Portfolio Optimization

Introduction

Modern Portfolio Theory

The modern portfolio theory (MPT) is a method that can be used by risk-averse investors to construct diversified portfolios that maximize their returns without unacceptable levels of risk. The modern portfolio theory argues that any given investment's risk and return characteristics should not be viewed alone but should be evaluated by how it affects the overall portfolio's risk and return. That is, an investor can construct a portfolio of multiple assets that will result in greater returns without a higher level of risk. As an alternative, starting with a desired level of expected return, the investor can construct a portfolio with the lowest possible risk that is capable of producing that return.

The expected return of the portfolio is calculated as a weighted sum of the returns of the individual assets. The portfolio's risk is a function of the variances of each asset and the correlations of each pair of assets.

In the Markowitz mean-variance portfolio theory, one models the rate of returns on assets as random variables. These random variables are considered to be normally distributed with mean and variance as defined above.

Its basic formulation is as follows -

$$\begin{aligned} \min_{w} \ \overline{\sigma} \ &= w^T \, \hat{\Sigma} \, w \\ s.t. \quad & \\ w^T \hat{\mu} &= \overline{r} \\ w^T 1 &= 1 \end{aligned}$$

The formula expresses that we minimize the variance - covariance risk σ, where the matrix Σ is an estimate of the covariance of the assets. The vector ω denotes the individual investments subject to the condition ω^T 1 = 1 that the available capital is fully invested. The expected or target return $\bar{r}$ is expressed by the condition ω^T μ = $\bar{r}$, where the p-dimensional vector μ estimates the expected mean of the assets.

Risk Free Rate

The risk-free rate of return is a theoretical number within the capital markets that pertains to an investment that provides guaranteed returns with negligible or zero risk. In the financial market, a risk-free rate of return is attributed to the interest payments or the rate of return received by an investor on the money invested in a risk-free financial instrument over a specific period.

For an investor wanting to replicate the theoretical number of a risk-free rate of return, the closest example is Treasury Bills. Treasury bills are issued by the government and mature within one year. These bills do not offer a fixed interest payment but offer returns at maturity by allowing investors to buy the bills at a lesser rate than the face value.

Sharpe Ratio
The Sharpe Ratio is a way to measure the performance of an investment by taking risk into account. It can be used to evaluate a single security or an entire investment portfolio. In either case, the higher the ratio, the better the investment in terms of risk-adjusted returns.

By comparing the return on an investment to the extra risk associated with it above and beyond a risk-free asset—typically, the Sharpe ratio gives investors a clear picture of whether higher returns are adequately compensating them for taking on additional risk.

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

where:

