ESG_Integrated_Port_Opt

July 5, 2024

1 Integrating ESG Considerations into Portfolio Optimization

The goal of this project is to demonstrate how Environmental, Social, and Governance (ESG) considerations can be integrated into portfolio optimization. By considering ESG factors alongside traditional financial metrics, investors can create portfolios that are not only financially sound but also socially responsible. This project will showcase the process of integrating ESG scores into portfolio optimization, and analyzing the results to improve investment strategies.

1.0.1 Utilizing PyPortfolioOpt Library

PyPortfolioOpt is a comprehensive library for portfolio optimization in Python. It provides tools for constructing and optimizing portfolios using various optimization methods. Key features of PyPortfolioOpt include:

- **Efficient Frontier Calculation**: PyPortfolioOpt can compute the efficient frontier, which represents the set of portfolios offering the highest expected return for a given level of risk.
- **Risk Models**: The library supports different risk models, including mean-variance optimization, minimum variance, and maximum Sharpe ratio.
- **Asset Allocation**: PyPortfolioOpt helps in determining the optimal asset allocation based on various constraints and objectives.
- **Performance Evaluation**: It includes functions to evaluate portfolio performance, such as calculating expected returns, volatility, and Sharpe ratios.

In this project, we use PyPortfolioOpt to integrate ESG scores into our portfolio optimization process. The library's flexibility and powerful optimization capabilities make it an ideal choice for this task.

[1]:
```
# Uncomment the line below before running
# pip install -U yfinance PyPortfolioOpt
```

[2]:
```
#imports
import yfinance as yf
import pandas as pd
import numpy as np
import cvxpy as cp
from pypfopt import expected_returns, risk_models, EfficientFrontier, plotting
from pypfopt.discrete_allocation import DiscreteAllocation, get_latest_prices
import matplotlib.pyplot as plt
from pypfopt.plotting import plot_efficient_frontier
from mpl_toolkits.mplot3d import Axes3D
```

```
[3]: stocks = ["AAPL", "MSFT", "GOOG", "AMZN", "META", "TSLA", "NVDA", "NFLX"]
     data = yf.download(stocks, start="2018-01-01", end="2024-01-01")['Adj Close']
```

```
[*********************100%%**********************]  8 of 8 completed
```

```
[4]: data_reset = data.reset_index()
     df_numeric = data_reset.drop(columns=['Date'])
     df_numeric.head()
```

```
[4]: Ticker        AAPL       AMZN       GOOG        META       MSFT        NFLX  \
     0        40.615894  59.450500  53.189472  181.047943  79.936737  201.070007
     1        40.608803  60.209999  54.062481  184.291290  80.308739  205.050003
     2        40.797436  60.479500  54.258255  183.951996  81.015587  205.630005
     3        41.261929  61.457001  55.048855  186.466827  82.020027  209.990005
     4        41.108673  62.343498  55.284088  187.893890  82.103714  212.050003

     Ticker       NVDA       TSLA
     0        4.930643  21.368668
     1        5.255147  21.150000
     2        5.282848  20.974667
     3        5.327617  21.105333
     4        5.490859  22.427334
```

Let's look at an unconstrained portfolio's performance. Our portfolio expects Capital Asset Pricing Model (CAPM) returns, and utilizes semi-covariance method to assess risk.

We can understand how a semicovariance matrix works as a risk model. `PyPortfolioOpt` relies on the concept of mean-variance optimization such that it requires a risk mode for optimization. A simple risk model can just be a sample covariance matrix, which describes asset volatilities and their co-dependence. This is important because one of the principles of diversification is that risk can be reduced by making many uncorrelated bets (correlation is just normalised covariance).

The semivariance is the variance of all returns which are below some benchmark B (typically the risk-free rate) – it is a common measure of downside risk.

```
[5]: mu = expected_returns.capm_return(df_numeric)
     S = risk_models.semicovariance(df_numeric, benchmark=((1.0392**(1/252))-1))

     ef = EfficientFrontier(mu, S)

     raw_weights = ef.max_sharpe()
     cleaned_weights = ef.clean_weights()
     print(cleaned_weights)
```

```
OrderedDict([('AAPL', 0.1167), ('AMZN', 0.10585), ('GOOG', 0.08325), ('META',
0.08834), ('MSFT', 0.25493), ('NFLX', 0.08199), ('NVDA', 0.15979), ('TSLA',
0.10914)])
```

```
[6]: ef.portfolio_performance(verbose=True)
```

```
Expected annual return: 31.7%
Annual volatility: 23.3%
Sharpe Ratio: 1.27
```

```
[6]: (0.317281920998982, 0.23323342242499903, 1.2746111509579165)
```

