UG PROJECT SEMESTER VI

PORTFOLIO OPTIMIZATION USING QUADRATIC PROGRAMMING

Dept. of Mathematical Sciences
Indian Institute of Technology
(Banaras Hindu University)
Varanasi 221005

Gaurav Gupta *20124019*

Supervisor Dr. Debdas Ghosh

UG Project Presentation

May 3, 2023

PORTFOLIO OPTIMIZATION USING QUADRATIC PROGRAMMING

A portfolio is a combination of assets, such as stocks, bonds, and real estate, in which an investor has invested their money. Portfolio optimization is the process of constructing an optimal combination of assets that maximizes the investor's expected return for a given level of risk. We need to invest in optimal weights w_i in each asset. We can characterize a portfolio by the weights allocated to it, let $w = [w_1, ..., w_n]$ be a vector of portfolio weights. One approach to portfolio optimization is through **mean-variance optimization (MVO)**, which was first introduced by Harry Markowitz in 1952. It is based on the principle that investors are risk-averse and seek to maximize their expected returns while minimizing their portfolio risk.

ASSUMPTIONS OF MPT

- Investors are rational and avoid risks whenever possible
- Investors aim for the maximum returns for their investment
- All investors share the aim maximizing their expected returns
- Commissions and taxes on the market are left out of consideration
- All investors have access to the same sources and level of all necessary information about investment decisions
- Investors have unlimited access to borrow and lend money at the risk free rate

Consider assets $S_1, S_2, ..., S_n (n \geq 2)$ with random returns. Let μ_i and σ_i denote the expected return and the standard deviation of the return of asset S_i. For $i \neq j$, ρ_{ij} denotes the correlation coefficient of the returns of assets S_i and S_j . Let $\mu = [\mu 1, ..., \mu n]^T$, and $\Sigma = (\sigma_{ij})$ be the $n \times n$ symmetric covariance matrix with $\sigma_{ii} = {\sigma_i}^2$ and $\sigma_{ij} = \rho_{ij}\sigma_i\sigma_j$ for $i \neq j$. Denoting by w_i the proportion of the total funds invested in security i, one can represent the expected return and the variance of the resulting portfolio $w = [w_1, ..., w_n]$ as follows:

$$\alpha_w = E[w] = w_1\mu_1 + ... + w_n\mu_n = \mu^T w,$$

and

$$ {\sigma_w}^2 = Var[w] = \sum_{(i,j)} \rho_{ij}\sigma_i\sigma_j w_i w_j = w^T \Sigma w,$$

Since variance is always nonnegative, it follows that $w^T \Sigma w \geq 0$ for any w , i.e., Σ is positive semidefinite. We further assume that the set of admissible portfolios is a nonempty polyhedral set and represent it as

$$W := \{w : Aw = b, Cw \geq d\},$$

where A is an $m \times n$ matrix, b is an m-dimensional vector, C is a $p \times n$ matrix and d is a p-dimensional vector. In particular one constraint

$$\sum_{i=1}^{n} w_i = 1$$

Linear portfolio constraints such as short-sale restrictions or limits on asset/sector allocations are subsumed in our generic notation W for the polyhedral feasible set.

Evaluate different portfolios w using the mean-variance pair of the portfolio: $(\alpha_w, {\sigma_w}^2)$ with preferences for: Higher expected returns α_w Lower variance σ_w

---

Markowitz' mean-variance optimization (MVO) problem can be formulated in three different but equivalent ways. Find the minimum variance portfolio of the securities 1 to n that yields at least a target value of expected return (say b). Mathematically, this formulation produces a quadratic programming problem:

Problem I: Risk Minimization

For a given choice of target mean return R, choose the portfolio w to

$$min_w \frac{1}{2} w^T \Sigma w$$

$$w^T \mu \geq R$$

$$w^T 1_{n \times 1} = 1$$

Solution

Apply the method of Lagrange multipliers to the convex optimization (minimization) problem subject to linear constraints:

- Define the Lagrangian

$$L(w, \lambda_1, \lambda_2) = \frac{1}{2} w^T \Sigma w + \lambda_1 (R - w^T \mu) + \lambda_2 (1 - w^T 1_{n \times 1})$$

- Derive the first-order conditions

$$\frac{\partial L}{\partial w} = 0_n = \Sigma w - \lambda_1 \mu - \lambda_2 1_n$$

$$\frac{\partial L}{\partial \lambda_1} = 0 = R - w^T \mu$$

$$\frac{\partial L}{\partial \lambda_2} = 0 = 1 - w^T 1_n$$

- Solve for w in terms of λ_1, λ_2:

$$w_0 = \lambda_1 \Sigma^{-1}\mu + \lambda_2 \Sigma^{-1} 1_n$$

-Solve for λ_1, λ_2 by substituting for w:

$$R = w^T\mu = \lambda_1(\mu^T\Sigma^{-1}\mu) + \lambda_2(\mu^T\Sigma^{-1}1_n)$$

$$1 = {w_0}^T 1_n = \lambda_1(\mu^T\Sigma^{-1}1_n) + \lambda_2({1_n}^T\Sigma^{-1}1_n)$$

let $\mu^T\Sigma^{-1}\mu = a$, similarly $\mu^T\Sigma^{-1}1_n = b$ and ${1_n}^T\Sigma^{-1}1_n = c$

$$\begin{bmatrix} R \\ 1 \end{bmatrix} = \begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}$$

