**Portfolio Optimization Using Data Science**

DISSERTATION

Submitted in partial fulfillment of the requirements of the

MTech Data Science and Engineering Degree programme

By

Kasi Viswanathan SP

2021fc04293

Under the supervision of

Kavya Dharani B

Technical Lead

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

Pilani (Rajasthan) INDIA

(September, 2023)

**Acknowledgement**

I would like to express my sincere gratitude to all those who have contributed to the successful completion of my project on "Portfolio Optimization Using Data Science." This project has been an incredible journey, and I am truly thankful for the support and guidance I have received throughout.

First and foremost, I would like to extend my heartfelt thanks to my supervisor and mentor, Kavya Dharani B. Her expertise as a Technical Lead has been instrumental in shaping the direction of this project. Her continuous encouragement, insightful feedback, and unwavering support have been invaluable, and I am immensely grateful for the opportunity to learn from her.

I would also like to acknowledge Mithun Kumar, my project evaluator and examiner. His expertise and critical evaluation have provided valuable insights that have contributed to the refinement of this project. I appreciate his time and dedication to assessing the work.

I am also thankful to my peers and colleagues who provided valuable input and feedback during various stages of this project. Their perspectives and discussions greatly enriched the quality of this work.

Finally, I would like to express my gratitude to my family and friends for their encouragement and understanding during the course of this project. Their support has been a source of motivation throughout.

This project would not have been possible without the collective effort and support of all these individuals, and for that, I am truly appreciative.

Thank you all for being a part of this endeavor.

Sincerely,

Kasi Viswanathan Sp

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**

**CERTIFICATE**

This is to certify that the Dissertation entitled **Portfolio Optimization Using Data Science** and submitted by **Mr Kasi Viswanathan** SP IDNo. **2021fc04293** in partial fulfillment of the requirements of DSECLZG628T Dissertation, embodies the work done by him/her under my supervision.

 Name : Kavya Dharani B

Signature of the Supervisor Designation : Technical Lead

Place : Bangalore

Date : 23/09/2023

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI**

**SECOND SEMESTER 2022-23**

DSECLZG628T **DISSERTATION**

Dissertation Title : Portfolio Optimization Using Data Science.

Name of Supervisor : Kavya Dharani B

Name of Student : Kasi Viswanathan SP

ID No. of Student : 2021fc04293

**Abstract**

Portfolio optimization is a critical task in the field of investment management, aiming to construct portfolios that maximize returns while minimizing risks. This project focuses on the application of various data science techniques to provide investors with optimized portfolios that align with their risk preferences, capture market dynamics, and maximize risk-adjusted returns.

The project acknowledges that traditional portfolio optimization approaches, such as Mean Variance Optimization, have limitations and may not fully capture the complexities and uncertainties of financial markets. Therefore, it seeks to enhance the portfolio optimization process by incorporating additional techniques such as Black-Litterman, Critical Line Algorithm and Time Series Analysis.

With the help of this project, investors can make more informed decisions, enhance portfolio performance, and manage risks effectively in dynamic financial markets. The scope of the project includes Technique implementation, Data acquisition and preprocessing, Comparison and evaluation, Robustness analysis and Performance visualization.

The resources needed for this project are financial data which I have planned to fetch from yahoo finance, computation power for which colab will help and will use the necessary python libraries. The main potential risks in this project is accuracy. The accuracy goes for a toss when there is change in the financial system or when there is a news that affects a stock / sector / the entire market . Portfolio Optimization is fairly a new topic for the Indian stock market.

The project begins with Mean Variance Optimization, a foundational technique in portfolio optimization. Mean Variance Optimization balances the trade-off between expected returns and risks by finding the optimal portfolio allocation. It utilizes historical data to estimate expected returns and covariance matrices of assets, considering the risk-return relationship. By incorporating Mean Variance Optimization, the project provides a starting point for efficient portfolio construction.

To further improve the portfolio optimization process, the project introduces the Black-Litterman model. Black-Litterman enables the incorporation of investor views and addresses the limitations of Mean Variance Optimization. As the user will have access to data like broker calls, sectoral rotation and macro situations. By blending investor views with market equilibrium returns, the model provides a more accurate estimate of expected returns and enhances portfolio allocations.

The Critical Line Algorithm is then introduced as an efficient and robust method to solve the quadratic optimization problem associated with portfolio optimization. This technique ensures that the optimal portfolio allocation lies on the efficient frontier, maximizing risk-adjusted returns. The Critical Line Algorithm overcomes computational challenges and provides an optimized portfolio allocation.

In this project, we leverage the Autoregressive Integrated Moving Average (ARIMA) model, a widely recognized time series forecasting method, to analyze and predict stock prices.The results of this analysis provide insights into the effectiveness of the ARIMA model in forecasting stock prices. Additionally, we discuss the limitations and challenges encountered during the modeling process and explore potential avenues for future research in improving prediction accuracy and robustness.

**List of Symbols & Abbreviations used**

ARIMA - Autoregressive Integrated Moving Average

UI - User Interface

CPU - Central Processing Unit

GPU - Graphics Processing Unit

ACF - Auto Correlation Function

PACF - Partial Auto Correlation Function

CLA - Critical Line Algorithm

**List of Tables**

5.7.2 Mean Variance Optimization Output - sample stocks data……..…………..………29

5.7.3 Mean Variance Optimization Output - forecasted sample stocks data…...…31

5.8.2 Final portfolio construction……………………..……………………………………………………………34

**List of Figures**

5.2.1 Import data line chart…………………………………………………………………………………………15

5.4.1 Chart on daily returns…………………………………………………………………………………………17

5.7.1 Effective Frontier Chart………………………………………………………………………………………26

5.8.1 Final Portfolio Pie Chart………………………………………………………………………………………33

**Table of contents**

[**1.Introduction 10**](#_3qbqb8miasbo)

[**2.Problem Statement 10**](#_ubxdmb6xjliu)

[2.1.Suboptimal Returns: 10](#_411nvs4nqqs9)

[2.2.Higher Risk Exposure: 10](#_lfp3t08e6xi5)

[2.3.Emotional Decision-Making: 11](#_vvi6298kyda7)

[2.4.Difficulty in Risk Management: 11](#_i5emnw48ulud)

[2.5.Over- or Under-Investment: 11](#_vjawpj9l5hy3)

[2.6.Inadequate Data Utilization: 11](#_ekfvwz4xuccy)

[2.7.Difficulty in Monitoring: 11](#_v2rd0xqgths)

[**3.Proposed Solution 11**](#_4tf0fp8m6tjt)

[3.1.Mean Optimization Theory Portfolio 12](#_abu4mzumqvpw)

[3.2.Time Series Analysis (ARIMA Model) Portfolio: 12](#_iwsku0p4e1mg)

[3.3.User-Intuition Portfolio: 12](#_mlh1n2pkq5r6)

[3.4.Combining Portfolios Based on User Weightages: 12](#_sejwfdp7yoj0)

[**4. Requirements and Specification 13**](#_jtt3dt3t5dm6)

[**5. Design and Implementation 14**](#_228nihtbrb67)

[5.1. Import data and pre-processing 14](#_nt9lmh4wekq7)

[5.2 Import Data Visualization 15](#_wvq7kl7yuep5)

[5.3. Compute stock returns 16](#_5tp17dysbr3b)

[5.4. Visualize Stock Returns Data 16](#_dgrh97ycuvki)

[5.5. Forecast Stock returns using ARIMA Model 17](#_80vhkq6q92fd)

[5.6. Calculate Mean and Variance-covariance Matrix 19](#_57yb731atxlp)

[5.7. Mean Variance Optimization 20](#_btogxmeqxtm2)

[5.8. Final Portfolio Construction 31](#_afilnxz7a5xu)

[**6.Conclusions / Recommendations 35**](#_rit9uar25p)

[**7.Directions for future work 35**](#_ljui397omlb)

[**8.Bibliography / References 37**](#_h68150tflgc1)

[**9.Appendix 38**](#_6raozj824uxa)

# 1.Introduction

In the world of finance and investment, the pursuit of optimal returns while managing risk is a perpetual challenge. The Portfolio Optimization Problem lies at the heart of this challenge, aiming to find the most efficient allocation of assets in a portfolio to achieve a desired balance between risk and return. This problem has been a focal point for investors, analysts, and financial experts for decades, and its importance remains undiminished in today's complex and dynamic financial markets.