Interpreting the Max Sharpe Ratio The Sharpe Ratio (S) is calculated using the formula: $S = \frac{R_p - R_f}{\sigma_p}$

Where: - R_p is the expected portfolio return, - R_f is the risk-free rate, - σ_p is the standard deviation of the portfolio's excess returns (a measure of risk).

- **Higher Sharpe Ratio**: Indicates a more attractive risk-adjusted return. A higher Sharpe ratio means the investment is better compensated for the risk taken.
- **Point on Efficient Frontier**: The Max Sharpe Ratio portfolio is often referred to as the Tangency Portfolio because it is tangent to the Capital Allocation Line (CAL) that intersects the risk-free rate.
- **Investment Decision Tool**: Investors use the Max Sharpe Ratio to choose among various portfolios or to adjust their current portfolios towards this optimal point.

We have set the risk-free rate as 0.02 which is a common rate to pick.

1.1 Environmental Social Governance (ESG) Metrics as an Investment Criteria too

ESG stands for **Environmental, Social, and Governance**, and it is a set of criteria used to evaluate the potential impact of an investment in these key areas:

- **Environmental** criteria consider how a company performs as a steward of nature. This includes issues like the management of waste, pollution, natural resource conservation, and the treatment of animals. The criteria also evaluate any environmental risks a company might face and how those risks are managed.

- **Social** criteria examine how a company manages relationships with employees, suppliers, customers, and the communities where it operates. This includes considerations of human rights, labor standards in the supply chain, employee relations and diversity, and the company's impact on the local communities.

- **Governance** criteria focus on a company's leadership, executive pay, audits, internal controls, and shareholder rights. These criteria help ensure that a company uses accurate and transparent accounting methods, and that shareholders are given the opportunity to vote on important issues.

ESG investing is the practice of incorporating these ESG criteria into investment decisions. This approach helps to identify companies with ethical practices that may lead to better long-term financial performance. It reflects a growing belief that these non-financial factors are significant in identifying risks and opportunities, as well as aligning investments with broader social or ethical values.

We aim to incorporate ESG metrics in our portfolio optimization, and are interested in analyzing ESG metrics in the risk and return space.

```
[7]: esg_data = pd.read_csv('SP 500 ESG Risk Ratings.csv')
     esg_data.set_index('Symbol', inplace=True)
     esg_data = esg_data.dropna(subset=['Total ESG Risk score'])
```

```
[8]: # Arranging ESG scores to be in the same order as the assets in our portfolio
     esg_scores = esg_data.loc[df_numeric.columns.intersection(esg_data.index),␣
      ↪'Total ESG Risk score']
```

```
[9]: # Filtering stock prices to include only the stocks for which we have ESG scores
     df_numeric_filtered = df_numeric[esg_scores.index]
```

```
[10]: mu1 = expected_returns.capm_return(df_numeric_filtered)
      S1 = risk_models.semicovariance(df_numeric_filtered, benchmark=((1.0392**(1/
       ↪252))-1))
```

```
[11]: # Expected annualised returns of individual stocks (that have ESG scores) in␣
       ↪our portfolio
      mu1
```

```
[11]: Ticker
      AAPL    0.264092
      AMZN    0.295504
      META    0.328801
      MSFT    0.261305
      NFLX    0.342124
      NVDA    0.426163
      TSLA    0.451308
      Name: mkt, dtype: float64
```

```
[12]: ef_esg = EfficientFrontier(mu1, S1)
```

Here, we set an overall arbitrary ESG target score for our portfolio. This is a case-based score, and can be decided accordingly. An investor or a client can choose a higher score if they are flexible with their ethical values.

We went with a risk score of 30 simply because it was closer to the mean risk score of our stocks selection.

