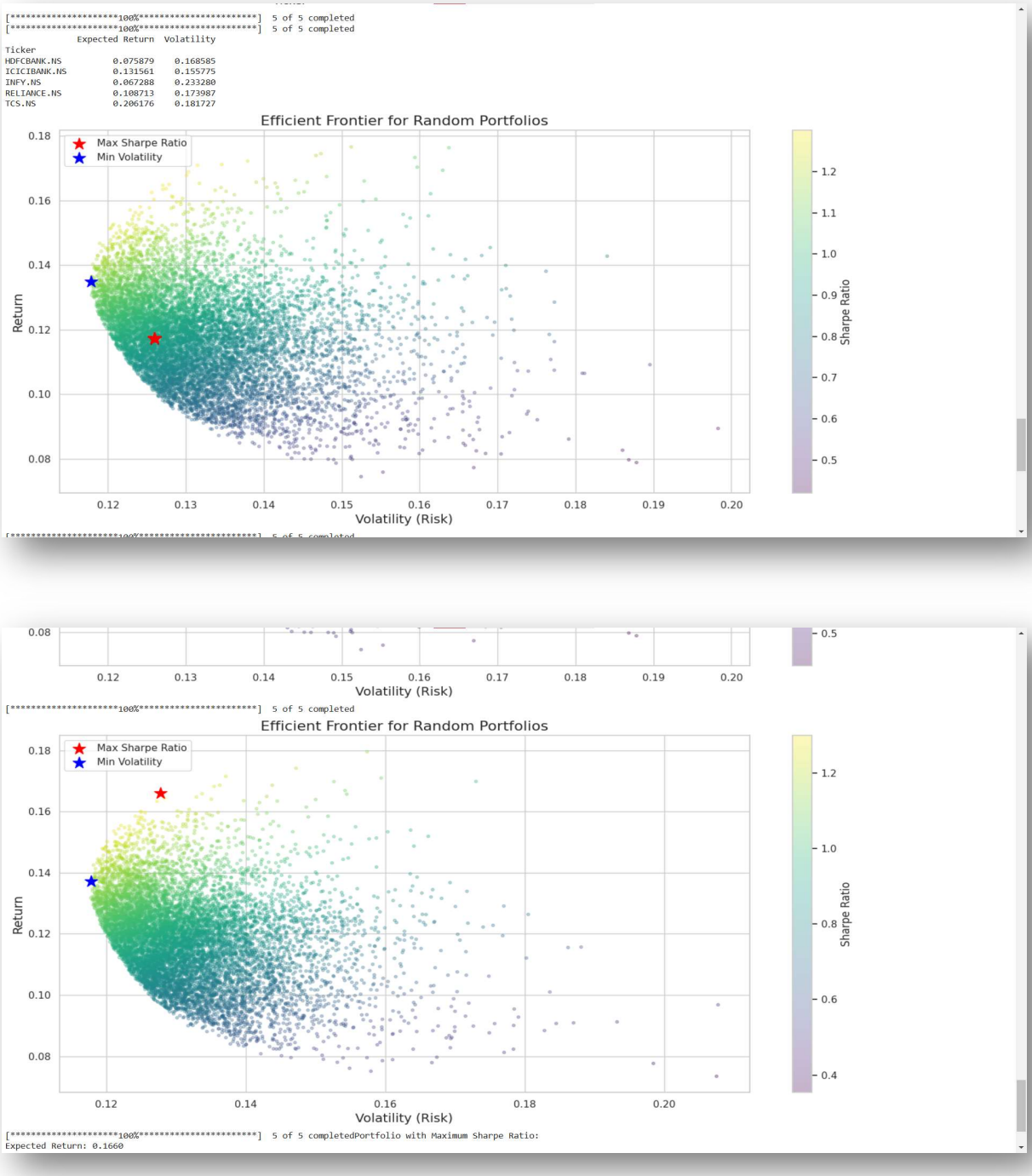


Fig8.6: Efficient Frontier

REFERENCES

[1] Markowitz, H.M. (March 1952). "Portfolio Selection". The Journal of Finance.

[2] Markowitz, *H.M. (1959)*. Portfolio Selection*:* Efficient Diversification of Investments. New York: John Wiley & Sons. (reprinted by Yale University Press)

[3] Cvitanić, Jakša; Polimenis, Vassilis*;* Zapatero, Fernando (1 January 2008). *"*Optimal portfolio allocation with higher moments". Annals of Finance. 4 (1)

[4] Kim, Young Shin*;* Giacometti, Rosella*;* Rachev, Svetlozar*;* Fabozzi, Frank J.*;* Mignacca, Domenico (21 November 2012). "Measuring financial risk and portfolio optimization with a non-Gaussian multivariate model". Annals of Operations Research. 201 (1)

[5] Merton, Robert. September 1972. "An analytic derivation of the efficient portfolio frontier," Journal of Financial and Quantitative Analysis 7.

[6] Martin, R. A. (2021). PyPortfolioOpt: portfolio optimization in Python. Journal of Open Source Software, *6*(61), 3066.

[7] Python, W. (2021). Python. Python releases for windows, *24*.

[8] De Bondt, W. F., & Thaler, R. (1985). Does the stock market overreact?. The Journal of finance, *40*(3), 793-805.

[9] Sharpe, W. F. (1994). The sharpe ratio. *Journal of portfolio management*, *21*(1), 49-58.

[10] Bodnar, T., & Schmid, W. (2009). Econometrical analysis of the sample efficient frontier. *The European journal of finance*, *15*(3), 317-335.

TURNITIN PLAGIARISM REPORT

Similarity Report

PAPER NAME

0901MC221015 Anupam Awasthi .pdf

WORD COUNT	CHARACTER COUNT
3926 Words	**22732 Characters**
PAGE COUNT	FILE SIZE
30 Pages	**2.9MB**
SUBMISSION DATE	REPORT DATE
Nov 18, 2024 3:08 PM GMT+5:30	**Nov 18, 2024 3:09 PM GMT+5:30**

10% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 3% Internet database
- 5% Publications database
- Crossref database
- Crossref Posted Content database
- 8% Submitted Works database

Excluded from Similarity Report

- Bibliographic material

MPRs (If Applicable)