- Variance of Optimal Portfolio with Return R with the given values of λ_1 and λ_2, the solution portfolio:

$$w_0 = \lambda_1 \Sigma^{-1}\mu + \lambda_2 \Sigma^{-1} 1_n$$

has minimum variance equal to

$${\sigma_0}^2 = {w_0}^T\Sigma w_0$$

$$= {\lambda_1}^2(\mu^T\Sigma^{-1}\mu) + 2\lambda_1\lambda_2(\mu^T\Sigma^{-1}1_n) + {\lambda_2}^2({1_n}^T\Sigma^{-1}1_n)$$

$$= \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}^T \begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}$$

- Substituting

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} a & b \\ b & c \end{bmatrix}^{-1} \begin{bmatrix} R \\ 1 \end{bmatrix}$$

gives

$${\sigma_0}^2 = \begin{bmatrix} R \\ 1 \end{bmatrix}^T \begin{bmatrix} a & b \\ b & c \end{bmatrix}^{-1} \begin{bmatrix} R \\ 1 \end{bmatrix}$$

$$= \frac{1}{ac - b^2}(cR^2 - 2bR + a)$$

Optimal portfolio has variance ${\sigma_0}^2$: parabolic in the mean

A feasible portfolio w is called efficient if it has the maximal expected return among all portfolios with the same variance, or alternatively, if it has the minimum variance among all portfolios that have at least a certain expected return.

The collection of efficient portfolios form the efficient frontier of the portfolio universe

The efficient frontier is often represented as a curve in a two-dimensional graph where the coordinates of a plotted point corresponds to the standard deviation and the expected return of an efficient portfolio which forms a parabola.

Equivalent Optimization Problems

Problem II: Expected Return Maximization

For a given choice of target return variance ${\sigma_0}^2$, choose the portfolio w to:

$$Maximize : E(R_w) = w^T\mu$$

$$s.t. : w^T\Sigma w = {\sigma_0}^2$$
$$w^T 1_n = 1$$

Problem III: Risk Aversion Optimization

Let $\lambda \geq 0$ denote the risk aversion index gauging the trade-ff between risk and return. Choose the portfolio w to:

$$Maximize : [E(R_w) - \frac{1}{2}\lambda var(R_w)] = w^T\mu - \frac{1}{2}\lambda w^T\Sigma w$$

$$s.t. : w^T 1_n = 1$$

- Problems I,II, and III solved by equivalent Lagrangians
- Efficient Frontier:$\{(R_0, \sigma_0^2) = (E(R_{w_0}), Var(R_{w_0}))|w_0 optimal\}$

IMPLEMENTATION

We are now going to apply MVO on **10 MOST ACTIVELY TRADED STOCKS ON NSE (National Stock Exchange)** and considering **GOLD** as a risk free investment and **NIFTY50** as our market

10 STOCKS

- Tata Steel Ltd
- ICICI Bank Ltd
- State Bank of India
- Tata Motors Ltd
- HDFC Bank Ltd
- Axis Bank Ltd
- ITC Ltd
- Reliance Industries Ltd
- Adani Enterprises Ltd
- Oil & Natural Gas Corpn Ltd

[27]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import yfinance as yf
from numpy import matrix, power
from math import sqrt
```

[28]:
```
selected = ["ADANIENT.NS",
            "AXISBANK.NS",
            "HDFCBANK.NS",
            "ICICIBANK.NS",
            "ITC.NS",
            "ONGC.NS",
            "RELIANCE.NS",
            "SBIN.NS",
            "TATAMOTORS.NS",
            "TATASTEEL.NS"]
```

[29]:
```
data = yf.download("TATASTEEL.NS ICICIBANK.NS SBIN.NS TATAMOTORS.NS HDFCBANK.NS␣
 ↪AXISBANK.NS ITC.NS RELIANCE.NS ADANIENT.NS ONGC.NS",period =␣
 ↪"500d",interval="1wk")
```