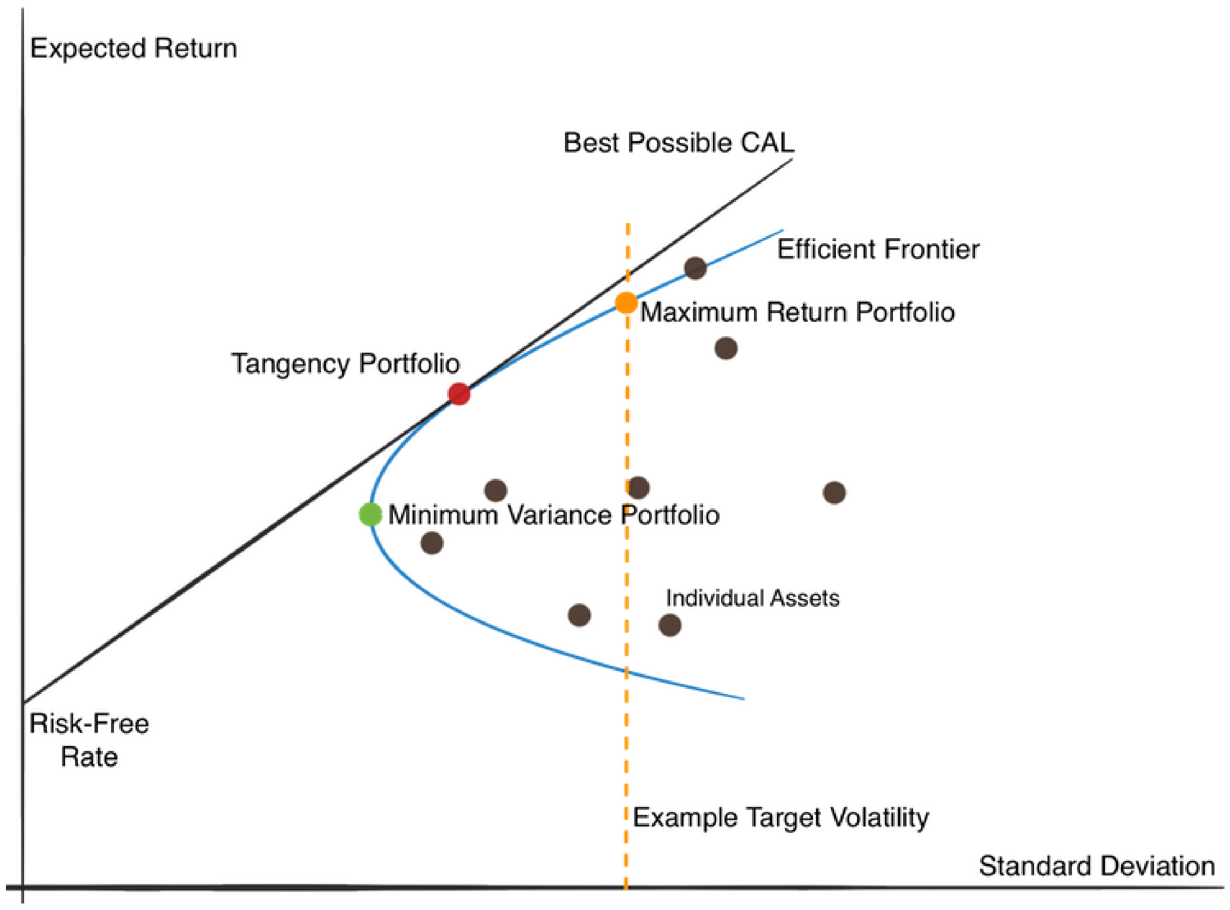
$R_p = \text{return of portfolio}$

$R_f = \text{risk-free rate}$

$\sigma_p = \text{standard deviation of the portfolio's excess return}$

Efficient Frontier

The Efficient Frontier is a hyperbola representing portfolios with all the different combinations of assets that result in efficient portfolios (i.e. with the lowest risk, given the same return and portfolios with the highest return, given the same risk). Risk is depicted on the X-axis and return is depicted on the Y-axis. The area inside the efficient frontier (but not directly on the frontier) represents either individual assets or all of their non-optimal combinations.



Tangency portfolio, the red point in the picture above, is the so-called optimal portfolio that realizes the highest possible Sharpe ratio. As we move from this point either to the right or to the left on the frontier, the Sharpe ratio, or in other words, the excess return-to-risk, will be lower.

The point where the hyperbola changes from convex to concave is where the **minimum variance portfolio** (green point in the picture) lies.

For a given level of volatility, there also exists a so-called **maximum return portfolio** (orange point in the picture), which, as the name suggests, maximizes the return given the level of volatility.

Let us now introduce the linear **Capital Market Line (CML)**. The point where the CML meets the Y-axis is where an investor's risk-free asset, like government securities, lies in terms of the return. This line is tangent to the efficient frontier exactly at the Maximum Sharpe portfolio point. The CML (tangency) line then represents a portfolio of different combinations of a risk-free asset and a tangency portfolio (also called a maximum Sharpe portfolio or sometimes an "optimal portfolio").

The Limited Assets Markowitz Model

(from the Research paper *Cesarone Scozzari Tardella - Algorithms for Constrained Portfolio Optimization*)

The classical Markowitz model is a convex quadratic programming problem which can be solved by a number of efficient algorithms with a moderate computational effort even for large instances. We now add to the MV model the realistic constraint that no more than K assets should be held in the portfolio (a cardinality constraint), and furthermore that the quantity xi of each asset that is included in the portfolio should be limited within a given interval (a quantity constraint or buy-in threshold). Thus we obtain the following Limited Assets Markowitz model:

$$\begin{array}{ll}
\text{Min} & \sum_{i=1}^{n}\sum_{j=1}^{n} \sigma_{ij} x_i x_j \\
st & \\
& \sum_{i=1}^{n} \mu_i x_i = \rho \\
& \sum_{i=1}^{n} x_i = 1 \\
& x_i = 0 \text{ or } \ell_i \le x_i \le u_i, \quad i = 1, \dots, n \\
& |supp(x)| \le K,
\end{array}$$

where $supp(x) = \{i : x_i > 0\}$.