At its core, portfolio optimization seeks to answer a fundamental question: How should one allocate their investments across different assets or securities to maximize returns while minimizing risk, or alternatively, to achieve a specific level of return while minimizing risk exposure? This question encapsulates the essence of rational investment decision-making, as it involves the careful consideration of various factors, including historical asset performance, volatility, correlations among assets, and individual risk preferences.

# 2.Problem Statement

The following are the list problems where Portfolio Optimization can play a vital role.

### 2.1.Suboptimal Returns:

Without a systematic approach to portfolio allocation, investors may miss out on opportunities to maximize their returns. A lack of optimization may result in an inefficient allocation of resources across assets, leading to lower-than-potential investment gains.

### 2.2.Higher Risk Exposure:

Failing to diversify effectively can result in concentrated risk in the portfolio. This increases vulnerability to market volatility and the potential for significant losses. A lack of risk management strategies can be particularly detrimental during market downturns.

### 2.3.Emotional Decision-Making:

Investors may make impulsive decisions based on emotions, market noise, or short-term trends, rather than following a disciplined, data-driven approach. This can lead to suboptimal investment choices and erratic portfolio changes.

### 2.4.Difficulty in Risk Management:

Effective risk management is essential in investment. Without optimization, investors may struggle to implement strategies to protect against downside risk, potentially leading to significant losses during market downturns.

### 2.5.Over- or Under-Investment:

Investors may allocate too much or too little capital to specific assets or asset classes, resulting in inefficient use of funds or missed opportunities.

### 2.6.Inadequate Data Utilization:

Investors may not effectively leverage historical data, economic indicators, and financial analysis to inform their investment decisions. This can result in a lack of informed decision-making.

### 2.7.Difficulty in Monitoring:

Without portfolio optimization, it can be challenging to continuously monitor and adjust the portfolio to adapt to changing market conditions and investor needs.

# 3.Proposed Solution

In my project, I've taken three different approaches to create investment portfolios, and I've allowed users to influence the final portfolio by assigning weightages to these approaches.

### 3.1.Mean Optimization Theory Portfolio

In this approach, you have applied mean optimization theory, which typically involves optimizing the portfolio to maximize expected returns while considering risk (usually measured as variance or standard deviation).The mean optimization theory portfolio is generated using quantitative techniques that aim to find the ideal allocation of assets to maximize returns relative to the level of risk an investor is willing to tolerate.This portfolio is driven by statistical analysis and historical data to create an efficient portfolio allocation.

### 3.2.Time Series Analysis (ARIMA Model) Portfolio:

In this approach, you have leveraged time series analysis, specifically the ARIMA (AutoRegressive Integrated Moving Average) model, to forecast the future performance of assets. The ARIMA model is used to make predictions about the future prices or returns of assets based on historical time series data. This portfolio is generated by considering the forecasted performance of each asset, taking into account potential future trends and volatility.

### 3.3.User-Intuition Portfolio:

This approach involves incorporating the subjective input of the user or investor into the portfolio generation process.Users provide their insights, preferences, and intuitions about how the portfolio should be structured. This input could be based on factors like personal beliefs, market sentiment,macro sentiments, broker calls, upcoming government policies or specific investment goals.The user-intuition portfolio reflects the qualitative judgments and opinions of the investor.

### 3.4.Combining Portfolios Based on User Weightages:

To accommodate different investment philosophies and preferences, you allow users to assign weightages to each of the three portfolios generated through the mean optimization theory, time series analysis, and user intuition.The weightages determine the relative importance of each portfolio in the final combined portfolio.The combined portfolio is created by blending the three portfolios according to the specified weightages. This approach aims to strike a balance between the quantitative and qualitative aspects of portfolio management.

By combining these three diverse approaches based on user-defined weightages, your project offers a comprehensive solution that accommodates both data-driven quantitative analysis and the subjective input of investors. This approach allows for flexibility and customization in portfolio construction, catering to a wide range of investor preferences and risk tolerance levels.

# 4. Requirements and Specification

I'm using Yahoo Finance to grab all the data I need. That means I can access historical financial data for the assets I'm interested in.

Now, before I can dive into the analysis, I've got to get that data in tip-top shape. That's where data preprocessing comes in. I'm cleaning it up, making sure it's structured just right, and handling any missing or wonky data.

For the heavy lifting in terms of computing power, I'm using Google Colab. It's a great platform that gives me the internet connection I need for data retrieval and all the computational muscle for my analysis. It's got the CPU, memory, and even GPU if I need it.

As for the Python side of things, I've got a few trusty libraries in my toolkit:

* pandas for data manipulation
* matplotlib to create those slick data visualizations
* numpy for crunching numbers
* statsmodels.tsa.arima.model for the ARIMA model in my time series analysis
* And of course, scipy for some scientific and statistical firepower.

# 

# 

# 5. Design and Implementation

## 5.1. Import data and pre-processing

Here, we have imported the necessary data from yahoo finance and we have used .bfill() method on the dataframe to backward-fill missing values. This means that any missing value will be filled with the last available value in the respective column. This helps ensure that your time series data is continuous and suitable for analysis.

pip install yfinance

**import** yfinance **as** yf  
**import** pandas **as** pd  
**from** google.colab **import** files  
  
*# List of stock symbols*  
sample\_stocks = [  
 "HEROMOTOCO.NS", "TITAN.NS"  
]  
  
*# Define the time range*  
start\_date = "2006-01-01"  
end\_date = "2023-01-01"  
  
*# Fetch historical data*  
data = yf.download(sample\_stocks, start=start\_date, end=end\_date)['Close']  
  
*# Backward-fill missing values with the last available value*  
data.bfill(inplace=True)  
  
*# Save the data to a CSV file*  
data.to\_csv('/content/samplestocks\_closing\_prices\_bfill.csv')  
  
*# Download the file within Colab*  
*#files.download('/content/samplestocks\_closing\_prices\_bfill.csv')*

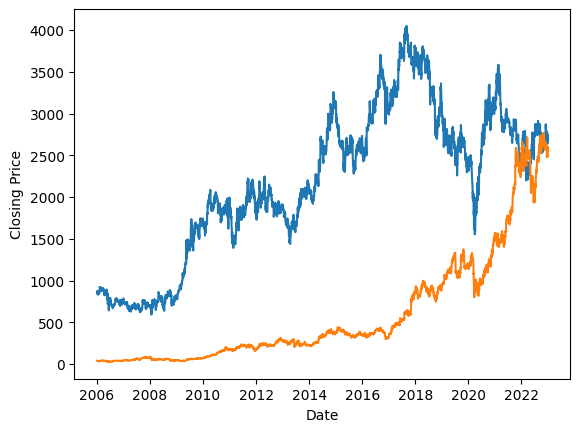
[\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*100%%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*] 2 of 2 completed

## 5.2 Import Data Visualization

Here we have plotted the imported data on a line chart using the necessary python libraries.