```
[*********************100%***********************]  10 of 10 completed
```

[30]:
```
close = data['Adj Close'].copy()
```

[31]:
```
returns_wk = close.pct_change()
returns_wk.dropna(inplace=True)
```

[32]:
```
returns_cov_wk = returns_wk.cov()
```

[33]:
```
returns_cov_wk
```

[33]:
```
               ADANIENT.NS  AXISBANK.NS  HDFCBANK.NS  ICICIBANK.NS    ITC.NS  \
ADANIENT.NS       0.008167     0.000528     0.000634      0.000578  0.000001
AXISBANK.NS       0.000528     0.001376     0.000593      0.000711  0.000317
HDFCBANK.NS       0.000634     0.000593     0.000832      0.000556  0.000241
ICICIBANK.NS      0.000578     0.000711     0.000556      0.000963  0.000261
ITC.NS            0.000001     0.000317     0.000241      0.000261  0.001116
ONGC.NS           0.001188     0.000317     0.000116      0.000141  0.000322
RELIANCE.NS       0.000947     0.000407     0.000269      0.000380  0.000280
SBIN.NS           0.001384     0.000847     0.000624      0.000782  0.000399
TATAMOTORS.NS     0.000861     0.000505     0.000597      0.000538  0.000574
TATASTEEL.NS      0.001206     0.000331     0.000894      0.000894  0.000584

                ONGC.NS  RELIANCE.NS   SBIN.NS  TATAMOTORS.NS  TATASTEEL.NS
ADANIENT.NS    0.001188     0.000947  0.001384       0.000861      0.001206
AXISBANK.NS    0.000317     0.000407  0.000847       0.000505      0.000331
HDFCBANK.NS    0.000116     0.000269  0.000624       0.000597      0.000894
ICICIBANK.NS   0.000141     0.000380  0.000782       0.000538      0.000894
ITC.NS         0.000322     0.000280  0.000399       0.000574      0.000584
ONGC.NS        0.001828     0.000677  0.000472       0.000566      0.000503
RELIANCE.NS    0.000677     0.001092  0.000419       0.000551      0.000088
SBIN.NS        0.000472     0.000419  0.001299       0.000758      0.000950
TATAMOTORS.NS  0.000566     0.000551  0.000758       0.002510      0.001079
```

```
TATASTEEL.NS   0.000503    0.000088  0.000950       0.001079      0.014411
```

[34]:
```
returns_annual = returns_wk.mean() * 50

returns_cov_annual = returns_cov_wk *50
```

To calculate RISK FREE RETURNS we will use Annual Returns of GOLD commodity - on 5 APRIL 2021 = Rs 46855 - on 13 APRIL 2023 = Rs 62535 Risk Free Return = 0.1673247252160922%

[35]:
```
risk_free_returns = (62535-46855)/46855* 0.5
```

For MARKET RETURNS we use NIFTY50 index

[36]:
```
nifty_data = yf.download("^NSEI",period = "500d",interval="1wk")
```

```
[*********************100%***********************]  1 of 1 completed
```

[37]:
```
nifty_close = nifty_data['Adj Close']
nifty_returns_wk = nifty_close.pct_change()
nifty_returns_wk.dropna(inplace=True)
market_returns = nifty_returns_wk.mean() *50
market_stdev = nifty_returns_wk.std()*50
```

[38]:
```
sharpe_ratio = []
port_returns = []
port_volatility = []
stock_weights = []

num_assets = 10
num_portfolios = 100000
```

USING MONTE CARLO SIMULATONS FOR FINDING THE EFFICIENT FRONTIER

[39]:
```
# populate the empty lists with each portfolios returns,risk and weights
for single_portfolio in range(num_portfolios):
    weights = np.random.random(num_assets)
    weights /= np.sum(weights)

    returns = np.dot(weights, returns_annual)
    volatility = np.sqrt(np.dot(weights.T, np.dot(returns_cov_annual, weights)))

    sharpe = (returns-risk_free_returns) / volatility
    sharpe_ratio.append(sharpe)

    port_returns.append(returns)
    port_volatility.append(volatility)
    stock_weights.append(weights)
```

[40]:

```python
portfolio = {'Returns': port_returns,
             'Volatility': port_volatility,
             'Sharpe Ratio': sharpe_ratio}

for counter,symbol in enumerate(selected):
    portfolio[symbol+' Weight'] = [Weight[counter] for Weight in stock_weights]
```

[41]:

```python
df = pd.DataFrame(portfolio)
```

[42]:

```python
df
```

[42]:

```
        Returns  Volatility  Sharpe Ratio  ADANIENT.NS Weight
0      0.326709    0.203051      0.784949            0.032995  \
1      0.320185    0.215169      0.710416            0.124285
2      0.320551    0.219401      0.698385            0.168499
3      0.305690    0.202871      0.682034            0.140946
4      0.291031    0.207542      0.596053            0.122329
…          …           …             …                  …
99995  0.272322    0.190210      0.552009            0.008319
99996  0.314183    0.233418      0.629163            0.194884
99997  0.277807    0.198870      0.555552            0.159344
99998  0.324289    0.215851      0.727190            0.130100
99999  0.321041    0.216271      0.710756            0.158472

       AXISBANK.NS Weight  HDFCBANK.NS Weight  ICICIBANK.NS Weight
0                0.053119            0.107137             0.043489  \
1                0.040287            0.143521             0.078472
2                0.099359            0.105085             0.034162
3                0.134003            0.076034             0.107282
4                0.123420            0.095219             0.095274
…                     …                  …                    …
99995            0.001164            0.144744             0.017903
99996            0.193459            0.060489             0.000867
99997            0.186022            0.149596             0.097632
99998            0.074615            0.084630             0.072068
99999            0.067594            0.065061             0.186530

