Question 3 (50 Points)

In this exercise, you will use weekly return data for 50 US stocks. The data is provided to you in `returns.pkl`. The objective is to construct a `minimum variance portfolio` (MVP) using two different methods to estimate the covariance matrix and evaluate the performance of the resulting portfolios.

Instructions:

1. **Covariance Estimation**: Using a rolling window of 100 weeks (including the current week), estimate the covariance matrix of stock returns every week using:

   - The sample covariance matrix (unadjusted).
   - The Ledoit and Wolf shrinkage estimator.

   To estimate the covariance matrix using Ledoit-Wolf shrinkage, use the `LedoitWolf` module in Python's `sklearn.covariance`. Make sure to carefully read the documentation to understand its input requirements. The document can be found at: LedoitWolf Documentation.

2. **Portfolio Construction**: For each week, construct a minimum variance portfolio (MVP) using both covariance estimation methods. The weights of the portfolio should minimize the variance of portfolio returns, subject to the constraint that the portfolio weights sum to 1. Short positions are allowed, meaning the weights of some stocks in the portfolio can be negative. Repeat this process for each week in the dataset to generate out-of-sample (OOS) returns for both MVPs.

3. **Performance Comparison**: Calculate the standard deviation of the OOS returns for both portfolios. Compare the two portfolios in terms of their OOS standard deviations. Discuss whether the portfolio constructed using the Ledoit-Wolf covariance matrix has a lower OOS standard deviation compared to the portfolio constructed using the sample covariance matrix. In your discussion, provide an interpretation of the results in terms of the accuracy of covariance matrix estimation and its impact on portfolio risk.

Please submit your Python code and ensure that it is clear and well-commented.

Part 1: Covariance Estimation

```
In [ ]: import pandas as pd
        import numpy as np
        from sklearn.covariance import LedoitWolf

        # Load the weekly return data
        returns = pd.read_pickle('returns.pkl')  # Ensure the file path is correct

        # Init variables and list to store cov matrix
        window_size = 100
        sample_cov_matrices = []
        ledoit_wolf_cov_matrices = []

        # Rolling window covariance estimation
        for start in range(len(returns) - window_size + 1):
            end = start + window_size
            window_data = returns.iloc[start:end]

            # Sample covariance matrix
            sample_cov = window_data.cov()
            sample_cov_matrices.append(sample_cov)

            # Ledoit-Wolf shrinkage estimator
            lw = LedoitWolf(assume_centered=False)
            lw_cov = lw.fit(window_data).covariance_
            ledoit_wolf_cov_matrices.append(pd.DataFrame(lw_cov, index=window_data.columns, columns=window_data.columns))
```

Part 2: Portfolio Construction

```
In [28]: def minimum_variance_portfolio(cov_matrix):
             """Constructs a minimum variance portfolio given a covariance matrix."""
             inv_cov_matrix = np.linalg.inv(cov_matrix)
             ones = np.ones(inv_cov_matrix.shape[0])
             weights = inv_cov_matrix @ ones
             weights /= weights.sum()  # Normalize weights to sum to 1
             return weights
```

```
In [ ]: # Initialize lists to store portfolio weights
        sample_mvp_weights = []
```

```python
ledoit_wolf_mvp_weights = []

# Construct MVP for each week
for i in range(len(sample_cov_matrices)):
    if i < len(returns) - window_size + 1:  # Ensure we have a valid covariance matrix
        sample_cov = sample_cov_matrices[i].values
        ledoit_cov = ledoit_wolf_cov_matrices[i].values

        # print(sample_cov) # debugging statements
        # break

        # Calculate minimum variance portfolio weights
        sample_weights = minimum_variance_portfolio(sample_cov)
        ledoit_weights = minimum_variance_portfolio(ledoit_cov)

        sample_mvp_weights.append(sample_weights)
        ledoit_wolf_mvp_weights.append(ledoit_weights)
```

In [36]:
```python
# Convert weights to DataFrame for easier manipulation
sample_mvp_weights = pd.DataFrame(sample_mvp_weights, columns=returns.columns)
ledoit_wolf_mvp_weights = pd.DataFrame(ledoit_wolf_mvp_weights, columns=returns.columns)
```

Part 3: Performance Comparison

In [37]:
```python
# Calculate out-of-sample returns
sample_oos_returns = (returns.iloc[window_size:] @ sample_mvp_weights.T).values
ledoit_oos_returns = (returns.iloc[window_size:] @ ledoit_wolf_mvp_weights.T).values

# Calculate standard deviation of OOS returns
sample_std_dev = np.std(sample_oos_returns)
ledoit_std_dev = np.std(ledoit_oos_returns)

# Print results
print(f"Sample Covariance MVP OOS Standard Deviation: {sample_std_dev:.4f}")
print(f"Ledoit-Wolf MVP OOS Standard Deviation: {ledoit_std_dev:.4f}")

# Discussion on results
if ledoit_std_dev < sample_std_dev:
    print("The Ledoit-Wolf covariance matrix results in a lower OOS standard deviation.")
```

```python
else:
    print("The sample covariance matrix results in a lower OOS standard deviation.")
```

```
Sample Covariance MVP OOS Standard Deviation: 0.0253
Ledoit-Wolf MVP OOS Standard Deviation: 0.0208
The Ledoit-Wolf covariance matrix results in a lower OOS standard deviation.
```

Discussion & Interpretation of Results

1. Potential cause of lower OOS standard deviation via Ledoit-Wolf Covariance Matrix

   - Ledoit-Wolf shrinkage estimator introduces a data-driven shrinking factor that regulates influence of noise and extreme values in the data.
   - This improves stability of estimates, without sacrificing flexibility, while adapting to complex data structures.
   - This is achieved by introducing bias (shrinking estimates), but is outweighed by signigicant reduction in variance.

2. Impact on Portfolio Risk

   - By reducing variance, the Ledoit-Wolf covariance matrix is better at capturing underlying structure of the data, this can lead to more accurate risk estimates in a given portfolio.
   - As mentioned in the lecture slides, the Ledoit-Wolf MVP does exhibit lower OOS voltility (std dev), hence implying a more accurate estimator.
   - In portfolio construction, the Ledoit-Wolf approach produces a more accurate covariance matrix, resulting in weights that more correctly reflect the true risk of assets.

   Thus, Ledoit-Wolf estimates often generate lower risk (lower std dev = lower volatility), that are also more robust when dealing with high dimensionality financial data.