**import** matplotlib.pylab **as** plt  
  
plt.xlabel("Date")  
plt.ylabel("Closing Price")  
plt.plot(data)

[<matplotlib.lines.Line2D at 0x7e72c9947c10>,  
 <matplotlib.lines.Line2D at 0x7e7300d5d090>]



HeroMotoCo Titan

5.2.1 Import data line chart

data\_rows, data\_columns = data.shape

## 5.3. Compute stock returns

This method calculates the return percentage of each stock per day.

return percentage = [(closing price on day t+1) - (closing price on day t)]/[closing price on day t]

We calculated the daily return percentage here using the necessary python libraries.

*#function to compute asset returns*  
**def** StockReturnsComputing(StockPrice, Rows, Columns):  
  
 **import** numpy **as** np  
  
 StockReturn = np.zeros([Rows-1, Columns])  
 **for** j **in** range(Columns): *# j: Assets*  
 **for** i **in** range(Rows-1): *# i: Daily Prices*  
 StockReturn[i,j]=((StockPrice[i+1, j]-StockPrice[i,j])/StockPrice[i,j])\* 100  
  
 **return** StockReturn

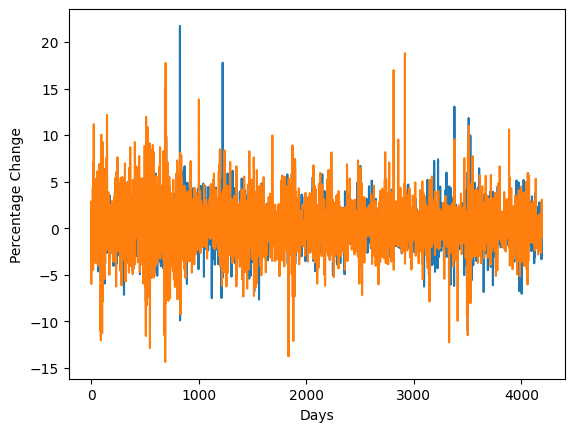
**import** numpy **as** np  
  
stockPriceArray = np.asarray(data)  
[Rows, Cols]=stockPriceArray.shape  
stockReturns = StockReturnsComputing(stockPriceArray, Rows, Cols)  
print('Daily returns of selective sample stocks\n', stockReturns)

Daily returns of selective sample stocks  
 [[ 1.79057143 -0.0299013 ]  
 [-0.53410457 2.86864544]  
 [ 1.31909035 -5.98432716]  
 ...  
 [-0.21277776 3.05964938]  
 [ 0.94933815 -1.0425713 ]  
 [ 0.61164304 1.73308528]]

## 5.4. Visualize Stock Returns Data

Here we have visualized the daily return data.

plt.xlabel("Days")  
plt.ylabel("Percentage Change")  
plt.plot(stockReturns)



HeroMotoCo Titan

5.4.1 Chart on daily returns

## 5.5. Forecast Stock returns using ARIMA Model

The Autoregressive Integrated Moving Average (ARIMA) model is a widely used and effective time series forecasting technique for predicting future stock prices. It is particularly valuable for capturing and modeling the inherent time-dependent patterns, trends, and seasonality present in financial data.

We have used daily return data for forecasting as that data would be stationary. As stationary data is suitable for the ARIMA model. Here we have forecasted the values for one year.

The ARIMA model's "AR" component accounts for the autocorrelation of a time series, which is the relationship between an observation and previous observations.In the context of stock price forecasting, this component considers how past stock prices influence future prices. It measures the linear dependence between the current price and past prices at various lags.The AR component is denoted as AR(p), where "p" represents the order of autoregression. It indicates how many previous time steps are used to predict the current value.

The "I" in ARIMA stands for "Integrated," and it represents the differencing operation applied to the time series data.Differencing involves subtracting the previous value from the current value, which helps in removing trends and making the data stationary (i.e., constant mean and variance over time). Stationary data is easier to model.The integrated component is denoted as I(d), where "d" represents the order of differencing. It signifies how many times differencing is performed to achieve stationarity.

The "MA" component of ARIMA takes into account the relationship between the current value and past prediction errors (residuals).In stock price forecasting, this component assesses how past forecast errors influence future stock price predictions.The MA component is denoted as MA(q), where "q" represents the order of the moving average. It specifies the number of lagged forecast errors considered in the model.

Selecting appropriate values for the orders "p," "d," and "q" is a crucial step in ARIMA modeling. It involves analyzing the autocorrelation function (ACF) and partial autocorrelation function (PACF) plots to identify the optimal values. As it is complex to calculate the values for each forecast, the values have been assumed to be 2 for each of the variables. The values are finalized based on generalization on the movement of stocks.

**from** statsmodels.tsa.arima.model **import** ARIMA  
  
columns\_data = {}  
  
**for** col **in** range(Cols):  
 DatatoPredict = [sublist[col] **for** sublist **in** stockReturns]  
 model = ARIMA(DatatoPredict,order=(2,2,2))  
 result\_ARIMA = model.fit()  
  
 columns\_data[col] = result\_ARIMA.forecast(steps=252)  
 *#252 number of trading days in a calendar year*  
  
DataGeneratedByTimeSeriesAnalysis = pd.DataFrame(columns\_data)  
  
*# Display the DataFrame*  
print(DataGeneratedByTimeSeriesAnalysis)

/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/statespace/sarimax.py:978: UserWarning: Non-invertible starting MA parameters found. Using zeros as starting parameters.  
 warn('Non-invertible starting MA parameters found.'

0 1  
0 -0.024830 -0.233313  
1 -0.011412 -0.173672  
2 0.023317 -0.196655  
3 0.022904 -0.193350  
4 0.021058 -0.191380  
.. ... ...  
247 0.031473 0.366844  
248 0.031515 0.369141  
249 0.031558 0.371438  
250 0.031600 0.373736  
251 0.031642 0.376033  
  
[252 rows x 2 columns]

[Generated\_Rows, Generated\_Cols]=DataGeneratedByTimeSeriesAnalysis.shape

## 5.6. Calculate Mean and Variance-covariance Matrix

The diagonal represents the variance of each stock. A[i,i] represents the variance of stock i.A[i,j] represents the covariance of stock i with stock j.