Note: A higher ESG Risk score of an asset denotes that the asset is worse-off.

```
[13]: # Setting a portfolio ESG risk score target
      max_average_esg_score = 20
```

```
[14]: weights = cp.Variable(len(esg_scores))
```

ESG constraints can be applied on two fronts in portfolio optimization:

1. **Pre-Optimization:** Investors can screen stocks based on ESG scores before allocating weights.
2. **During Optimization:** Investors can allocate portfolio weights to stocks based on ESG scores.

We'll look at the latter first.

Here we constraint the weighted ESG risk score of our value to be below or equal to our target score. Intuitively, it means our portfolio will allocate lesser weight to a stock that has a ESG risk score of over 20, and a higher weight to a stock with scores of less than 20.

[15]:
```
# The weighted ESG risk scores of the stocks should adhere to the portfolio␣
↪target
esg_constraint = cp.sum(cp.multiply(esg_scores, weights)) <=␣
↪max_average_esg_score
```

[16]:
```
# Add the ESG constraint
ef_esg.add_constraint(lambda w: esg_constraint)
```

[17]:
```
# Continue with optimization
weights = ef_esg.max_sharpe()
cleaned_weights_esg = ef_esg.clean_weights()
print(cleaned_weights_esg)
```

```
OrderedDict([('AAPL', 0.13185), ('AMZN', 0.12218), ('META', 0.09089), ('MSFT',
0.24011), ('NFLX', 0.09995), ('NVDA', 0.18007), ('TSLA', 0.13496)])
```

[18]:
```
ef_esg.portfolio_performance(verbose=True)
```

```
Expected annual return: 33.5%
Annual volatility: 24.1%
Sharpe Ratio: 1.31
```

[18]:
```
(0.3353909521792219, 0.24113256092787572, 1.3079567146203757)
```

Our weighted ESG portfolio has a higher sharpe ratio. That's actually a positive sign in this scenario.

[19]:
```
comparison_df = pd.DataFrame({
    'Ticker': list(esg_scores.keys()),
    'Expected Returns': mu1.values,
    'ESG Risk Score': esg_scores.values,
    'Basic Portfolio Weights': [cleaned_weights[ticker] for ticker in␣
↪esg_scores.keys()],
    'ESG Portfolio Weights': [cleaned_weights_esg[ticker] for ticker in␣
↪esg_scores.keys()]
})

comparison_df
```

[19]:
```
  Ticker  Expected Returns  ESG Risk Score  Basic Portfolio Weights  \
0   AAPL          0.264092            17.0                  0.11670
1   AMZN          0.295504            30.0                  0.10585
2   META          0.328801            33.0                  0.08834
3   MSFT          0.261305            15.0                  0.25493
4   NFLX          0.342124            16.0                  0.08199
5   NVDA          0.426163            13.0                  0.15979
6   TSLA          0.451308            29.0                  0.10914

   ESG Portfolio Weights
0                0.13185
1                0.12218
2                0.09089
3                0.24011
4                0.09995
5                0.18007
6                0.13496
```

`PyPortfolioOpt` does not allow for strict inequalities as constraints. Therefore, the <= on the `max_average_esg_score` seems to be forcing stocks with ESG risk scores of less than 20 but high expected returns (like NVDA) to experience a big jump in allocated weights. But upon observing, all stocks except for MSFT (with the lowest expected return) experience a jump. That goes against our intuition. So, why is that?

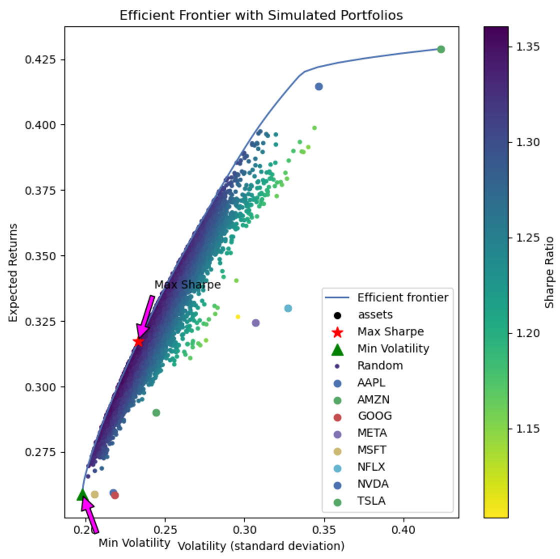
Upon investigating, we can observes that the weighted ESG risk score of basic portfolio is 18.45, whereas that of the ESG portfolio is 20.33 (which is closer to the constraint). Hence, all stocks seem to have higher weights because of that.