The problem is no longer a convex optimization problem because of the non-convexity of its feasible region. But the problem can be reformulated as a Mixed Integer Quadratic Program (MIQP) with the addition of n binary variables:

$$\begin{array}{lll}
\text{Min} & \sum_{i=1}^{n}\sum_{j=1}^{n} \sigma_{ij} x_i x_j & \\
st & & \\
& \sum_{i=1}^{n} \mu_i x_i = \rho & \\
& \sum_{i=1}^{n} x_i = 1 & \\
& \sum_{i=1}^{n} y_i \le K & \\
& \ell_i y_i \le x_i \le u_i y_i & i = 1, \dots, n \\
& x_i \ge 0 & i = 1, \dots, n \\
& y_i \in \{0, 1\} & i = 1, \dots, n
\end{array}$$

Methodology

Data Preparation

We have collected data of stock prices for the last two years from Yahoo Finance using the *yfinance* library of Python. It's an open-source tool that uses Yahoo's publicly available APIs, and is intended for research and educational purposes. The number of stocks (*N*) and the choice of stocks is upto the user and it may be based on one of the following criterias -

1. Top *N* companies according to the full Market Capitalisation
2. Top *N* companies based on the expected rate of return (μ_i)

For this project we have chosen the top 30 companies (N=30) based on market capitalization from the Indian market.

After getting the data, we have calculated the mean and covariance matrix -

$r_t = (R_{t+1} - R_t) / R_t$

$\mu_i = \sum_{t=1}^{T} r_t / T$

Basic Markowitz Model

In this model, we try to mind out the minimum risk portfolio corresponding to the given rate of return. We will need to solve the following optimization problem -

$$\begin{aligned} \min_{w} \ \overline{\sigma} \ &= w^T \hat{\Sigma} \, w \\ s.t. & \\ w^T \hat{\mu} &= \overline{r} \\ w^T 1 &= 1 \end{aligned}$$

Set -
$i \in I$ {list of stocks - $s_1, s_2, \ldots\ldots, s_N$}
Variables -
w - vector of length *N* corresponding to the weight assigned to each stock

Parameters -
r - required rate of return
μ - vector of length *N* corresponding to the mean rate of return of each stock
$\sum$ - matrix of size $N \times N$ with $\sum_{ij} = \mathrm{Cov}(s_i, s_j)$

The first constraint ensures that the total expected return on investment is equal to *r* which is specified by the user.
The second constraint ensures that the weights sum to get one so that the weights are normalized.

This problem is Quadratic (non-linear) optimization problem. Gurobi can solve it. As the problem is best suited for matrix formulation, we use Gurobi Python matrix interface.

Maximizing the Sharpe Ratio

Set -
i ∈ I {list of stocks - s_1, s_2,…….., s_N}

Variables -
x - vector of length *N* corresponding to the weight assigned to each stock

Parameters -
r_f - risk-free rate
μ - vector of length *N* corresponding to the mean rate of return of each stock
∑ - matrix of size *N*×*N* with $\Sigma_{ij} = \mathrm{Cov}(s_i, s_j)$

$$\begin{aligned} \max_{x} \quad & \frac{\mu^\top x - r_f}{\sqrt{x^\top \Sigma x}} \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i = 1 \\ & x \geq 0. \end{aligned}$$

Constraint 1 specifies that the weights are normalized
Constraint 2 specifies that the weights are non-negative (short selling is not allowed)

This model is non-convex making it difficult to solve.

We can reformulate the problem as follows -

$$\begin{aligned} \max_{y} \quad & \frac{1}{\sqrt{y^\top \Sigma y}} \\ \text{s.t.} \quad & (\mu - r_f)^\top y = 1 \\ & y \geq 0. \end{aligned}$$

where $y_i = x_i / \mu^T x$

equivalently, $x_i = y_i / \sum_{j=1}^{N} y_j$

Still the model is non-convex but maximizing the given objective is equivalent to minimizing $y^T \Sigma y$ and thus we get a convex optimization problem corresponding to the original problem -

$$\begin{aligned} \min_{y} \quad & y^\top \Sigma y \\ \text{s.t.} \quad & (\mu - r_f)^\top y = 1 \\ & y \geq 0. \end{aligned}$$

And we can get back x_i using the following transformation -

$$x_i = y_i / \sum y_j$$

This problem is again a Quadratic Optimization problem similar to the Mean Variance Optimization problem and similar methods can be used to solve it.

Adding Additional Constraints

To make the model more realistic, we can add some more constraints to any of the above two formulations. Some important constraints are as follows -

1. Maximum Weight constraint - Limit the weight fraction of each asset
2. Number of asset constraint - Limit on number of stocks to invest in

Adding the first constraint is not a problem but if we add the second constraint, we will need to add N binary variables and the problem will become MILP which is more difficult to solve.