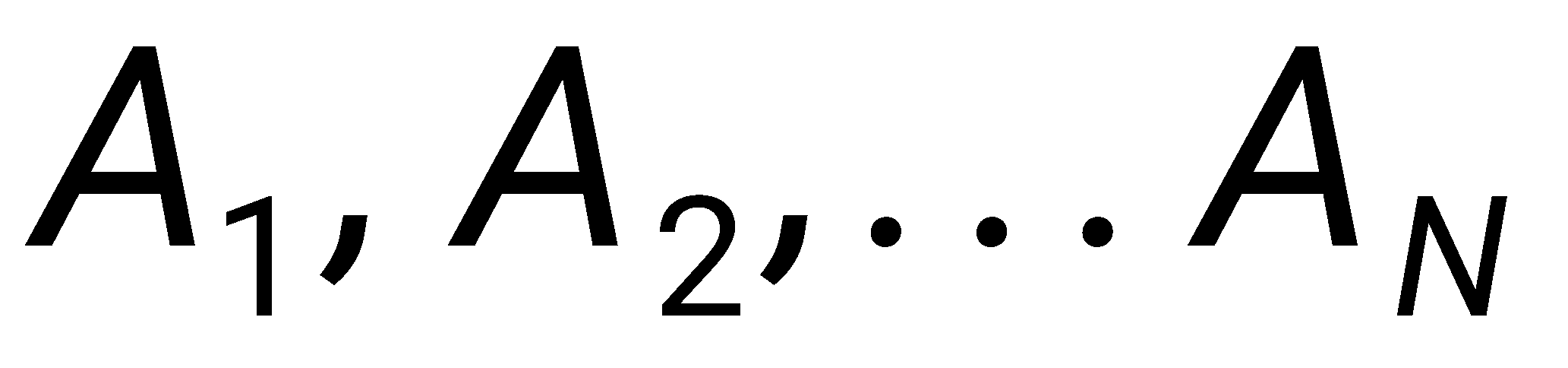
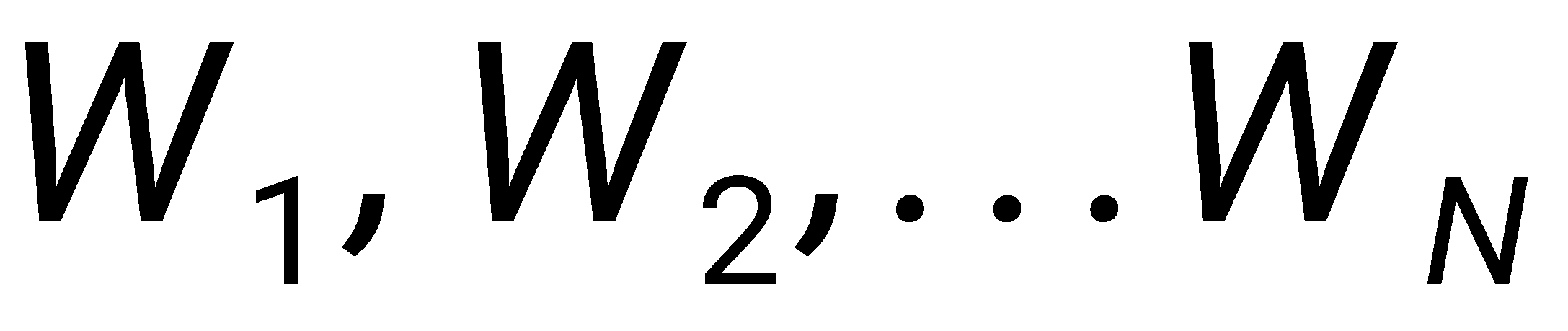
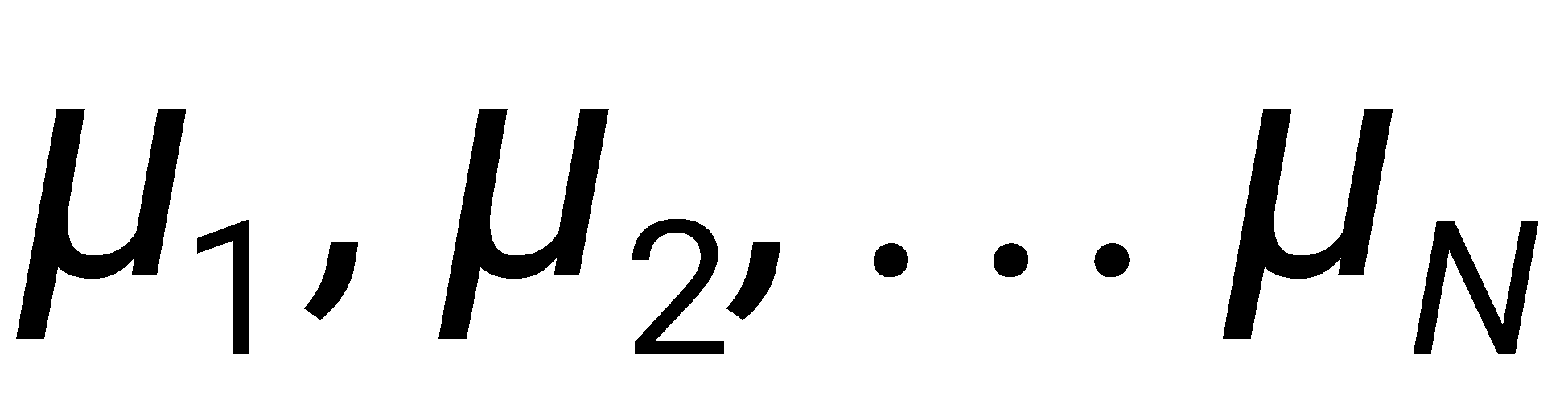
meanReturns = np.mean(stockReturns, axis = 0)  
print('Mean returns of sample Stocks:\n', meanReturns)  
covReturns = np.cov(stockReturns, rowvar=False)  
print('Variance-covariance matrix of returns of sample Stocks:\n')  
print(covReturns)  
  
GeneratedStockReturns = np.asarray(DataGeneratedByTimeSeriesAnalysis)  
GeneratedmeanReturns = np.mean(GeneratedStockReturns, axis = 0)  
print('Mean returns of forecasted data:\n', GeneratedmeanReturns)  
GeneratedcovReturns = np.cov(GeneratedStockReturns, rowvar=False)  
print('Variance-covariance matrix of returns of forecasted data:\n')  
print(GeneratedcovReturns)

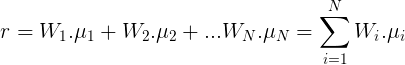
Mean returns of sample Stocks:  
 [0.04702897 0.12933651]  
Variance-covariance matrix of returns of sample Stocks:  
  
[[3.82916379 1.2252536 ]  
 [1.2252536 5.9883011 ]]  
Mean returns of forecasted data:  
 [0.02602395 0.08770405]  
Variance-covariance matrix of returns of forecasted data:  
  
[[2.52151630e-05 6.05936892e-04]  
 [6.05936892e-04 2.80634047e-02]]

The Variance-covariance matrix of returns of sample stocks state that that stock HEROMOTOCO has a variance of 3.82916379 and its covariance with TITAN is 1.2252536. And the stock TITAN has a covariance of 1.2252536 with HEROMOTOCO and its variance is 5.9883011.

Similarly, The Variance-covariance matrix of returns of forecasted data state that that stock HEROMOTOCO has a variance of 2.52151630e-05 and its covariance with TITAN is 6.05936892e-04. And the stock TITAN has a covariance of 6.05936892e-04 with HEROMOTOCO and its variance is 2.80634047e-02.

## 5.7. Mean Variance Optimization

By Mean Vairance Optimization Model,Let P be a portfolio comprising assets  with weights  and  as the asset returns. The portfolio return *r* determined by a weighted summation of its individual asset returns is given by,



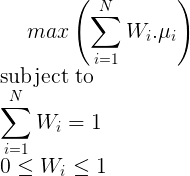
By substituting the known values for sample stocks, we get

return = 0.04702897(Weight of HEROMOTOCO ) + 0.12933651(Weight of TITAN)

Similarly, By substituting the known values for forecasted data of sample stocks, we get

return = 0.02602395(Weight of HEROMOTOCO ) + 0.08770405(Weight of TITAN)

The maximum return we can get is given by the below formula, Note: the sum of all weights should be equal to 1 and each weight should be greater than or equal to 0 and it should be less than or equal to 1.



Now, We need to calculate the max(return) satisfying the conditions mentioned above, for that we have used optimize.linprog() method in scipy, Since the problem model is linear, linprog function from the package scipy.optimize is invoked to execute linear programming to solve the problem model. the documentation for that is as follow.

SciPy is an open-source Python library dedicated to scientific computation. The optimize package in SciPy provides several common optimization algorithms such as least squares, minimization, curve fitting, maximization etc.

The optimize.linprog() function is from the domain of linear programming, which minimizes a linear objective function subject to linear equality and inequality constraints.

Parameters The function optimize.linprog() accepts the following parameters:

c: This is a one-dimensional array representing the coefficients of the linear objective function.

A\_ub: This is a two-dimensional array representing the inequality constraint matrix. Each row of the matrix represents the coefficients of a linear inequality.

b\_ub: This is a one-dimensional array representing the inequality constraint vector.