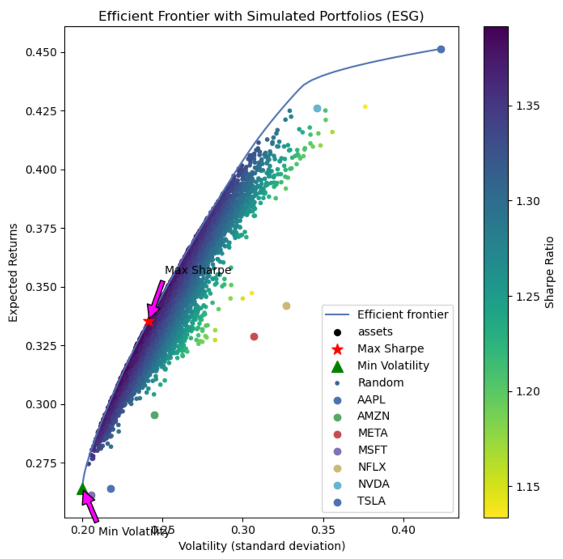
We can visualize the efficient frontier of the two portfolios below.

[20]:
```python
fig, ax = plt.subplots(1, 2, figsize=(14, 7))

# Visualization 1
ef_for_plotting = EfficientFrontier(mu, S)
plotting.plot_efficient_frontier(ef_for_plotting, ax=ax[0], show_assets=True)

ef_max_sharpe = EfficientFrontier(mu, S)
weights = ef_max_sharpe.max_sharpe()
ret_tangent, std_tangent, _ = ef_max_sharpe.portfolio_performance()
ax[0].scatter(std_tangent, ret_tangent, marker="*", s=100, c="r", label="Max␣
 ↪Sharpe", zorder=3)

ef_min_vol = EfficientFrontier(mu, S)
weights_min_vol = ef_min_vol.min_volatility()
ret_min_vol, std_min_vol, _ = ef_min_vol.portfolio_performance()
ax[0].scatter(std_min_vol, ret_min_vol, marker="^", s=100, c="g", label="Min␣
 ↪Volatility", zorder=3)
```

```
# Portfolio Simulations
n_samples = 10000
rand_weights = np.random.dirichlet(alpha=np.ones(ef_for_plotting.n_assets),␣
 ↪size=n_samples)
rand_rets = rand_weights.dot(mu)
rand_stds = np.sqrt(np.diag(rand_weights @ S @ rand_weights.T))
sharpes = rand_rets / rand_stds
scatter = ax[0].scatter(rand_stds, rand_rets, c=sharpes, cmap="viridis_r",␣
 ↪marker='.', label="Random")

cbar = plt.colorbar(scatter, ax=ax[0])
cbar.set_label("Sharpe Ratio")

for ticker in df_numeric.columns:
    ticker_volatility = np.sqrt(S.loc[ticker, ticker])
    ticker_return = mu[ticker]
    ax[0].scatter(ticker_volatility, ticker_return, marker='o', label=ticker,␣
 ↪zorder=2)

ax[0].annotate('Max Sharpe', xy=(std_tangent, ret_tangent),␣
 ↪xytext=(std_tangent+0.01, ret_tangent+0.02),
            arrowprops=dict(facecolor='magenta', shrink=0.05))
ax[0].annotate('Min Volatility', xy=(std_min_vol, ret_min_vol),␣
 ↪xytext=(std_min_vol+0.01, ret_min_vol-0.02),
            arrowprops=dict(facecolor='magenta', shrink=0.05))

ax[0].set_title("Efficient Frontier with Simulated Portfolios")
ax[0].set_xlabel('Volatility (standard deviation)')
ax[0].set_ylabel('Expected Returns')
ax[0].legend()

# Visualization 2
ef_for_esg = EfficientFrontier(mu1, S1)
plotting.plot_efficient_frontier(ef_for_esg, ax=ax[1], show_assets=True)

ef_max_sharpe = EfficientFrontier(mu1, S1)
weights = ef_max_sharpe.max_sharpe()
ret_tangent, std_tangent, _ = ef_max_sharpe.portfolio_performance()
ax[1].scatter(std_tangent, ret_tangent, marker="*", s=100, c="r", label="Max␣
 ↪Sharpe", zorder=3)

ef_min_vol = EfficientFrontier(mu1, S1)
weights_min_vol = ef_min_vol.min_volatility()
ret_min_vol, std_min_vol, _ = ef_min_vol.portfolio_performance()
ax[1].scatter(std_min_vol, ret_min_vol, marker="^", s=100, c="g", label="Min␣
 ↪Volatility", zorder=3)
```

```
rand_weights = np.random.dirichlet(alpha=np.ones(ef_for_esg.n_assets),␣
↪size=n_samples)
rand_rets = rand_weights.dot(mu1)
rand_stds = np.sqrt(np.diag(rand_weights @ S1 @ rand_weights.T))
sharpes = rand_rets / rand_stds
scatter = ax[1].scatter(rand_stds, rand_rets, c=sharpes, cmap="viridis_r",␣
↪marker='.', label="Random")

cbar = plt.colorbar(scatter, ax=ax[1])
cbar.set_label("Sharpe Ratio")

for ticker in df_numeric_filtered.columns:
    ticker_volatility = np.sqrt(S1.loc[ticker, ticker])
    ticker_return = mu1[ticker]
    ax[1].scatter(ticker_volatility, ticker_return, marker='o', label=ticker,␣
↪zorder=2)

ax[1].annotate('Max Sharpe', xy=(std_tangent, ret_tangent),␣
↪xytext=(std_tangent+0.01, ret_tangent+0.02),
            arrowprops=dict(facecolor='magenta', shrink=0.05))
ax[1].annotate('Min Volatility', xy=(std_min_vol, ret_min_vol),␣
↪xytext=(std_min_vol+0.01, ret_min_vol-0.02),
            arrowprops=dict(facecolor='magenta', shrink=0.05))

ax[1].set_title("Efficient Frontier with Simulated Portfolios (ESG)")
ax[1].set_xlabel('Volatility (standard deviation)')
ax[1].set_ylabel('Expected Returns')
ax[1].legend()

plt.tight_layout()
plt.show()
```