       ITC.NS Weight  ONGC.NS Weight  RELIANCE.NS Weight  SBIN.NS Weight
0           0.172025        0.225282            0.067152        0.052702  \
1           0.134428        0.087241            0.097646        0.096833
2           0.143128        0.081340            0.038018        0.160041
3           0.137440        0.148129            0.043032        0.008550
4           0.071339        0.106560            0.121128        0.130572
…              …               …                   …               …
99995       0.145193        0.099740            0.129757        0.173763
99996       0.072057        0.197646            0.063500        0.107481
99997       0.145530        0.095570            0.028139        0.012321
```

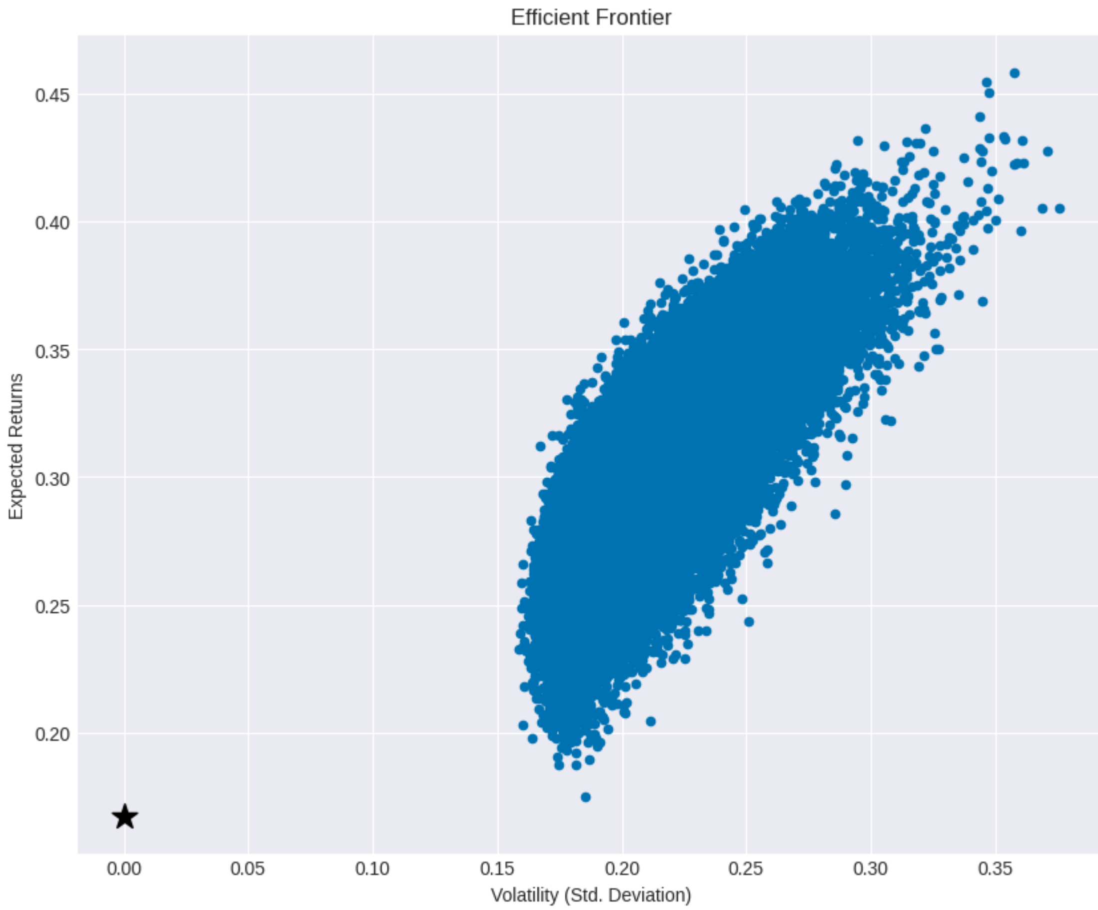
```
99998       0.106064       0.158907          0.055952       0.098987
99999       0.112997       0.100110          0.054312       0.130358

       TATAMOTORS.NS Weight  TATASTEEL.NS Weight
0                  0.133092             0.113006
1                  0.083650             0.113638
2                  0.100710             0.069659
3                  0.157714             0.046870
4                  0.049110             0.085049
…                     …                    …
99995              0.258199             0.021219
99996              0.028708             0.080909
99997              0.109833             0.016014
99998              0.125072             0.093604
99999              0.042853             0.081714

[100000 rows x 13 columns]
```

[43]:

```
plt.style.use('seaborn-v0_8-colorblind')
df.plot.scatter(x='Volatility', y='Returns', figsize=(10, 8), grid=True)
plt.scatter(x=0, y=risk_free_returns, c='black', marker='*', s=200 )
plt.xlabel('Volatility (Std. Deviation)')
plt.ylabel('Expected Returns')
plt.title('Efficient Frontier')
plt.show()
```



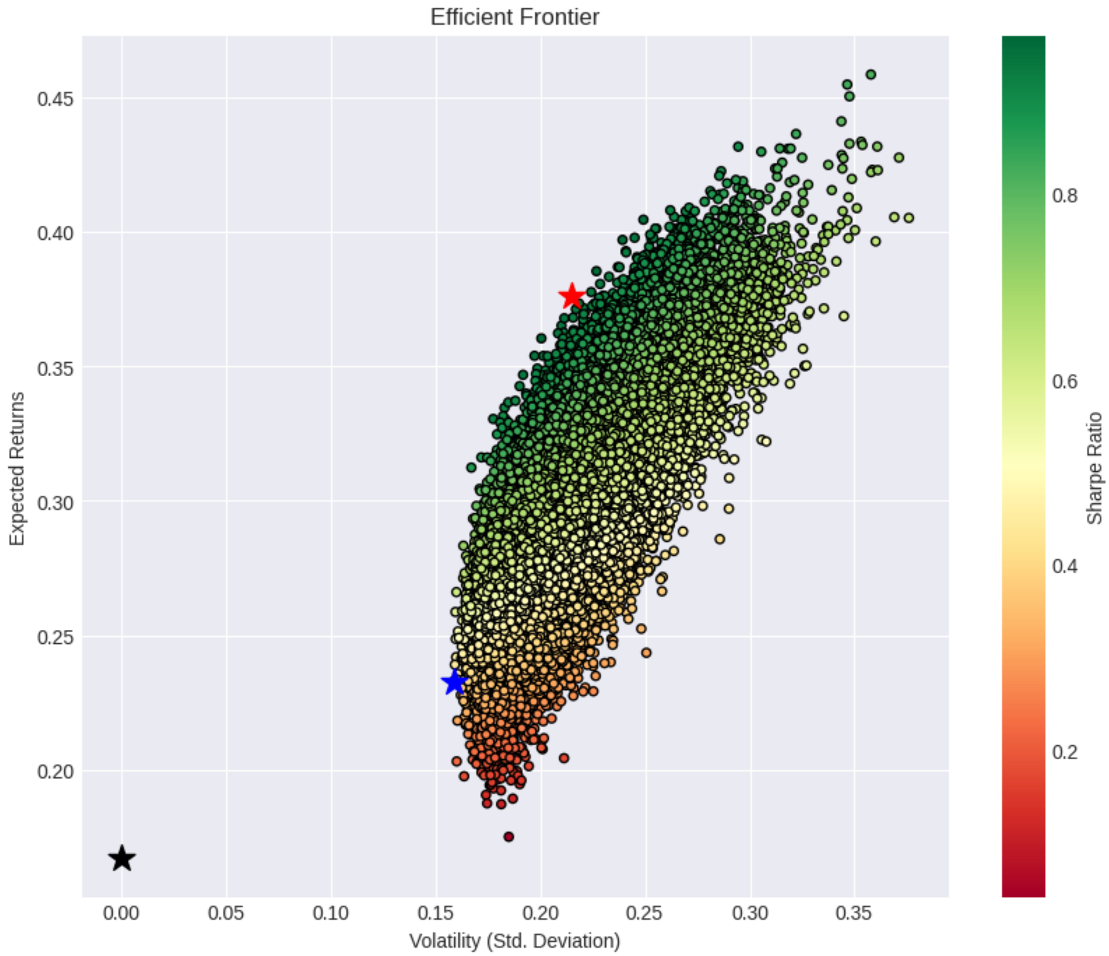
[44]:
```
min_volatility = df['Volatility'].min()
max_sharpe = df['Sharpe Ratio'].max()

# use the min, max values to locate and create the two special portfolios
max_sharpe_portfolio = df.loc[df['Sharpe Ratio'] == max_sharpe]
min_variance_port = df.loc[df['Volatility'] == min_volatility]


plt.style.use('seaborn-v0_8-dark')
df.plot.scatter(x='Volatility', y='Returns', c='Sharpe Ratio',
                cmap='RdYlGn', edgecolors='black', figsize=(10, 8), grid=True)

plt.scatter(x=max_sharpe_portfolio['Volatility'],␣
 ↪y=max_sharpe_portfolio['Returns'], c='red', marker='*', s=200)
plt.scatter(x=min_variance_port['Volatility'], y=min_variance_port['Returns'],␣
 ↪c='blue', marker='*', s=200 )
plt.scatter(x=0, y=risk_free_returns, c='black', marker='*', s=200 )
plt.xlabel('Volatility (Std. Deviation)')
```

```
plt.ylabel('Expected Returns')
plt.title('Efficient Frontier')
plt.show()
```



[45]:
```
print('MINIMUM RISK PORTFOLIO')
print(min_variance_port.T)
print('')
print('')
print('MAXIMUM SHARPE RATIO PORTFOLIO')
print(max_sharpe_portfolio.T)
```