A\_eq: This is a two-dimensional array representing the equality constraint matrix. Each row of the matrix represents the coefficients of a linear equality.

b\_eq: This is a one-dimensional array representing the equality constraint vector.

bounds: This is a sequence of the (min,max) pair defining the minimum and maximum value of the decision variable.

callback: This is an optional function argument. It is invoked on every iteration.

method: This is the algorithm used to solve the standard form problem.

options: This is a dictionary of solver options.

x0: This is a one-dimensional array that represents the guess values of the decision variables.

Return value : This function returns a solution represented by the OptimizeResult object. As part of the object, the following components are returned.

x: This contains the values of the decision variables that minimize/maximize the objective function while meeting the defined constraints.

fun: This tells the optimal value of the objective function.

slack: This tells the (nominally positive) values of the slack variables. Slack variables are the differences between the values of the left and right sides of the constraints.

con: This represents the equality constraints residuals.

status: This is a value between 0-4 that represents the exit status of the algorithm.

success: This tells us that the algorithm has found an optimal solution.

nit: This tells us the total number of iterations in all phases.

message: This displays the message produced on the algorithm’s termination.

*#function obtains maximal return portfolio using linear programming*  
  
**def** MaximizeReturns(MeanReturns, PortfolioSize):  
  
 *#dependencies*  
 **from** scipy.optimize **import** linprog  
 **import** numpy **as** np  
  
 c = (np.multiply(-1, MeanReturns))  
 A = np.ones([PortfolioSize,1]).T  
 b=[1]  
 res = linprog(c, A\_ub = A, b\_ub = b, bounds = (0,1), method = 'simplex')  
  
 **return** res

*#Maximal expected portfolio return computation for the portfolio*  
result1 = MaximizeReturns(meanReturns, data\_columns)  
maxReturnWeights = result1.x  
print("The solution for the weights are")  
print(maxReturnWeights)  
maxExpPortfolioReturn = np.matmul(meanReturns.T, maxReturnWeights)  
print("Maximal Expected Portfolio Return: %7.4f" % maxExpPortfolioReturn )  
  
Generatedresult1 = MaximizeReturns(GeneratedmeanReturns, Generated\_Cols)  
GeneratedmaxReturnWeights = Generatedresult1.x  
print("The solution for the weights are")  
print(GeneratedmaxReturnWeights)  
GeneratedmaxExpPortfolioReturn = np.matmul(GeneratedmeanReturns.T, GeneratedmaxReturnWeights)  
print("Maximal Expected Portfolio Return: %7.4f" % GeneratedmaxExpPortfolioReturn )

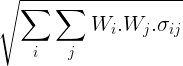
The solution for the weights are  
[0. 1.]  
Maximal Expected Portfolio Return: 0.1293  
The solution for the weights are  
[0. 1.]  
Maximal Expected Portfolio Return: 0.0877

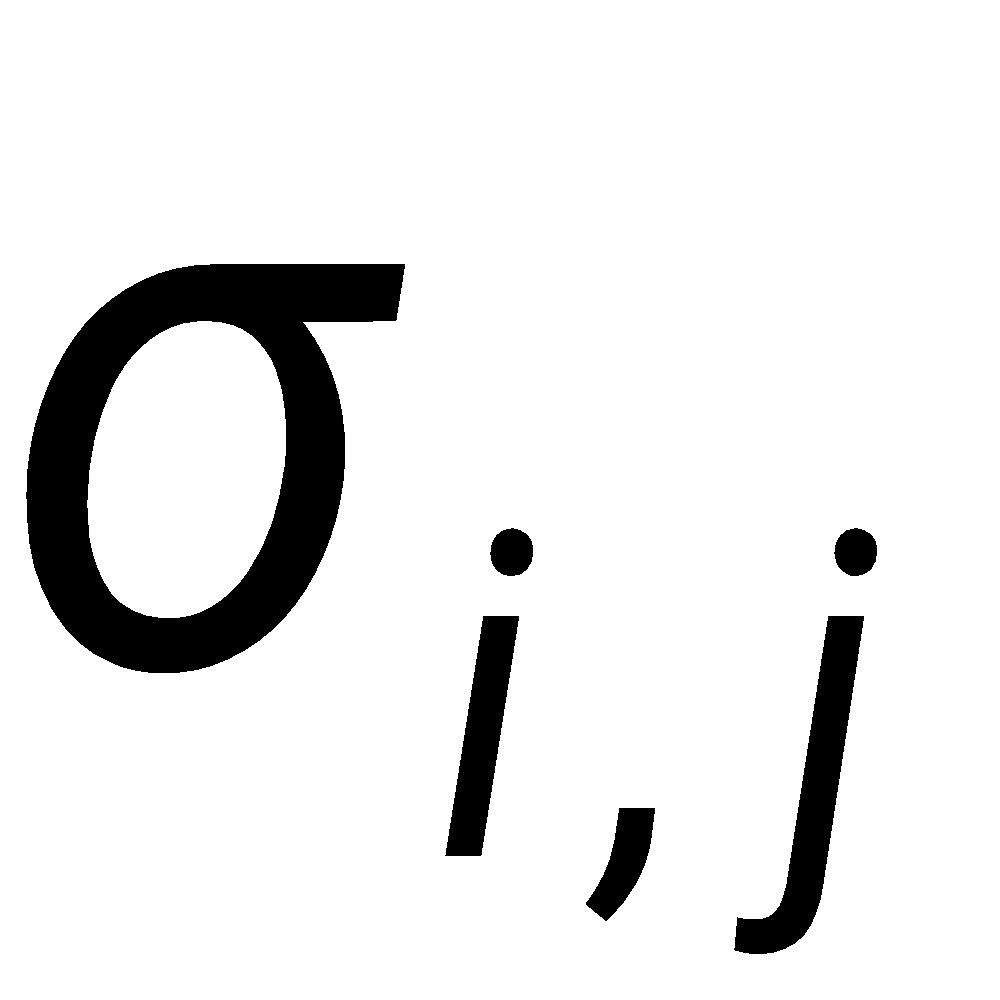
By solving the return equation of sample stocks, we get the expected maximum return on one day = 0.1293% and it is obtained by Giving a weight of 0 to HEROMOTOCO and 0 to TITAN

Similarly, By solving the return equation of forecasted data of sample stocks, we get the expected maximum return on one day = 0.0877% and it is obtained by Giving a weight of 0 to HEROMOTOCO and 0 to TITAN

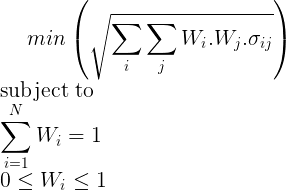
Now, let us calculate the returns we can get by taking minimum risk.

By Mean Variance Optimization Model,Portfolio risk is the standard deviation of its returns and is given by,



where  is the variance-covariance matrix of returns.

To get the returns we can get by taking minimal risk, we have to calculate the below expression and it should satisfy the following conditions.



The Python function MinimizeRisk employs scipy.optimize and NumPy to obtain the optimal weights.

Functions and constraints describe the non-linear objective function and the fully invested constraint described above, respectively. optimize.minimize function executes minimization of scalar functions with one or more variables.