We can make the following analysis:

1. Efficient Frontier line is the line that encapsulates all the risk/return possibilities of the portfolio.
2. Portfolios tend to have higher sharpe ratios along the efficient frontier line, since it maximizes return probabilities and minimizes volatility
3. Adding a constraint to our portfolio does not change the efficient frontier line but changes the position of the max sharpe ratio on the line.
4. Max Sharpe ratio portfolio traverses along the eficient frontier line when constraints are changed.
5. Since, the slope of the efficient frontier line is steep to begin with. We can assume the slope to be greater than 1 i.e. there is a greater marginal increase in expected returns for a marginal increase in portfolio volatility. Therefore, when the ESG constraints forced greater stock allocations, the volatility of the portfolio increased. But the returns increased marginally more. This is why we observed the Max Sharpe ratio to move up the efficient frontier line.

1.2 Visualizing an Efficient Surface in the Risk, Return and ESG space

Now let's simulate and visualize portfolios for different ESG target scores.

[21]:
```
portfolio_results = []

# Defining a range for the maximum average ESG score
esg_range = np.linspace(10, 70, 100)

for target_esg in esg_range:
    ef = EfficientFrontier(mu1, S1)
    ef.add_constraint(lambda w: cp.sum(cp.multiply(esg_scores, w)) <=␣
 ↪target_esg)
```

```
    ef.add_constraint(lambda w: cp.sum(cp.multiply(esg_scores, w)) >=␣
↪(target_esg - 10))

    try:
        ef.max_sharpe()
        performance = ef.portfolio_performance()
        cleaned_weights = ef.clean_weights()
        actual_esg_score = np.dot(list(cleaned_weights.values()), esg_scores)

        portfolio_results.append({
            'Risk': performance[1],
            'Return': performance[0],
            'ESG_Score': actual_esg_score
        })
    except Exception as e:
        print(f"Optimization failed for target ESG score {target_esg} with␣
↪error: {e}")
```

```
Optimization failed for target ESG score 10.0 with error: ('Please check your
objectives/constraints or use a different solver.', 'Solver status: infeasible')
Optimization failed for target ESG score 10.606060606060606 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 11.212121212121213 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 11.818181818181818 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 12.424242424242424 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 43.333333333333336 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 43.93939393939394 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 44.54545454545455 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 45.151515151515156 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 45.75757575757576 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
```

