Assignment 1 Report

1. Objective

This project implements a **Mean-Variance Portfolio (MVP)** optimization framework using fundamental stock selection within the S&P 500 universe. The goal is to identify efficient portfolios that balance return and risk, visualize the efficient frontier, and evaluate performance through backtesting.

---

2. Data and Pipeline Overview

Step 0 — Fundamental Stock Selection

The first step of the workflow generates the `stock_selected.csv` file, which contains the list of selected S&P 500 stocks based on fundamental criteria.

This process is implemented in **`stock_selection.py`**, which reads raw financial data from `final_ratios.csv` and applies filters on key accounting ratios, such as:

Metric	Filter Logic	Example Threshold
P/E Ratio	Positive and below 40	`0 < PE < 40`
ROE	Above industry median	`ROE > median(ROE)`
Debt-to-Equity	Below 1	`DE < 1`
EPS Growth	Positive trend over last 3 years	`ΔEPS > 0`

Each stock that satisfies all conditions is kept for further analysis.

Output:
The script exports `./result/stock_selected.csv`, which serves as the input for the portfolio construction stage (`fundamental_portfolio.py`).

Example Command:

```
python source_codes/stock_selection.py \
  --input_data ./data/final_ratios.csv \
  --output_path ./result/stock_selected.csv
```

> In this project, the selection was based on the raw `gvkey` identifier (not converted to `tic`), consistent throughout all subsequent steps.

Input Datasets

- `sp500_tickers_daily_price.csv`: daily price history of S&P 500 constituents.
- `stock_selected.csv`: selected stocks based on fundamental screening.

Generated Outputs

- `portfolio_output/` directory containing:
  - `Result_Metrics.json`: summary of backtest performance.
  - Portfolio weight files:
    - `mean_weighted.xlsx`
    - `minimum_weighted.xlsx`
    - `equally_weighted.xlsx`
  - `efficient_frontier.png`: visualization of efficient frontier.
- Backtest visualizations:
  - `Portfolio_Values.png`
  - `mean portfolio.png`
  - `MVP portfolio.png`

Pipeline Execution

```
# Step 1: Stock Selection
python source_codes/stock_selection.py

# Step 2: Portfolio Construction
python source_codes/fundamental_portfolio.py \
  --stocks_price "../sp500_tickers_daily_price.csv" \
  --stock_selected "./result/stock_selected.csv" \
  --output_dir "./portfolio_output"

# Step 3: Plot Efficient Frontier
python source_codes/plot_efficient_frontier.py --output_dir
./portfolio_output
```

---

3. Methodology

3.1 Mean–Variance Portfolio (MVP) Model

The optimization problem minimizes total portfolio variance under budget and non-negative constraints:

$$\min_{w} \; w^T \Sigma \, w \quad \text{s.t.} \quad \sum_i w_i = 1, \; w_i \geq 0$$

where:

- Σ = covariance matrix of returns
- w = portfolio weights

3.2 Core MVP Implementation

Below shows the essential logic used in the optimization routine:

```python
import numpy as np
from scipy.optimize import minimize

def get_mvp_weights(cov_matrix):
    n = cov_matrix.shape[0]
    init_w = np.ones(n) / n

    def portfolio_variance(w):
        return w.T @ cov_matrix @ w

    constraints = {'type': 'eq', 'fun': lambda w: np.sum(w) - 1}
    bounds = [(0, 1) for _ in range(n)]

    result = minimize(portfolio_variance, init_w,
                          method='SLSQP', bounds=bounds,
constraints=constraints)
    return result.x / np.sum(result.x)
```