*#function obtains minimal risk portfolio*  
  
*#dependencies*  
**import** numpy **as** np  
**from** scipy **import** optimize  
  
**def** MinimizeRisk(CovarReturns, PortfolioSize):  
  
 **def** f(x, CovarReturns):  
 func = np.matmul(np.matmul(x, CovarReturns), x.T)  
 **return** func  
  
 **def** constraintEq(x):  
 A=np.ones(x.shape)  
 b=1  
 constraintVal = np.matmul(A,x.T)-b  
 **return** constraintVal  
  
 xinit=np.repeat(0.1, PortfolioSize)  
 cons = ({'type': 'eq', 'fun':constraintEq})  
 lb = 0  
 ub = 1  
 bnds = tuple([(lb,ub) **for** x **in** xinit])  
  
 opt = optimize.minimize (f, x0 = xinit, args = (CovarReturns), bounds = bnds, \  
 constraints = cons, tol = 10\*\*-3)  
  
 **return** opt

result2 = MinimizeRisk(covReturns, data\_columns)  
minRiskWeights = result2.x  
print("The solution for the weights are")  
print(minRiskWeights)  
minRiskExpPortfolioReturn = np.matmul(meanReturns.T, minRiskWeights)  
print("Expected Return of Minimum Risk Portfolio: %7.4f" % minRiskExpPortfolioReturn)  
  
Generatedresult2 = MinimizeRisk(GeneratedcovReturns, Generated\_Cols)  
GeneratedminRiskWeights = Generatedresult2.x  
print("The solution for the weights are")  
print(GeneratedminRiskWeights)  
GeneratedminRiskExpPortfolioReturn = np.matmul(GeneratedmeanReturns.T, GeneratedminRiskWeights)  
print("Expected Return of Minimum Risk Portfolio: %7.4f" % GeneratedminRiskExpPortfolioReturn)

The solution for the weights are  
[0.64691982 0.35308018]  
Expected Return of Minimum Risk Portfolio: 0.0761  
The solution for the weights are  
[0.50280382 0.49719618]  
Expected Return of Minimum Risk Portfolio: 0.0567

By substituting the known values and solving the sample stocks equation, we get that the expected return we can get from the minimum risk portfolio on one day = 0.0761%. And it can we obtained by giving a weight of 0.64691982 to HEROMOTOCOand 0.35308018 to TITAN

Similarly , By substituting the known values and solving the forecasted data equation, we get that the expected return we can get from the minimum risk portfolio on one day = 0.0567%. And it can we obtained by giving a weight of 0.50280382 to HEROMOTOCOand 0.49719618 to TITAN

In terms of risk and return, the general saying is that more risk -> more return / more return -> more risk and low risk -> low return / low return -> low risk. But what everyone wants is low risk and high return.

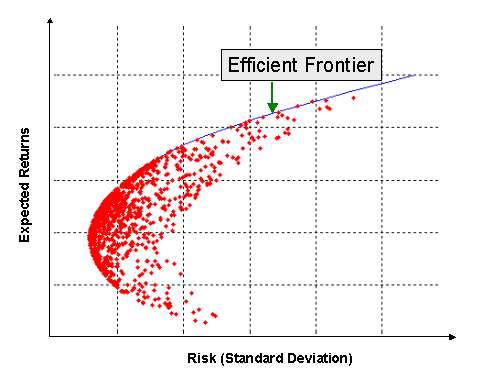
Using Mean Variance Optimization Model, we have calculated the expected highest return for that portfolio. and we have calculated the expected return for that portfolio with lowest risk.

Note: We have not calculated the expected return with lowest return as practically no one will be investing with the idea of generating lowest return. The Mean Variance Optimization suggests that lowest risk return > lowest return possible.

By Mean Variance Optimization Model, we have the min and max range for return.

In investing, we can make a return of say 15% by taking both high risk and low risk. The weightage of the stocks in the portfolio differs according to the risk. Making 15% by taking low risk is always better than doing the same with high risk as the reward is the same in both the cases, and taking more risk increases the downside potential.

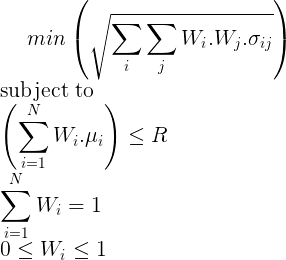
Now, as we have the range for return, we will be incrementing it from its low value to the high value by a small number and we will be calculating the optimal weightage so that we take the lowest risk for that each of the return in that range using the following expression suggested by Mean Variance Optimization Model.



5.7.1 Effective Frontier Chart

Each red dot in the above figure represents a portfolio combination obtained by Mean Variance Optimization and CLA(Critical Line Algorithm) states that all the dots that are not on the curve are ineffective. Let us assume that there are two points on the same y axis(return is same). Let point 1 be inside the Effective Frontier arc and point 2 be on the arc. Then point 1 is ineffective because we could generate the same return by taking minimum risk which is point 2 which is effective.

The mathematical model for this sub problem is defined as follows, where for each R, R(min\_risk)<=R<=R(max\_return). Where R is the return. The problem is repeatedly solved to arrive at the optimal weight set, each of which determines a portfolio with minimum risk and maximum return.



The function MinimizeRiskConstr employs scipy and Numpy to obtain the optimal weight set. Function optimize.minimize used the Trust-region Constrained algorithm to solve the constrained optimization problem with both equality and inequality constraints.

*#function obtains Minimal risk and Maximum return portfolios*  
  
*#dependencies*  
**import** numpy **as** np  
**from** scipy **import** optimize  
  
**def** MinimizeRiskConstr(MeanReturns, CovarReturns, PortfolioSize, R):  
  
 **def** f(x,CovarReturns):  
  
 func = np.matmul(np.matmul(x,CovarReturns ), x.T)  
 **return** func  
  
 **def** constraintEq(x):  
 AEq=np.ones(x.shape)  
 bEq=1  
 EqconstraintVal = np.matmul(AEq,x.T)-bEq  
 **return** EqconstraintVal  
  
 **def** constraintIneq(x, MeanReturns, R):  
 AIneq = np.array(MeanReturns)  
 bIneq = R  
 IneqconstraintVal = np.matmul(AIneq,x.T) - bIneq  
 **return** IneqconstraintVal  
  
  
 xinit=np.repeat(0.1, PortfolioSize)  
 cons = ({'type': 'eq', 'fun':constraintEq},  
 {'type':'ineq', 'fun':constraintIneq, 'args':(MeanReturns,R) })  
 lb = 0  
 ub = 1  
 bnds = tuple([(lb,ub) **for** x **in** xinit])  
  
 opt = optimize.minimize (f, args = (CovarReturns), method ='trust-constr', \  
 x0 = xinit, bounds = bnds, constraints = cons, tol = 10\*\*-3)  
  
 **return** opt

*#compute efficient set for the maximum return and minimum risk portfolios*  
*#increment = 0.001*  
increment = 0.01  
low = minRiskExpPortfolioReturn  
high = maxExpPortfolioReturn  
  
*#initialize optimal weight set and risk-return point set*  
xOptimal =[]  
minRiskPoint = []  
expPortfolioReturnPoint =[]  
  
*#repeated execution of function MinimizeRiskConstr to determine the efficient set*  
**while** (low < high):  
  
 result3 = MinimizeRiskConstr(meanReturns, covReturns, data\_columns, low)  
 xOptimal.append(result3.x)  
 expPortfolioReturnPoint.append(low)  
 low = low+increment  
  
*#gather optimal weight set*  
xOptimalArray = np.array(xOptimal)  
  
*#obtain annualized risk for the efficient set portfolios*  
*#for trading days = 252*  
minRiskPoint = np.diagonal(np.matmul((np.matmul(xOptimalArray,covReturns)),\  
 np.transpose(xOptimalArray)))  
riskPoint = np.sqrt(minRiskPoint\*252)  
  
*#obtain expected portfolio annualized return for the*  
*#efficient set portfolios, for trading days = 252*  
retPoint = 252\*np.array(expPortfolioReturnPoint)  
  