The modified model for the maximizing Sharpe Ratio formulation will be as follows -

$$\begin{aligned} \min_{y} \quad & y^\top \Sigma y \\ \text{s.t.} \quad & (\mu - r_f)^\top y = 1 \\ & y \geq 0. \end{aligned}$$

$$y_i \leq L z_i \vee \ i \in I$$

$$\sum_{i=1}^{N} z_i \leq K \ \vee \ i \in I$$

$$z_i \in \{0, 1\} \vee \ i \in I$$

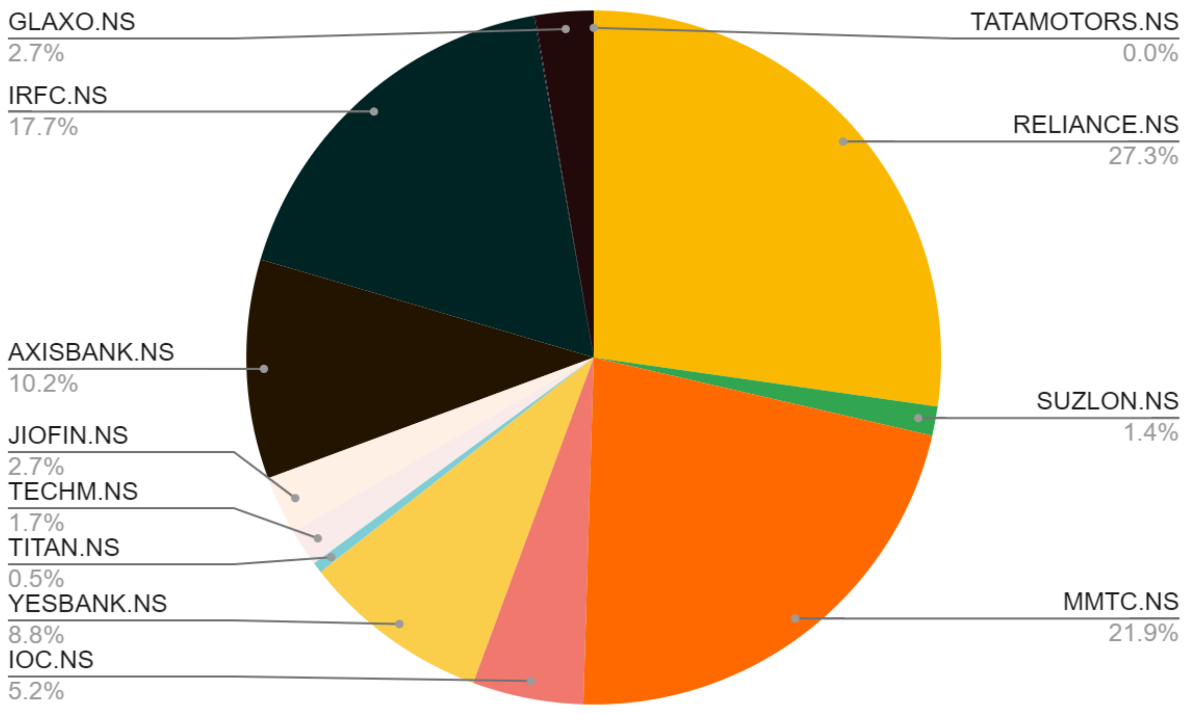
Gurobi uses Branch and Cut method to solve such problems.

Results

Basic Markowitz Model

Volatility - 0.007566
Expected Return -0.000831

TATAMOTORS.NS	0.000000
TATAPOWER.NS	0.000000
RELIANCE.NS	0.272622
SUZLON.NS	0.013578
MMTC.NS	0.218586
ADANIPOWER.NS	0.000000
ADANIPORTS.NS	0.000000
IOC.NS	0.051618
YESBANK.NS	0.087540
ZOMATO.NS	0.000000
AWL.NS	0.000000
TITAN.NS	0.004916
ASIANPAINT.NS	0.000000
BANDHANBNK.NS	0.000000
IDBI.NS	0.000000
JWL.NS	0.000000
NHPC.NS	0.000000
VEDL.NS	0.000000
WIPRO.NS	0.000000
TECHM.NS	0.017418
MCX.NS	0.000000
PNB.NS	0.000001
JIOFIN.NS	0.027125
GSPL.NS	0.000000
AXISBANK.NS	0.102206
IRFC.NS	0.176799
RVNL.NS	0.000000
BHEL.NS	0.000514
IREDA.NS	0.000000
GLAXO.NS	0.027075