```
MINIMUM RISK PORTFOLIO
                          6464
Returns               0.232866
Volatility            0.158635
Sharpe Ratio          0.413156
ADANIENT.NS Weight    0.016746
```

```
AXISBANK.NS Weight    0.037173
HDFCBANK.NS Weight    0.234893
ICICIBANK.NS Weight   0.172123
ITC.NS Weight         0.229005
ONGC.NS Weight        0.031703
RELIANCE.NS Weight    0.214222
SBIN.NS Weight        0.056907
TATAMOTORS.NS Weight  0.002091
TATASTEEL.NS Weight   0.005135


MAXIMUM SHARPE RATIO PORTFOLIO
                         14191
Returns               0.376095
Volatility            0.214929
Sharpe Ratio          0.971344
ADANIENT.NS Weight    0.123532
AXISBANK.NS Weight    0.023385
HDFCBANK.NS Weight    0.033578
ICICIBANK.NS Weight   0.054824
ITC.NS Weight         0.291677
ONGC.NS Weight        0.152740
RELIANCE.NS Weight    0.007353
SBIN.NS Weight        0.186549
TATAMOTORS.NS Weight  0.023192
TATASTEEL.NS Weight   0.103169
```

EFFICIENT FRONTIER and CAPITAL MARKET LINE

The CML is a straight line that starts at the risk-free rate of return and extends to the expected return and volatility of the market portfolio, which is a theoretical portfolio that includes all investable assets in the market in proportion to their market values. The slope of the CML represents the market's risk premium, which is the excess return that investors demand for taking on the risk of the market portfolio.

[46]:
```
df['Volatility' ] = df['Volatility'].round(5)
```

[47]:
```
df_ef = pd.DataFrame(df.groupby(by='Volatility')['Returns'].max())
df_ef.reset_index(inplace=True)
```

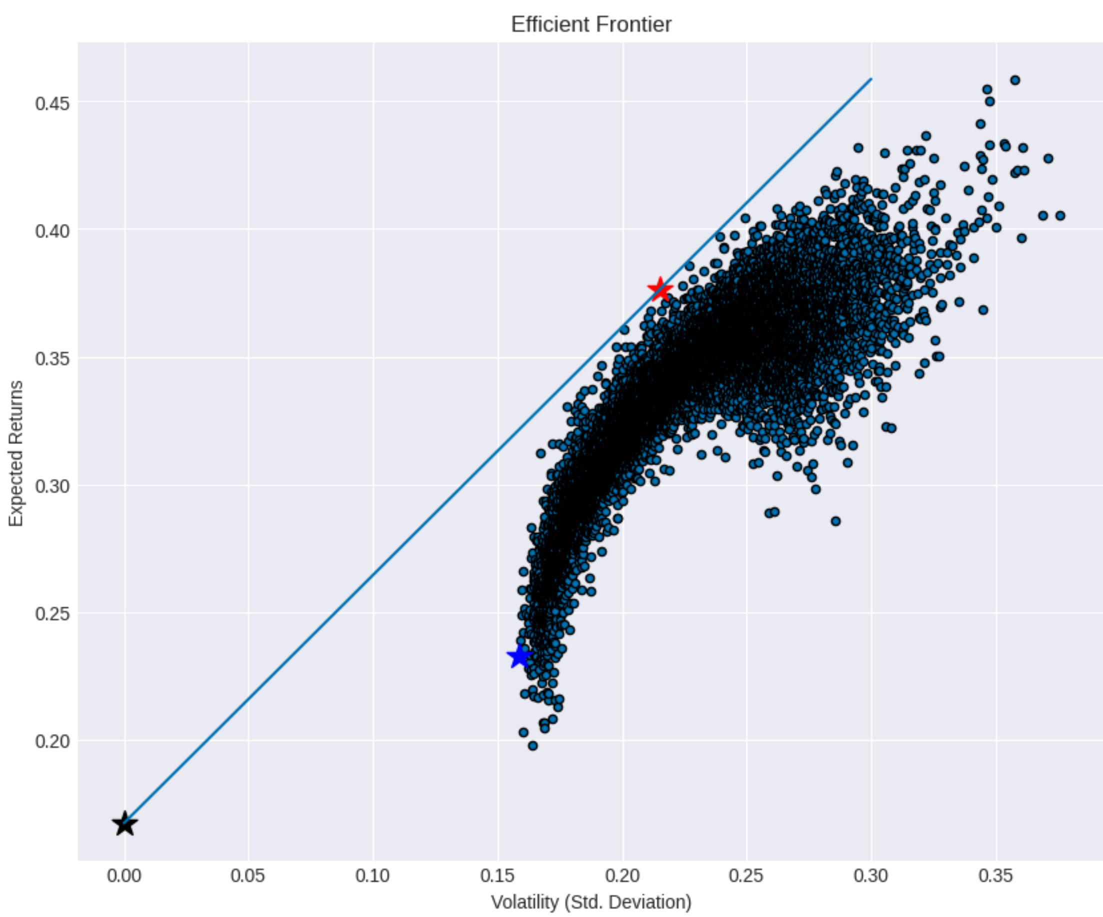
[48]:
```
min_volatility = df['Volatility'].min()
max_sharpe = df['Sharpe Ratio'].max()