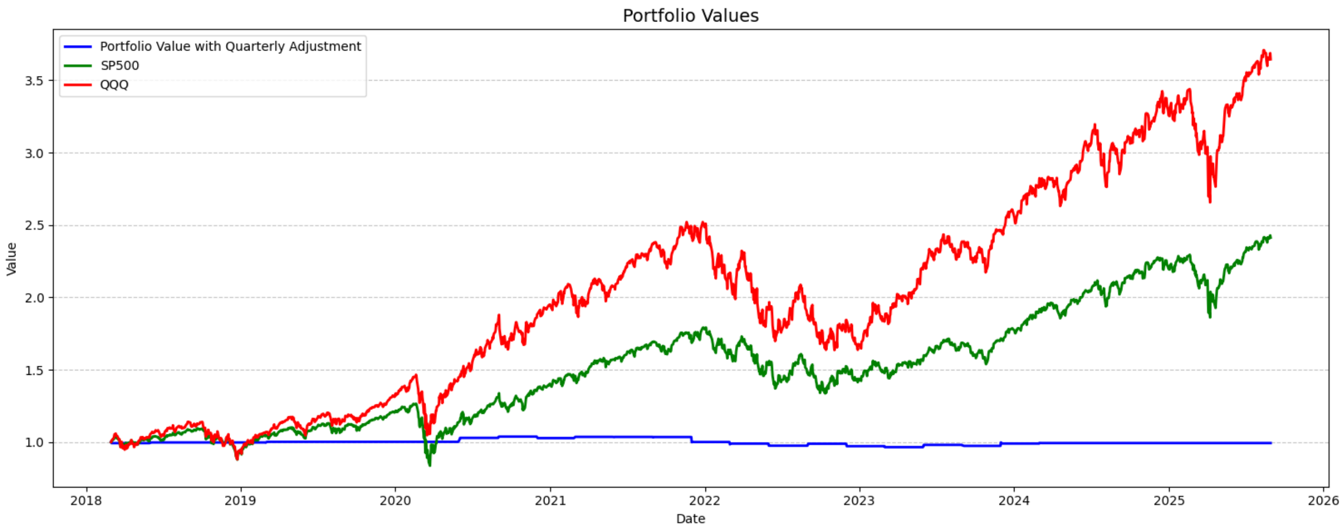
This function calculates the minimum-variance weights using **SciPy's SLSQP solver**, ensuring no short sales and full investment.

---

4. Results

4.1 Portfolio Backtest Performance

Backtest Period: 2018-01-01 to 2025-10-01



Metric	Value
Cumulative Return	-0.0073
Annual Return	-0.00098
Max Drawdown	-0.0702
Annual Volatility	0.0221

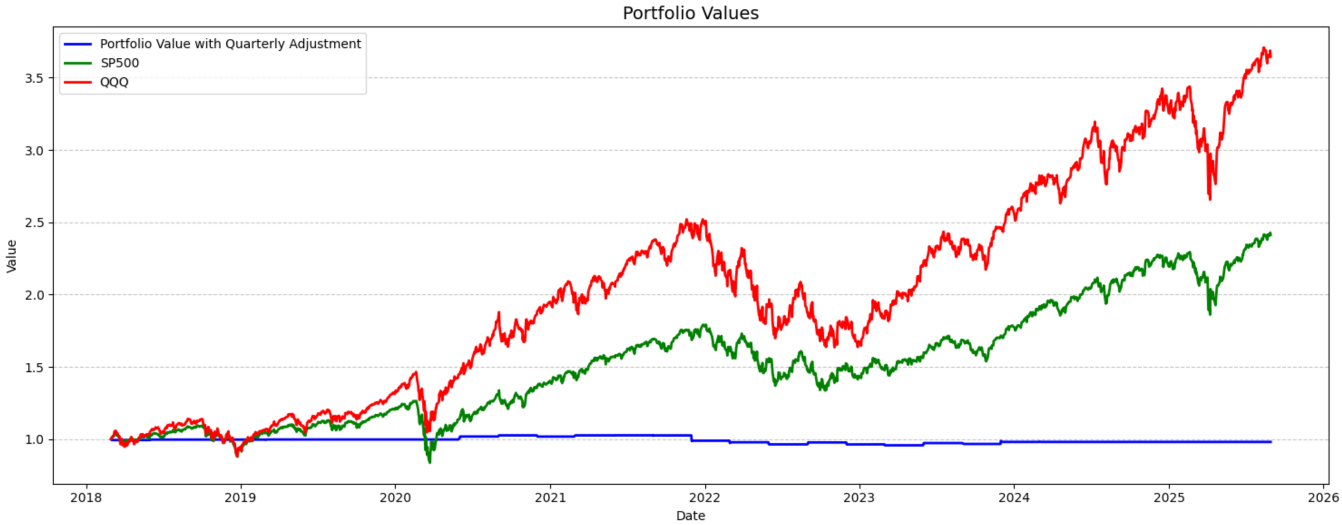
Metric	Value
Sharpe Ratio	-0.0333
Win Rate	0.5135
Information Ratio	-0.6952