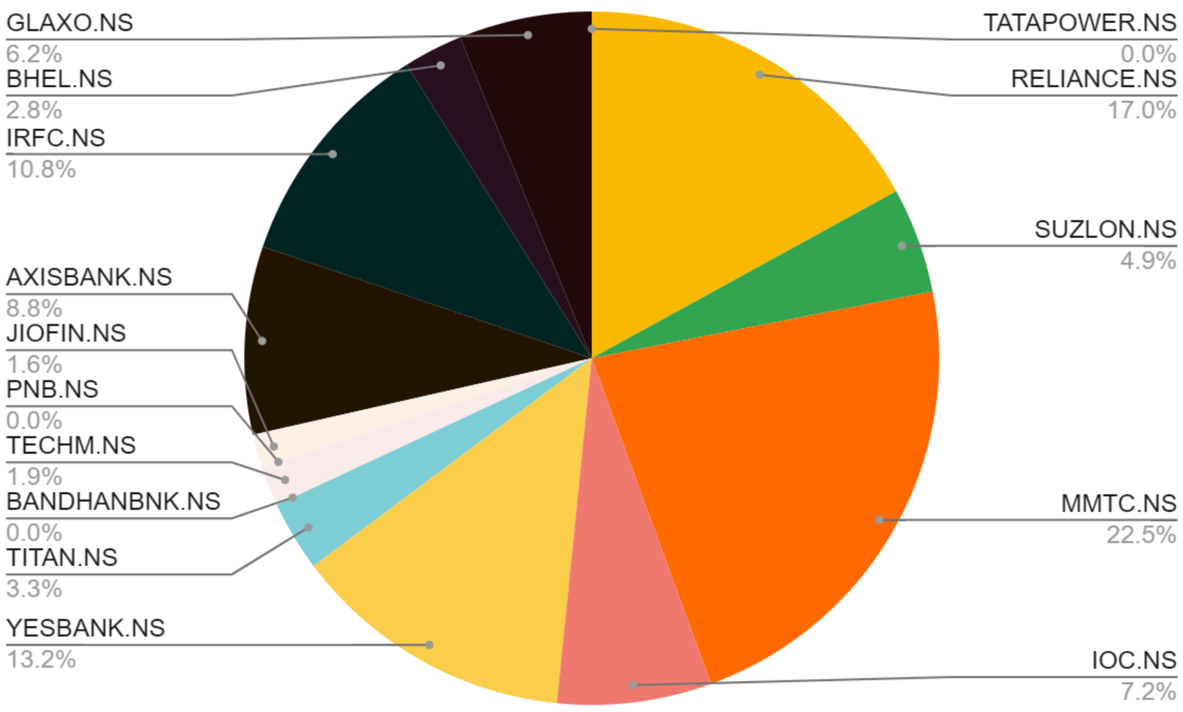


Maximizing the Sharpe Ratio

TATAMOTORS.NS	0.000000
TATAPOWER.NS	0.000000
RELIANCE.NS	0.170013
SUZLON.NS	0.049109
MMTC.NS	0.225046
ADANIPOWER.NS	0.000000
ADANIPORTS.NS	0.000000
IOC.NS	0.071805
YESBANK.NS	0.131827
ZOMATO.NS	0.000000
AWL.NS	0.000000
TITAN.NS	0.032697
ASIANPAINT.NS	0.000000
BANDHANBNK.NS	0.000001
IDBI.NS	0.000000

JWL.NS	0.000000
NHPC.NS	0.000000
VEDL.NS	0.000000
WIPRO.NS	0.000000
TECHM.NS	0.018537
MCX.NS	0.000000
PNB.NS	0.000004
JIOFIN.NS	0.015522
GSPL.NS	0.000000
AXISBANK.NS	0.087580
IRFC.NS	0.108431
RVNL.NS	0.000000
BHEL.NS	0.027515
IREDA.NS	0.000000
GLAXO.NS	0.061912