# use the min, max values to locate and create the two special portfolios
max_sharpe_portfolio = df.loc[df['Sharpe Ratio'] == max_sharpe]
min_variance_port = df.loc[df['Volatility'] == min_volatility]
```

```
plt.style.use('seaborn-v0_8-dark')
df_ef.plot.scatter(x='Volatility', y='Returns',edgecolors='black', figsize=(10,␣
 ↪8), grid=True)
plt.scatter(x=max_sharpe_portfolio['Volatility'],␣
 ↪y=max_sharpe_portfolio['Returns'], c='red', marker='*', s=200)
plt.scatter(x=min_variance_port['Volatility'], y=min_variance_port['Returns'],␣
 ↪c='blue', marker='*', s=200 )
plt.scatter(x=0, y=risk_free_returns, c='black', marker='*', s=200 )

x = np.linspace(0, 0.3,200,dtype=float)
y = max_sharpe* x + risk_free_returns
# Plot the line
plt.plot(x, y)

plt.xlabel('Volatility (Std. Deviation)')
plt.ylabel('Expected Returns')
plt.title('Efficient Frontier')
plt.show()
```



[49]:
```python
max_sr_weights = np.array(max_sharpe_portfolio.values[0][-10:])
min_risk_weights = np.array(min_variance_port.values[0][-10:])
min_risk_weights = np.append(min_risk_weights,0.0)
```

[50]:
```python
cum_returns = returns_wk.cumsum()

# cum_returns.drop(columns=['sr_prft','risk_prft'],inplace=True)
```

[51]:
```python
cum_returns['sr_prft'] = cum_returns.dot(max_sr_weights)
cum_returns['risk_prft'] = cum_returns.dot(min_risk_weights)
cum_returns
```

[51]:
```
                           ADANIENT.NS  AXISBANK.NS  HDFCBANK.NS
Date
2021-05-03 00:00:00+05:30     0.120998     0.002588     0.001735  \
2021-05-10 00:00:00+05:30     0.060523    -0.041709    -0.017986
2021-05-17 00:00:00+05:30     0.138165     0.025298     0.061655
2021-05-24 00:00:00+05:30     0.128660     0.037543     0.065762
2021-05-31 00:00:00+05:30     0.434472     0.040517     0.064099
…                                  …           …           …
2023-04-03 00:00:00+05:30     0.854424     0.247797     0.224433
2023-04-10 00:00:00+05:30     0.920955     0.262352     0.240096
2023-04-17 00:00:00+05:30     0.884740     0.262178     0.229549
2023-04-24 00:00:00+05:30     0.952377     0.257318     0.237312
2023-05-01 00:00:00+05:30     0.949728     0.269702     0.237105

                           ICICIBANK.NS    ITC.NS   ONGC.NS  RELIANCE.NS
Date
2021-05-03 00:00:00+05:30      0.010991  0.013574  0.030513    -0.031462  \
2021-05-10 00:00:00+05:30     -0.005151  0.047175  0.043972    -0.028588
2021-05-17 00:00:00+05:30      0.070439  0.032098  0.042201     0.005092
2021-05-24 00:00:00+05:30      0.071373  0.050515  0.038654     0.051159
2021-05-31 00:00:00+05:30      0.070828  0.031022  0.155254     0.096843
…                                   …         …        …            …
2023-04-03 00:00:00+05:30      0.434871  0.796161  0.587194     0.222823
2023-04-10 00:00:00+05:30      0.462300  0.817459  0.641311     0.228824
2023-04-17 00:00:00+05:30      0.447450  0.849436  0.646665     0.226064
2023-04-24 00:00:00+05:30      0.483581  0.891812  0.642593     0.256503
2023-05-01 00:00:00+05:30      0.488431  0.889227  0.675936     0.264993