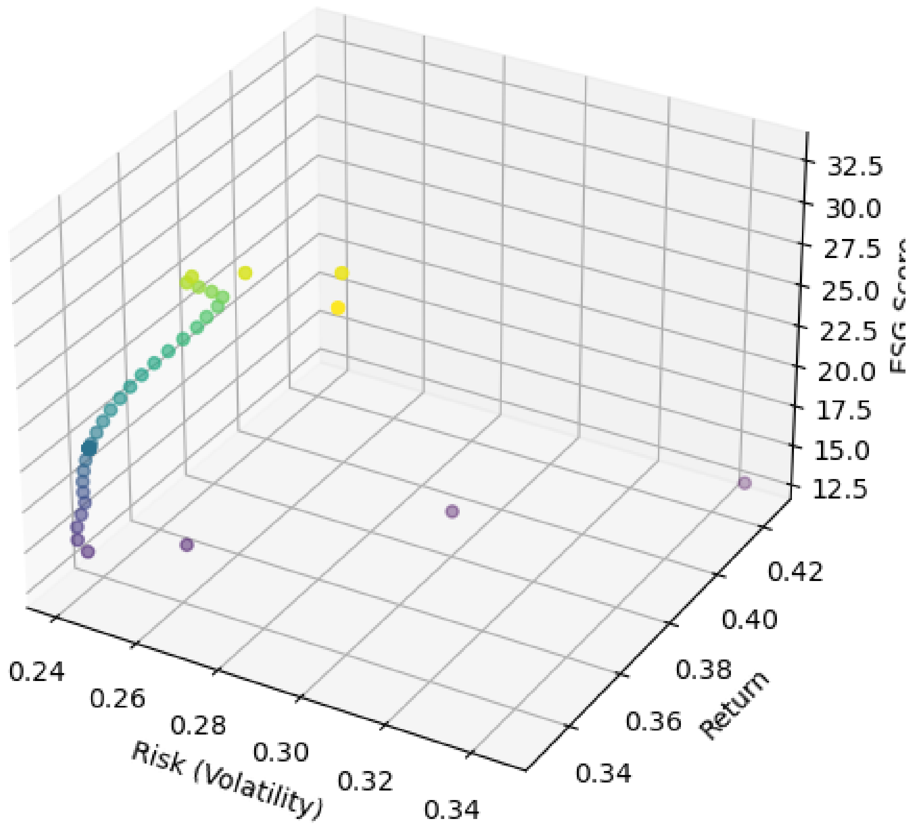
```
Optimization failed for target ESG score 46.36363636363637 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 46.96969696969697 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 47.57575757575758 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 48.18181818181818 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 48.78787878787879 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 49.3939393939394 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 50.0 with error: ('Please check your
objectives/constraints or use a different solver.', 'Solver status: infeasible')
Optimization failed for target ESG score 50.60606060606061 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 51.21212121212121 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 51.81818181818182 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 52.42424242424242 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 53.03030303030303 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 53.63636363636364 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 54.24242424242424 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 54.84848484848485 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 55.45454545454545 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 56.06060606060606 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 56.66666666666667 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 57.27272727272727 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 57.87878787878788 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 58.484848484848484 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 59.09090909090909 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 59.696969696969695 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 60.303030303030305 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 60.909090909090914 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 61.515151515151516 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 62.121212121212125 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 62.72727272727273 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 63.333333333333336 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 63.93939393939394 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 64.54545454545455 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 65.15151515151516 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 65.75757575757575 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 66.36363636363637 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 66.96969696969697 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 67.57575757575758 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 68.18181818181819 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 68.78787878787878 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 69.3939393939394 with error: ('Please
check your objectives/constraints or use a different solver.', 'Solver status:
infeasible')
Optimization failed for target ESG score 70.0 with error: ('Please check your
objectives/constraints or use a different solver.', 'Solver status: infeasible')
```

Error messages are for those portfolios that fail to meet the optimization constraint.