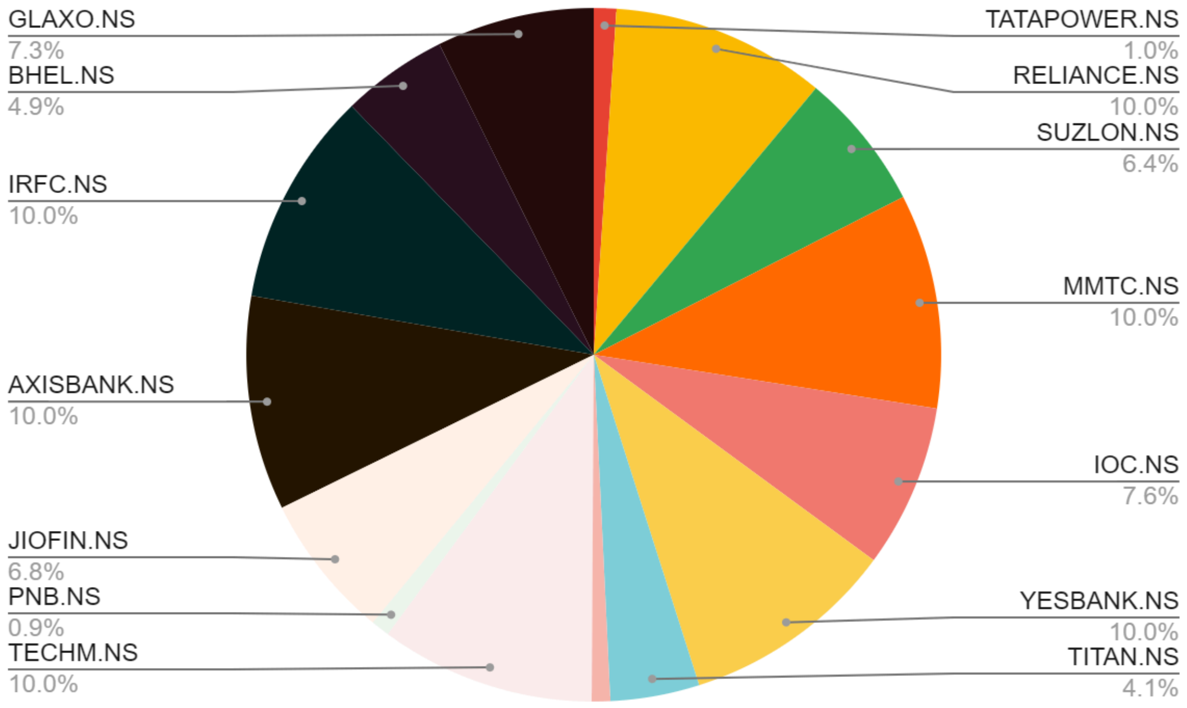
Volatility - 0.060892
Expected Return - 0.001686



The Limited Assets Model

TATAMOTORS.NS	0.000000
TATAPOWER.NS	0.010434
RELIANCE.NS	0.100000
SUZLON.NS	0.064372
MMTC.NS	0.100000
ADANIPOWER.NS	0.000000
ADANIPORTS.NS	0.000000
IOC.NS	0.076116
YESBANK.NS	0.100000
ZOMATO.NS	0.000000
AWL.NS	0.000000
TITAN.NS	0.041472
ASIANPAINT.NS	0.000000
BANDHANBNK.NS	0.008532
IDBI.NS	0.000000
JWL.NS	0.000000
NHPC.NS	0.000000
VEDL.NS	0.000000
WIPRO.NS	0.000000
TECHM.NS	0.100000
MCX.NS	0.000000
PNB.NS	0.008597
JIOFIN.NS	0.067728
GSPL.NS	0.000000
AXISBANK.NS	0.100000
IRFC.NS	0.100000
RVNL.NS	0.000000
BHEL.NS	0.049412
IREDA.NS	0.000000
GLAXO.NS	0.073337

Volatility - 0.064062
Expected Return - 0.002419



Analysis

Model	**Volatility**	**Expected Return**
Basic Markowitz Model	0.007566	0.000831
Sharpe Ratio Model	0.060892	0.001686
Limited Asset Model	0.064062	0.002419

We can see that as we move from Basic Markowitz Model to Sharpe Ratio model both Expected Return and Volatility increases. This is in line with the basic principle of working of this models. Markowitz model focuses on minimizing the risk and hence expected return is also less. In case of Sharpe Ratio model, it tries to take a balanced approach and hence both are increased.

While adding the Maximum Weight constraint, when we set L=0.3, the answer doesn’t change as none of the original weight value exceeded the threshold. When we tried to reduce L, like L=0.25, the model couldn't solve the problem. Similar problem was faced when we tried to put a Minimum Weight constraint, it couldn’t solve the problem even if the lower limit is kept 0.01.

When we added the limit on number of assets, there was again an increase in expected return and volatility value. We could also observe that many assets that were assigned insignificant weights in the earlier two models were not chosen in this case making their weight to be zero. Also, as we tried to decrease K (K=10) the weight assigned to each chosen asset was 0.1 so if we try to decrease K to a small value, the portfolio would not have a chance to diversify and hence choose the best K assets and assign them weight.

Conclusion

Markowitz model and Sharpe Ratio model both give optimal portfolios but their mode of working is different, Markowitz model takes the safest way minimizing the risk while Sharpe Ratio model takes a balanced approach between risk and return. The choice of model depends on the investor.

