A.Y.: 2023-24 **Class/Sem: B.E.B.Tech/ Sem-VII** **Sub:** Quantitative Portfolio Management

Experiment 3

Name: Umang Kirit Lodaya **SAP ID: 60009200032**

Batch: D11

Aim: Implementation of Asset Efficient Frontier on a given dataset.

Objective:

- Understand the concept of the efficient frontier.
- Learn how to calculate the efficient frontier for a given dataset.
- Implement the efficient frontier in Python.

Theory:

The efficient frontier is a set of portfolios that offer the highest expected return for a given level of risk. It is a curve that is formed by plotting the expected returns of a portfolio against its standard deviation of returns. The efficient frontier is a useful tool for investors who want to maximize their returns for a given level of risk.

To calculate the efficient frontier, we need to know the expected returns and standard deviations of the assets in our portfolio. Once we have this information, we can use a mathematical optimization algorithm to find the portfolios that lie on the efficient frontier.

The efficient frontier is based on the concept of mean-variance analysis. Mean-variance analysis is a method for evaluating the risk and return of a portfolio. It uses two measures of risk:

- Expected return: The expected return of a portfolio is the average return that we expect to earn over a given period of time.
- Standard deviation: The standard deviation of a portfolio is a measure of its volatility. It tells us how much the returns of a portfolio are likely to fluctuate from one period to the next.

The efficient frontier is a set of portfolios that offer the highest expected return for a given level of risk. In other words, no other portfolio on the efficient frontier offers a higher expected return for the same level of risk.

The efficient frontier is a curve that is formed by plotting the expected returns of a portfolio against its standard deviation of returns. The curve is typically upward-sloping, which means that portfolios with higher expected returns also have higher levels of risk.

The efficient frontier is a useful tool for investors who want to maximize their returns for a given level of risk. By plotting the efficient frontier, investors can see the different portfolios that are available to them and choose the portfolio that best suits their risk tolerance.

Here are some additional points about the efficient frontier:

- The efficient frontier is a theoretical concept. In practice, there may be other factors that investors need to consider, such as taxes and transaction costs.
- The efficient frontier is not static. It changes over time as the expected returns and standard deviations of the assets in the portfolio change.
- The efficient frontier can be extended to include a risk-free asset. The risk-free asset is a security that has no risk and a guaranteed return. By adding a risk-free asset to a portfolio, investors can reduce their risk without sacrificing too much return.

Lab Experiment to be done by students:

1. Download the dataset of asset returns.
2. Calculate the expected returns and standard deviations of the assets in the dataset.
3. Implement the efficient frontier in Python.
4. Plot the efficient frontier.
5. Analyze the efficient frontier and identify the portfolios that are most suitable for your risk tolerance.

NAME: UMANG KIRIT LODAYA

SAP ID: 60009200032

BATCH: D11

```
In [ ]: import yfinance as yf
        import numpy as np
        import matplotlib.pyplot as plt
```

```
In [ ]: tickers = ['AAPL', 'MSFT', 'GOOGL', 'TSLA', 'AMZN']  # Add m
        e tickers if needed
        start_date = '2020-01-01'
        end_date = '2021-12-31'
```

```
In [ ]: #Step 1: Download asset data
        def download_asset_data(tickers, start_date, end_date):
            data = yf.download(tickers, start=start_date, end=end_dat
        e)['Adj Close']
            return data
        asset_data = download_asset_data(tickers, start_date, end_dat
        e)
```

```
[*********************100%***********************]  5 of 5 co
pleted
```

```
In [ ]: # Step 2: Calculate returns
        def calculate_returns(data):
            returns = data.pct_change().dropna()
            return returns

        returns = calculate_returns(asset_data)
```

```
In [ ]: # Step 3: Calculate expected returns and standard deviations
        def calculate_expected_returns(returns):
            expected_returns = returns.mean() * 252  # Assuming 252 tr
        ading days in a year
            return expected_returns

        expected_returns = calculate_expected_returns(returns)
        cov_matrix = returns.cov()
```