[22]:
```
portfolios_df = pd.DataFrame(portfolio_results)
portfolios_df.head()
```

[22]:
```
       Risk    Return  ESG_Score
0  0.344920  0.425314   13.03030
1  0.297586  0.384172   13.63636
2  0.257681  0.345084   14.24242
3  0.243270  0.330734   14.84851
4  0.241018  0.330426   15.45453
```

[23]:
```
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(portfolios_df['Risk'], portfolios_df['Return'],␣
 ↪portfolios_df['ESG_Score'], c=portfolios_df['ESG_Score'], cmap='viridis')
ax.set_xlabel('Risk (Volatility)')
ax.set_ylabel('Return')
ax.set_zlabel('ESG Score')

plt.show()
```

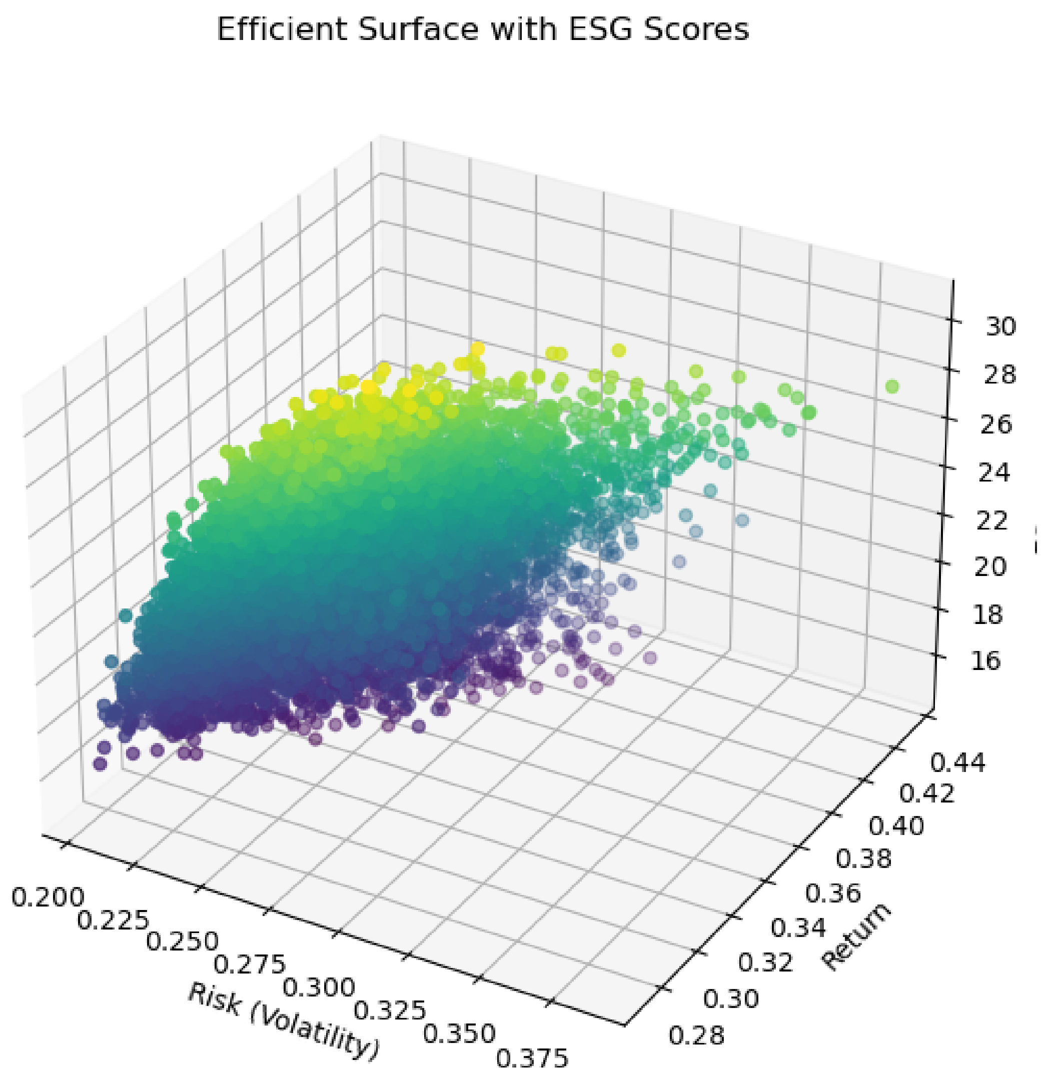


Here, purple portfolios represent lower ESG scores while yellow portfolios represent higher ESG scores. In our simulation, we can somewhat observe an efficient frontier line that made satisfied the constraints. But we don't observe an efficient surface.