Adding additional constraints makes the model more realistic but the parameter values need to be chosen appropriately otherwise the p[ortfolio may not be upto the mark or even problem may become infeasible.

References

1. https://www.investopedia.com/terms/m/modernportfoliotheory.asp
2. https://quantpedia.com/markowitz-model/
3. https://www.forbes.com/advisor/in/investing/sharpe-ratio/
4. https://www.indiainfoline.com/knowledge-center/share-market/what-is-the-risk-free-rate-of-return
5. https://www.actuaries.org/AFIR/Colloquia/Rome2/Cesarone_Scozzari_Tardella.pdf
6. Mean-Variance Portfolio Optimization: Eigendecomposition-Based Methods (diva-portal.org) (Chapter 3)
7. Gurobi Optimizer Reference Manual (iitk.ac.in)
8. https://www.investopedia.com/articles/basics/06/invest1000.asp
9. https://fortune.com/recommends/investing/how-to-start-investing/
10. ECO 712: INTERNATIONAL FINANCE AND INVESTMENT

Appendix

```
# Commented out IPython magic to ensure Python compatibility.
# %pip install gurobipy yfinance

import yfinance as yf
import gurobipy as gp
from gurobipy import GRB
from math import sqrt
import pandas as pd

"""## Collecting Data"""

stocks = ['TATAMOTORS.NS', 'TATAPOWER.NS', 'RELIANCE.NS',
'SUZLON.NS','MMTC.NS', 'ADANIPOWER.NS', 'ADANIPORTS.NS', 'IOC.NS',
'YESBANK.NS', 'ZOMATO.NS', 'AWL.NS', 'TITAN.NS', 'ASIANPAINT.NS',
'BANDHANBNK.NS', 'IDBI.NS', 'JWL.NS', 'NHPC.NS', 'VEDL.NS', 'WIPRO.NS',
'TECHM.NS', 'MCX.NS', 'PNB.NS', 'JIOFIN.NS', 'GSPL.NS', 'AXISBANK.NS',
'IRFC.NS', 'RVNL.NS', 'BHEL.NS', 'IREDA.NS', 'GLAXO.NS']
```

```
data = yf.download(stocks, period='1y') # period= '2y', '5y'
data=data.dropna()

print(data)

"""## Compute Mean and Variance matrix"""

import numpy as np

closes = np.transpose(np.array(data.Close)) # matrix of daily closing
prices
absdiff = np.diff(closes)                   # change in closing price each
day
reldiff = np.divide(absdiff, closes[:,:-1]) # relative change in daily
closing price
delta = np.mean(reldiff, axis=1)            # mean price change
sigma = np.cov(reldiff)                     # covariance
std = np.std(reldiff, axis=1)               # standard deviation

"""## Minimize risk by solving QP model"""

# Create an empty model
m = gp.Model('portfolio')

# Add matrix variable for the stocks
x = m.addMVar(len(stocks))

# Objective is to minimize risk (squared).  This is modeled using the
# covariance matrix, which measures the historical correlation between
stocks
portfolio_risk = x @ sigma @ x
m.setObjective(portfolio_risk, GRB.MINIMIZE)

# Fix budget with a constraint
m.addConstr(x.sum() == 1, 'budget')

# Verify model formulation
m.write('portfolio_selection_optimization.lp')
```

```
# Optimize model to find the minimum risk portfolio
m.optimize()

"""## Display minimum risk portfolio using Pandas"""

minrisk_volatility = sqrt(m.ObjVal)
minrisk_return = delta @ x.X
pd.DataFrame(data=np.append(x.X, [minrisk_volatility, minrisk_return]),
             index=stocks + ['Volatility', 'Expected Return'],
             columns=['Minimum Risk Portfolio'])

"""## Compute the efficient frontier"""

# Create an expression representing the expected return for the portfolio
portfolio_return = delta @ x
target = m.addConstr(portfolio_return == minrisk_return, 'target')

# Solve for efficient frontier by varying target return
frontier = np.empty((2,0))
for r in np.linspace(delta.min(), delta.max(), 25):
    target.rhs = r
    m.optimize()
    frontier = np.append(frontier, [[sqrt(m.ObjVal)],[r]], axis=1)

"""## Plot results"""

import matplotlib.pyplot as plt
#plt.figure(figsize=(10,10))

fig, ax = plt.subplots(figsize=(10,8))

# Plot volatility versus expected return for individual stocks
ax.scatter(x=std, y=delta,
           color='Blue', label='Individual Stocks')
for i, stock in enumerate(stocks):
    ax.annotate(stock, (std[i], delta[i]))