*#display efficient set portfolio parameters*  
print("Size of the efficient set:", xOptimalArray.shape )  
print("Optimal weights of the efficient set portfolios: \n", xOptimalArray)  
print("Annualized Risk and Return of the efficient set portfolios: \n", \  
 np.c\_[riskPoint, retPoint])

Size of the efficient set: (6, 2)  
Optimal weights of the efficient set portfolios:   
 [[0.6418583 0.35804326]  
 [0.52526586 0.47473414]  
 [0.40361994 0.59638006]  
 [0.28156822 0.71843178]  
 [0.16085484 0.83914516]  
 [0.04632925 0.95368317]]  
Annualized Risk and Return of the efficient set portfolios:   
 [[27.07232186 19.17471351]  
 [27.57390304 21.69471351]  
 [29.02701981 24.21471351]  
 [31.30987737 26.73471351]  
 [34.21903759 29.25471351]  
 [37.4413653 31.77471351]]

The above results obtained by the sample stocks data imply the follow

| Annual Risk | Annual Return | HEROMOTOCO weightage | TITAN weightage |
| --- | --- | --- | --- |
| 27.07232186% | 19.17471351% | 64.18583% | 35.804326% |
| 27.57390304% | 21.69471351% | 52.526586% | 47.473414% |
| 29.02701981% | 24.21471351% | 40.361994% | 59.638006% |
| 31.30987737% | 26.73471351% | 28.156822% | 71.843178% |
| 34.21903759% | 29.25471351% | 16.085484% | 83.914516% |
| 37.4413653% | 31.77471351% | 4.632925% | 95.368317% |

5.7.2 Mean Variance Optimization Output - sample stocks data

*#compute efficient set for the maximum return and minimum risk portfolios*  
*#increment = 0.001*  
increment = 0.01  
low = GeneratedminRiskExpPortfolioReturn  
high = GeneratedmaxExpPortfolioReturn  
  
*#initialize optimal weight set and risk-return point set*  
GeneratedxOptimal =[]  
GeneratedminRiskPoint = []  
GeneratedexpPortfolioReturnPoint =[]  
  
*#repeated execution of function MinimizeRiskConstr to determine the efficient set*  
**while** (low < high):  
  
 Generatedresult3 = MinimizeRiskConstr(GeneratedmeanReturns, GeneratedcovReturns, Generated\_Cols, low)  
 GeneratedxOptimal.append(Generatedresult3.x)  
 GeneratedexpPortfolioReturnPoint.append(low)  
 low = low+increment  
  
*#gather optimal weight set*  
GeneratedxOptimalArray = np.array(GeneratedxOptimal)  
  
*#obtain annualized risk for the efficient set portfolios*  
*#for trading days = 252*  
GeneratedminRiskPoint = np.diagonal(np.matmul((np.matmul(GeneratedxOptimalArray,GeneratedcovReturns)),\  
 np.transpose(GeneratedxOptimalArray)))  
GeneratedriskPoint = np.sqrt(GeneratedminRiskPoint\*252)  
  
*#obtain expected portfolio annualized return for the*  
*#efficient set portfolios, for trading days = 252*  
GeneratedretPoint = 252\*np.array(GeneratedexpPortfolioReturnPoint)  
  
*#display efficient set portfolio parameters*  
print("Size of the efficient set:", GeneratedxOptimalArray.shape )  
print("Optimal weights of the efficient set portfolios: \n", GeneratedxOptimalArray)  
print("Annualized Risk and Return of the efficient set portfolios: \n", \  
 np.c\_[GeneratedriskPoint, GeneratedretPoint])

Size of the efficient set: (4, 2)  
Optimal weights of the efficient set portfolios:   
 [[0.41158594 0.58822747]  
 [0.28731898 0.71268102]  
 [0.12923434 0.87076566]  
 [0.01270392 0.98729385]]  
Annualized Risk and Return of the efficient set portfolios:   
 [[ 1.58808057 14.28614728]  
 [ 1.91180988 16.80614728]  
 [ 2.32307526 19.32614728]  
 [ 2.62625893 21.84614728]]

Considering 252 trading days in a year, we have calculated the one year risk , return and the corresponding weitage of the stocks in that portfolio for that risk and return.

The above results obtained by the forecasted sample stocks data imply the follow

| Annual Risk | Annual Return | HEROMOTOCO weightage | TITAN weightage |
| --- | --- | --- | --- |
| 1.58808057% | 14.28614728% | 41.158594% | 58.822747% |
| 1.91180988% | 16.80614728% | 28.731898% | 71.268102% |
| 2.32307526% | 19.32614728% | 12.923434% | 87.076566% |
| 2.62625893% | 19.32614728% | 1.270392% | 98.729385% |

5.7.3 Mean Variance Optimization Output - forecasted sample stocks data

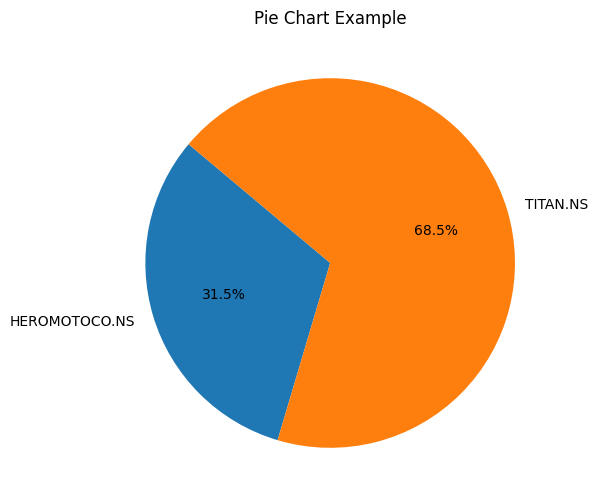
## 5.8. Final Portfolio Construction

UserExpectedReturn = 21.0  
**def** find\_closest\_element\_index(arr, target\_value):  
 *# Calculate the absolute differences between each element and the target value*  
 differences = np.abs(arr - target\_value)  
  
 *# Find the index of the minimum difference*  
 index\_of\_closest\_element = np.argmin(differences)  
  
 **return** index\_of\_closest\_element  
  
StatisticsWiseClosestindex = find\_closest\_element\_index(retPoint,UserExpectedReturn)  
DataScienceWiseClosestindex = find\_closest\_element\_index(GeneratedretPoint,UserExpectedReturn)  
  
print('weightage suggested by Statistics')  
print(xOptimalArray[StatisticsWiseClosestindex])  
print('weightage suggested by Data Science and Statistics')  
print(GeneratedxOptimalArray[DataScienceWiseClosestindex])

weightage suggested by Statistics  
[0.52526586 0.47473414]  
weightage suggested by Data Science and Statistics  
[0.01270392 0.98729385]

PortfolioByUserintuition = np.array([0.5,0.5])  
  
UserConfidenceOnStatisticsBasedPortfolio = 0.4  
UserConfidenceOnDataScienceBasedPortfolio = 0.4  
UserConfidenceOnIntuitionBasedPortfolio = 0.2  
*#Sum of all Confidence should be equal to 1.0*  
  
AdjustedStatisticsBasedPortfolio = xOptimalArray[StatisticsWiseClosestindex]\*UserConfidenceOnStatisticsBasedPortfolio  
AdjustedDataScienceBasedPortfolio = GeneratedxOptimalArray[DataScienceWiseClosestindex]\*UserConfidenceOnDataScienceBasedPortfolio  
AdjustedUserIntuitionBasedPortfolio = PortfolioByUserintuition\*UserConfidenceOnIntuitionBasedPortfolio  
  
print(AdjustedStatisticsBasedPortfolio)  
print(AdjustedDataScienceBasedPortfolio)  
print(AdjustedUserIntuitionBasedPortfolio)  
  
PortfolioWeightates=[AdjustedStatisticsBasedPortfolio,AdjustedDataScienceBasedPortfolio,AdjustedUserIntuitionBasedPortfolio]  
  
FinalPortfolio = np.zeros\_like(AdjustedUserIntuitionBasedPortfolio)  
  
**for** arr **in** PortfolioWeightates:  
 FinalPortfolio += arr  
  
print('Final Portfolio')  
  
print(FinalPortfolio)  
  
*# Create a pie chart*  
plt.figure(figsize=(6, 6)) *# Set the figure size*  
plt.pie(FinalPortfolio, labels=sample\_stocks, autopct='%1.1f%%', startangle=140)  
  
*# Add a title*  
plt.title('Pie Chart Example')  
  
*# Display the pie chart*  
plt.show()

[0.21010634 0.18989366]  
[0.00508157 0.39491754]  
[0.1 0.1]  
Final Portfolio  
[0.31518791 0.6848112 ]



HeroMotoCo Titan

5.8.1 Final Portfolio Pie Chart

Now, based on the user suggested return, we choose the suitable input data portfolio and forecasted data portfolio that is closest to the user suggested return. Then we get the user intuition portfolio.