To plot the effience surface, let's simulate portfolios without any constraints. Then we can calculate weighted risk, return and ESG scores of the portfolio simulations that can be plotted in a 3D space.

```
[24]: n_samples = 10000
n_assets = len(mu1)

rand_weights = np.random.dirichlet(alpha=np.ones(n_assets), size=n_samples)
mu1 = np.array(mu1)
esg_scores = np.array(esg_scores)

# Monte Carlo Simulations of portfolios
portfolio_returns = np.dot(rand_weights, mu1)
portfolio_risks = np.sqrt(np.einsum('ij,ji->i', rand_weights @ S1, rand_weights.
  ↪T))
portfolio_esg_scores = np.dot(rand_weights, esg_scores)
```

```
portfolios_df = pd.DataFrame({
    'Risk': portfolio_risks,
    'Return': portfolio_returns,
    'ESG_Score': portfolio_esg_scores
})

fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(portfolios_df['Risk'], portfolios_df['Return'],␣
 ↪portfolios_df['ESG_Score'],
           c=portfolios_df['ESG_Score'], cmap='viridis', marker='o')
ax.set_xlabel('Risk (Volatility)')
ax.set_ylabel('Return')
ax.set_zlabel('ESG Score')
ax.set_title('Efficient Surface with ESG Scores')

plt.show()
```



Now we can clearly observe an efficient surface in a 3D space.

Here too, purple portfolios represent lower ESG scores while yellow portfolios represent higher ESG scores.

Analysis: 1. Lower ESG score weighted portfolios tend to offer lower returns, with no benefit of risk mitigation. This is not a generalization but merely an observation for our chosen set of stocks. The stocks we have chosen, average an ESG score of atleast 20. Hence, allocating stocks to hit a lower target score will tend to exclude all stocks with higher returns. 2. Highest ESG score weighted portfolios not only tend to offer higher returns, but also tend to minimise risk. For an ESG conscious investor, this analysis suggest that our chosen set of stocks do not yield the maximum risk-adjusted returns. 3. Here, lower ESG score weighted portfolios tend to be risky. Hence, an ethically conscious investor is better off staying just under the yellow portfolios, and around the darker gradient portfolios. This assures a balanced approach.

Overall, an Effiient Surface allows an investor to further analyse different perspectives on their investment strategies.

1.3 Practical Applications of ESG-Integrated with Portfolio Optimization

1. **Enhanced Portfolio Screening:**
    - Investors can utilize ESG scores to filter out low-performing companies in terms of ESG criteria before performing portfolio optimization. This pre-optimization step ensures that only companies meeting the desired ESG standards are included, enhancing the ethical alignment of the portfolio.
2. **Optimized Weight Allocation:**
    - During the optimization process, ESG scores can be incorporated to adjust the weights of individual assets. Companies with higher ESG scores can be allocated higher weights, while those with lower scores receive less emphasis. This method allows the optimization model to balance financial returns with ESG performance.
3. **Setting ESG Constraints:**
    - Investors can set specific constraints related to ESG performance within the optimization framework. For example, requiring the portfolio to achieve a minimum average ESG score or limiting exposure to industries with high environmental risks. This ensures compliance with ESG standards while optimizing for returns and risk.
4. **Dual Objective Optimization:**
    - The optimization algorithm can be modified to simultaneously optimize for financial returns and ESG performance. This dual objective approach ensures that the portfolio not only seeks to maximize returns but also meets predefined ESG goals, creating a balanced investment strategy. Instead of `PyPortfolioOpt`, we can customize mean-variance optimization algorithms or use integer programming algorithms.
5. **Dynamic Portfolio Rebalancing:**
    - ESG scores can guide the dynamic rebalancing of the portfolio. As ESG scores and financial metrics are updated, the portfolio can be adjusted to maintain alignment with ESG objectives, ensuring continuous adherence to sustainability goals.
6. **Stress Testing with ESG Scenarios:**
    - By incorporating ESG scenarios into stress testing, investors can evaluate how the portfolio performs under various conditions, such as regulatory changes or environmental events. This analysis helps in understanding the portfolio's resilience and adaptability to ESG-related risks.

This project serves as a foundational demonstration of how the above-listed practical applications can be acieved and analyzed.

```
[ ]:
```