# Plot volatility versus expected return for minimum risk portfolio
ax.scatter(x=minrisk_volatility, y=minrisk_return, color='DarkGreen')
ax.annotate('Minimum\nRisk\nPortfolio', (minrisk_volatility,
minrisk_return),
            horizontalalignment='right')

# Plot efficient frontier
ax.plot(frontier[0], frontier[1], label='Efficient Frontier',
color='DarkGreen')

# Format and display the final plot
ax.axis([frontier[0].min()*0.7, frontier[0].max()*1.3, delta.min()*1.2,
delta.max()*1.2])
ax.set_xlabel('Volatility (standard deviation)')
ax.set_ylabel('Expected Return')
ax.legend()
ax.grid()
plt.show()

"""Sharpe Ratio Optimization"""

# Create an empty model
m = gp.Model('sharp_ratio_portfolio')

# Add matrix variable for the stocks
x = m.addMVar(len(stocks))

# Objective is to minimize risk (squared).  This is modeled using the
# covariance matrix, which measures the historical correlation between
stocks
portfolio_risk = x @ (sigma- (1.02**(1/252)-1)/100)/std @ x
m.setObjective(portfolio_risk, GRB.MINIMIZE)

# Fix budget with a constraint
m.addConstr(x.sum() == 1, 'budget')

# Verify model formulation
m.write('portfolio_selection_optimization_sharpe.lp')

# Optimize model to find the minimum risk portfolio
m.optimize()
```

```
minrisk_volatility = sqrt(m.ObjVal)
minrisk_return = delta @ x.X
pd.DataFrame(data=np.append(x.X, [minrisk_volatility, minrisk_return]),
             index=stocks + ['Volatility', 'Expected Return'],
             columns=['Minimum Risk Portfolio'])

"""Sharpe Ratio with Constraint on min and max limit of investment


"""

# Create an empty model
m = gp.Model('sharp_ratio_portfolio')

# Add matrix variable for the stocks
x = m.addMVar(len(stocks))

# Objective is to minimize risk (squared).  This is modeled using the
# covariance matrix, which measures the historical correlation between
stocks
portfolio_risk = x @ (sigma- (1.02**(1/252)-1)/100)/std @ x
m.setObjective(portfolio_risk, GRB.MINIMIZE)

# Fix budget with a constraint
m.addConstr(x.sum() == 1, 'budget')
for i in range(len(stocks)):
  m.addConstr(x[i] <= 0.3)

# Verify model formulation
m.write('portfolio_selection_optimization_sharpe.lp')

# Optimize model to find the minimum risk portfolio
m.optimize()

minrisk_volatility = sqrt(m.ObjVal)
minrisk_return = delta @ x.X
pd.DataFrame(data=np.append(x.X, [minrisk_volatility, minrisk_return]),
             index=stocks + ['Volatility', 'Expected Return'],
             columns=['Minimum Risk Portfolio'])
```

```
"""## Sharpe Ratio with constratint on both max investment and number of
stocks held"""

# Create an empty model
m = gp.Model('sharp_ratio_portfolio')

# Add matrix variable for the stocks
x = m.addMVar(len(stocks))
y = m.addMVar(len(stocks), vtype=GRB.BINARY)  # Binary decision variables

# Objective is to minimize risk (squared). This is modeled using the
# covariance matrix, which measures the historical correlation between
stocks
portfolio_risk = x @ (sigma- (1.02**(1/252)-1)/100)/std @ x
m.setObjective(portfolio_risk, GRB.MINIMIZE)

# Fix budget with a constraint
m.addConstr(x.sum() == 1, 'budget')
for i in range(len(stocks)):
  m.addConstr(x[i] <= 0.3)

# Limit on number of stocks to invest in
max_stocks_to_invest = 15  # Define your maximum number of stocks to
invest in
m.addConstr(y.sum() == max_stocks_to_invest, 'max_stocks')

# # Linearization constraint: Ensure x_i is zero if y_i is zero
for i in range(len(stocks)):
    m.addConstr(y[i] >= x[i])            # Ensure y[i] = 1 if x[i] > 0
    m.addConstr(x[i] <= y[i] * 0.1)      # Ensure y[i] = 0 if x[i] = 0

# Verify model formulation
m.write('portfolio_selection_optimization_sharpe.lp')

# Optimize model to find the minimum risk portfolio
m.optimize()

minrisk_volatility = sqrt(m.ObjVal)
minrisk_return = delta @ x.X
pd.DataFrame(data=np.append(x.X, [minrisk_volatility, minrisk_return]),
             index=stocks + ['Volatility', 'Expected Return'],
             columns=['Minimum Risk Portfolio'])
```