                            SBIN.NS  TATAMOTORS.NS  TATASTEEL.NS   sr_prft
Date
2021-05-03 00:00:00+05:30  0.013437       0.030288      0.143472  0.042068  \
2021-05-10 00:00:00+05:30  0.019578       0.061667      0.100972  0.041380
2021-05-17 00:00:00+05:30  0.132631       0.064389      0.084189  0.074358
2021-05-24 00:00:00+05:30  0.184600       0.082434      0.075564  0.088052
2021-05-31 00:00:00+05:30  0.211967       0.133258      0.091151  0.146165
```

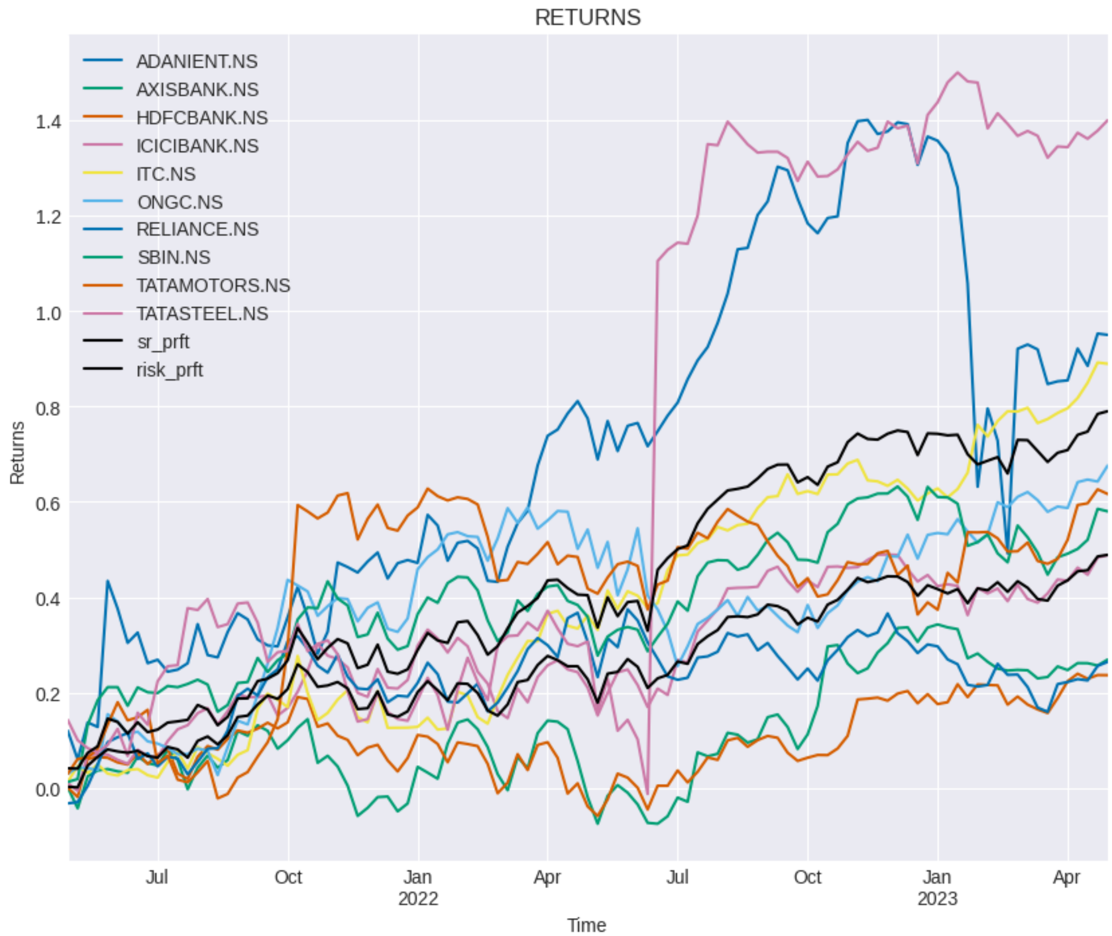
```
…                                   …            …            …       …
2023-04-03 00:00:00+05:30  0.491440       0.520655      1.342779  0.708556
2023-04-10 00:00:00+05:30  0.501485       0.593431      1.372966  0.740343
2023-04-17 00:00:00+05:30  0.520623       0.597051      1.360407  0.747180
2023-04-24 00:00:00+05:30  0.585338       0.626232      1.377365  0.784124
2023-05-01 00:00:00+05:30  0.580151       0.616540      1.399134  0.789800

                           risk_prft
Date
2021-05-03 00:00:00+05:30   0.003322
2021-05-10 00:00:00+05:30   0.002186
2021-05-17 00:00:00+05:30   0.047755
2021-05-24 00:00:00+05:30   0.066101
2021-05-31 00:00:00+05:30   0.081611
…                                …
2023-04-03 00:00:00+05:30   0.435714
2023-04-10 00:00:00+05:30   0.454527
2023-04-17 00:00:00+05:30   0.456814
2023-04-24 00:00:00+05:30   0.485736
2023-05-01 00:00:00+05:30   0.489018

[105 rows x 12 columns]
```

[52]:

```
plt.style.use('seaborn-v0_8-dark')
cum_returns.plot(style={'sr_prft': 'black', 'risk_prft': 'black'},figsize=(10,
 ↪8), grid=True)
plt.xlabel('Time')
plt.ylabel('Returns')
plt.title('RETURNS')
plt.show()
```


RETURNS
ADANIENT.NS
AXISBANK.NS
HDFCBANK.NS
ICICIBANK.NS
ITC.NS
ONGC.NS
RELIANCE.NS
SBIN.NS
TATAMOTORS.NS
TATASTEEL.NS
sr_prft
risk_prft
Returns
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Jul
Oct
Jan
2022
Apr
Jul
Oct
Jan
2023
Apr
Time

```
[ ]:
```