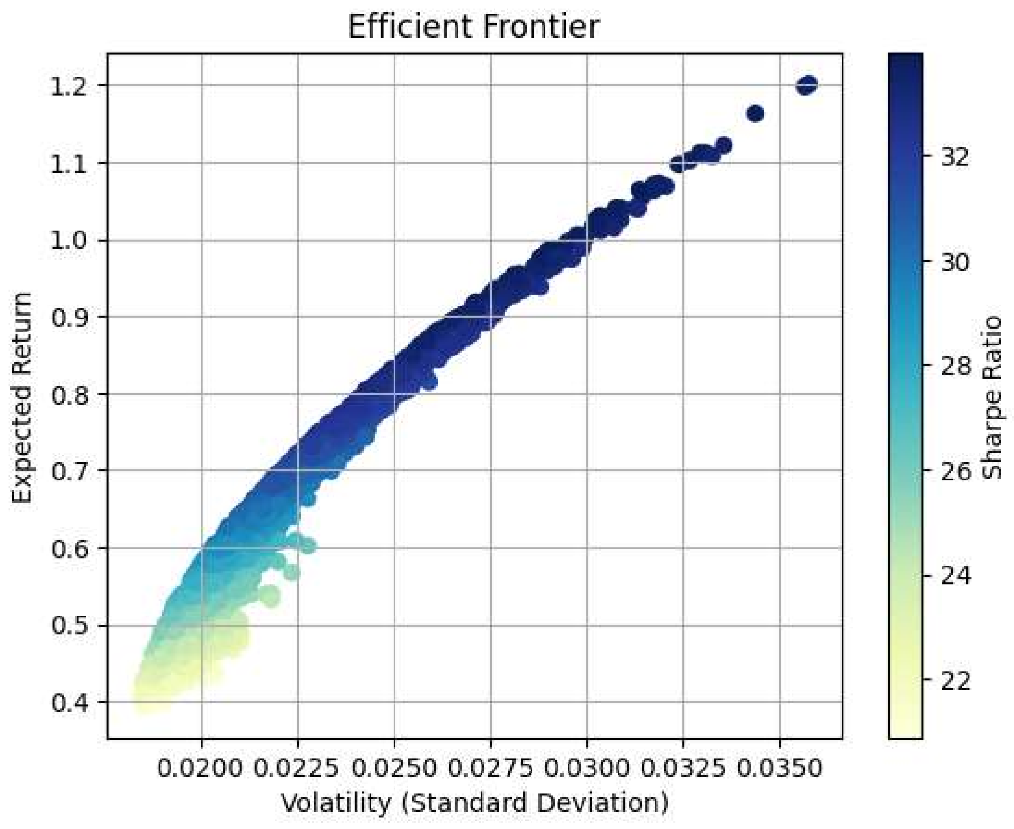
In [ ]:
```
# Step 4: Plot the efficient frontier
def plot_efficient_frontier(expected_returns, cov_matrix, num_
portfolios):
    results = np.zeros((3, num_portfolios))
    risk_free_rate = 0  # You can adjust the risk-free rate if
needed

    for i in range(num_portfolios):
        weights = np.random.random(len(expected_returns))
        weights /= np.sum(weights)
        portfolio_return = np.sum(expected_returns * weights)
        portfolio_std_dev = np.sqrt(np.dot(weights.T, np.dot(c
ov_matrix, weights)))
        portfolio_sharpe_ratio = (portfolio_return - risk_free
_rate) / portfolio_std_dev

        results[0,i] = portfolio_return
        results[1,i] = portfolio_std_dev
        results[2,i] = portfolio_sharpe_ratio

    plt.scatter(results[1,:], results[0,:], c=results[2,:], cm
ap='YlGnBu', marker='o')
    plt.colorbar(label='Sharpe Ratio')
    plt.xlabel('Volatility (Standard Deviation)')
    plt.ylabel('Expected Return')
    plt.title('Efficient Frontier')
    plt.grid(True)
    plt.show()


num_portfolios = 5000  # You can adjust the number of portfoli
os
plot_efficient_frontier(expected_returns, cov_matrix, num_port
folios)
```


Efficient Frontier
Expected Return
Volatility (Standard Deviation)
Sharpe Ratio