Based on the user provided weightage, we calculate the final portfolio.

Based on user provided expected return, the closest input data mean variance optimization portfolio is 52.526586% of HEROMOTOCO + 47.473414% of TITAN.

Similarly, Based on user provided expected return, the closest input data mean variance optimization portfolio is 1.270392% of HEROMOTOCO + 98.729385% of TITAN.

The portfolio based on user intuition is 50% of HEROMOTOCO and 50% of TITAN

Based on user provided weightage based on the confidence in each approach, the final portfolio is given as follows.

|  | HEROMOTOCO weightage | TITAN weightage | User confidence on approach | HEROMOTOCO influence in final portfolio | TITAN influence in final portfolio |
| --- | --- | --- | --- | --- | --- |
| Statics based portfolio | 52.526586% | 47.473414% | 40% | 21.010634% | 18.989366% |
| Time series Based portfolio | 1.270392% | 98.729385% | 40% | 0.508157% | 39.491754% |
| User intuition portfolio | 50% | 50% | 20% | 10% | 10% |
|  |  |  |  | Total = 31.5% | Total = 68.5% |

5.8.2 Final portfolio construction

# 

# 6.Conclusions / Recommendations

In this project, we embarked on a comprehensive exploration of portfolio generation and optimization using a multi-faceted approach. We leveraged three distinct methods: mean optimization theory, time series analysis with the ARIMA model, and user intuition-driven portfolio creation. The integration of these methodologies allowed for a holistic and customizable approach to portfolio management.

We applied three different portfolio generation techniques, each with its own strengths and characteristics. The mean optimization theory offered a data-driven approach focused on historical returns and risk, while the ARIMA model introduced a time series forecasting perspective. User intuition added a qualitative dimension to portfolio construction.

By involving users in the portfolio-building process and allowing them to assign weightages to the different portfolios, we acknowledged the importance of aligning investment strategies with individual preferences and risk tolerance. This user-centric approach enhances the personalization of investment decisions.

The ability to adapt and rebalance portfolios in response to changing market conditions is essential. This project laid the groundwork for flexible portfolio adjustments based on user inputs and market dynamics.

We can further fine tune the process by considering the other financial parameters other than using closing price only. We can further improve ARIMA model forecasting by evaluating each stock and providing p,d and q values based on it.

# 7.Directions for future work

Explore the integration of machine learning algorithms, such as reinforcement learning or deep learning, for portfolio optimization. These techniques can capture more complex patterns and adapt to changing market conditions.

Invest in feature engineering to extract additional meaningful features from financial data, economic indicators, news sentiment, and alternative data sources. Enhanced features can improve the accuracy of forecasting models.

Investigate the implementation of algorithmic trading strategies within the portfolio management framework. Implement trading algorithms that execute portfolio adjustments automatically based on predefined rules and signals.

Develop a comprehensive backtesting framework that evaluates portfolio performance across various historical scenarios. This allows for a thorough assessment of the project's strategies.

Enhance the user interactions by developing an interactive UI with interacting and dynamic visualization.

# 

# 

# 8.Bibliography / References

1.A Journal Paper :

Markowitz, H. (1952). Portfolio Selection. The Journal of Finance, 7(1), 77-91.

2. A Journal Paper :

Litterman, R. B. (1986). Modern Investment Management: An Equilibrium Approach. BlackRock.

3. A Book:

Fabozzi, F. J., & Markowitz, H. M. (2011). The Theory and Practice of Investment Management. John Wiley & Sons.

4.Online Reference OECD Report :

<https://www.oecd.org/finance/financial-markets/Artificial-intelligence-machine-learning-big-data-in-finance.pdf>

# 

# 9.Appendix

A. Data Sources

A.1. Yahoo Finance Data

The historical stock price data for the project was sourced from Yahoo Finance (https://finance.yahoo.com/).

The data retrieval process and parameters used for fetching stock price data are documented in Section 5 of the project.

B. Python Libraries and Tools

B.1. Python Libraries

The following Python libraries were utilized extensively throughout the project for data analysis, modeling, and visualization:

pandas: Used for data manipulation and analysis.

matplotlib: Employed for creating data visualizations and plots.

numpy: Utilized for numerical computations and array operations.

statsmodels.tsa.arima.model: Used for implementing the ARIMA model for time series analysis.

scipy: Applied for scientific and statistical calculations.

B.2. Google Colab

Google Colab, a cloud-based Python development environment, was used for its computing power and convenient access to resources. The project leveraged Google Colab for code execution and data analysis.

C. Portfolio Optimization Models

C.1. Mean Optimization Theory

Details on the mean optimization theory approach are presented in Section 5 of the project.

Mathematical formulations and algorithms used for mean optimization are outlined in Section 5.

C.2. Time Series Analysis (ARIMA Model)

The application of the ARIMA model for time series analysis is described in Section 5 of the project.

Model parameters, order selection methods, and forecasting procedures are documented in Section 5.

C.3. User-Intuition Portfolio

Information about the user-intuition-driven portfolio creation process is provided in Section 5 of the project.

User inputs, preferences, and guidelines for portfolio construction are discussed in Section 5.

C.4. Combining Portfolios

The methodology for combining the three portfolios based on user-provided weightages is explained in Section 5 of the project.

Specific details on the combination process and the rationale behind it are outlined in Section 5.

D. User Guide

D.1. Portfolio Optimization User Guide

A comprehensive user guide detailing how to interact with and utilize the portfolio optimization tool is presented in Section 5 of the project.

Instructions for inputting user preferences, selecting weightages, and generating a customized portfolio are included.

E. Project Code

E.1. Code Documentation

Detailed code documentation, including comments and explanations, is provided within the code repository to aid users in understanding and replicating the project's functionality.

**Check list of items for the Final report**

1. Is the CoverPage in proper format? Y/N
2. Is the Title page in proper format? Y/N
3. Is the Certificate from the Supervisor in proper format? Has it been signed? Y/N
4. Is Abstract included in the Report? Is it properly written? Y/N
5. Does the Table of Contents page include chapter page numbers? Y/N
6. Does the Report contain a summary of the literature survey? Y/N

(i) Are the Pages numbered properly? Y/N

(ii)Are the Figures numbered properly? Y/N

(iii)Are the Tables numbered properly? Y/N

(iv)Are the Captions for the Figures and Tables proper? Y/N

(v)Are the Appendices numbered? Y/N

g)Does the Report have Conclusion / Recommendations of the work? Y/N

h)Are References/Bibliography given in the Report? Y/N

i)Have the References been cited in the Report? Y/N

j)Is the citation of References / Bibliography in proper format? Y/N