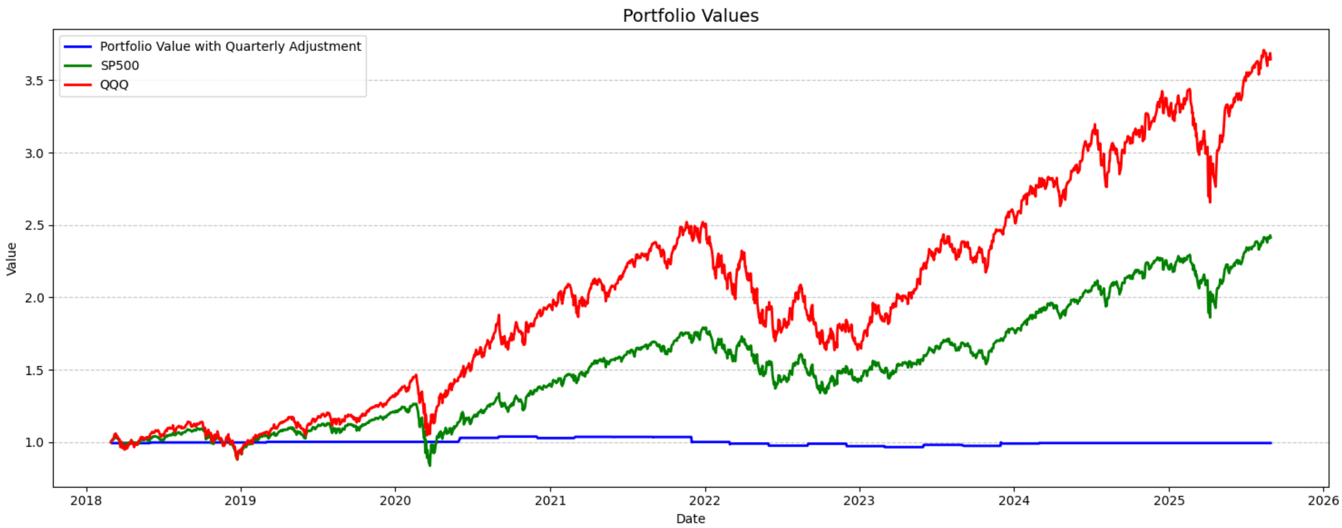
> The MVP strategy maintained low volatility but achieved slightly negative cumulative returns. Its defensive nature results in stability but limited upside compared to benchmarks like S&P 500.

4.2 Comparison Across Weighting Schemes

Portfolio Type	Cumulative Return	Annual Volatility	Sharpe Ratio
Minimum Variance	-0.0073	0.022	-0.03
Mean Weighted	-0.0197	0.0215	-0.11
Equal Weighted	-0.0073	0.0221	-0.03

Performance Visualization:





5. Discussion

- The MVP achieved the lowest volatility among all portfolios but underperformed in total return.
- The mean-weighted portfolio showed slightly better stability with modest improvement in win rate but remained negative overall.
- The model's conservative allocation and lack of predictive expected returns constrained upside potential.
- Potential improvements:
  - Incorporate factor-based expected returns (value, momentum, quality).
  - Relax constraints (e.g., allow limited short selling).
  - Integrate FinGPT/DRL modules to enhance dynamic rebalancing.

---

6. Deliverables Summary

File	Description
`fundamental_portfolio.py`	Core MVP calculation logic
`plot_efficient_frontier.py`	Efficient frontier visualization
`Portfolio_Values.png`	Backtest performance graph
`Result_Metrics.json`	Quantitative results summary
`mean_weighted.xlsx` / `minimum_weighted.xlsx` / `equally_weighted.xlsx`	Final portfolio weights

---

7. Requirements

```
pandas
numpy
scipy
matplotlib
pypfopt
```

---

8. Conclusion

The MVP framework successfully constructs stable, low-risk portfolios and replicates the efficient frontier. While performance remains conservative, the workflow establishes a strong foundation for incorporating predictive or explainable AI modules in future iterations.
