main

December 11, 2024

Install necessary packages

```
[ ]: pip install -r /workspaces/uzh-digfintools-research/resources/requirementsshort.
      ↪txt
```

```
Requirement already satisfied: yfinance in
/home/codespace/.python/current/lib/python3.12/site-packages (from -r
/workspaces/uzh-digfintools-research/requirementsshort.txt (line 1)) (0.2.45)
Requirement already satisfied: pandas in
/home/codespace/.local/lib/python3.12/site-packages (from -r /workspaces/uzh-
digfintools-research/requirementsshort.txt (line 2)) (2.2.3)
Requirement already satisfied: numpy in
/home/codespace/.local/lib/python3.12/site-packages (from -r /workspaces/uzh-
digfintools-research/requirementsshort.txt (line 3)) (2.1.1)
Requirement already satisfied: matplotlib-inline in
/home/codespace/.local/lib/python3.12/site-packages (from -r /workspaces/uzh-
digfintools-research/requirementsshort.txt (line 4)) (0.1.7)
Requirement already satisfied: matplotlib in
/home/codespace/.local/lib/python3.12/site-packages (from -r /workspaces/uzh-
digfintools-research/requirementsshort.txt (line 5)) (3.9.2)
Requirement already satisfied: seaborn in
/home/codespace/.local/lib/python3.12/site-packages (from -r /workspaces/uzh-
digfintools-research/requirementsshort.txt (line 6)) (0.13.2)
Requirement already satisfied: cvxopt in
/home/codespace/.python/current/lib/python3.12/site-packages (from -r
/workspaces/uzh-digfintools-research/requirementsshort.txt (line 7)) (1.3.2)
Requirement already satisfied: requests>=2.31 in
/home/codespace/.local/lib/python3.12/site-packages (from yfinance->-r
/workspaces/uzh-digfintools-research/requirementsshort.txt (line 1)) (2.32.3)
Requirement already satisfied: multitasking>=0.0.7 in
/home/codespace/.python/current/lib/python3.12/site-packages (from yfinance->-r
/workspaces/uzh-digfintools-research/requirementsshort.txt (line 1)) (0.0.11)
Requirement already satisfied: lxml>=4.9.1 in
/home/codespace/.python/current/lib/python3.12/site-packages (from yfinance->-r
/workspaces/uzh-digfintools-research/requirementsshort.txt (line 1)) (5.3.0)
Requirement already satisfied: platformdirs>=2.0.0 in
/home/codespace/.local/lib/python3.12/site-packages (from yfinance->-r
/workspaces/uzh-digfintools-research/requirementsshort.txt (line 1)) (4.3.6)
```

```
Requirement already satisfied: pytz>=2022.5 in
/home/codespace/.local/lib/python3.12/site-packages (from yfinance->-r
/workspaces/uzh-digfintools-research/requirementsshort.txt (line 1)) (2024.2)
Requirement already satisfied: frozendict>=2.3.4 in
/home/codespace/.python/current/lib/python3.12/site-packages (from yfinance->-r
/workspaces/uzh-digfintools-research/requirementsshort.txt (line 1)) (2.4.6)
Requirement already satisfied: peewee>=3.16.2 in
/home/codespace/.python/current/lib/python3.12/site-packages (from yfinance->-r
/workspaces/uzh-digfintools-research/requirementsshort.txt (line 1)) (3.17.7)
Requirement already satisfied: beautifulsoup4>=4.11.1 in
/home/codespace/.local/lib/python3.12/site-packages (from yfinance->-r
/workspaces/uzh-digfintools-research/requirementsshort.txt (line 1)) (4.12.3)
Requirement already satisfied: html5lib>=1.1 in
/home/codespace/.python/current/lib/python3.12/site-packages (from yfinance->-r
/workspaces/uzh-digfintools-research/requirementsshort.txt (line 1)) (1.1)
Requirement already satisfied: python-dateutil>=2.8.2 in
/home/codespace/.local/lib/python3.12/site-packages (from pandas->-r
/workspaces/uzh-digfintools-research/requirementsshort.txt (line 2))
(2.9.0.post0)
Requirement already satisfied: tzdata>=2022.7 in
/home/codespace/.local/lib/python3.12/site-packages (from pandas->-r
/workspaces/uzh-digfintools-research/requirementsshort.txt (line 2)) (2024.2)
Requirement already satisfied: traitlets in
/home/codespace/.local/lib/python3.12/site-packages (from matplotlib-inline->-r
/workspaces/uzh-digfintools-research/requirementsshort.txt (line 4)) (5.14.3)
Requirement already satisfied: contourpy>=1.0.1 in
/home/codespace/.local/lib/python3.12/site-packages (from matplotlib->-r
/workspaces/uzh-digfintools-research/requirementsshort.txt (line 5)) (1.3.0)
Requirement already satisfied: cycler>=0.10 in
/home/codespace/.local/lib/python3.12/site-packages (from matplotlib->-r
/workspaces/uzh-digfintools-research/requirementsshort.txt (line 5)) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/home/codespace/.local/lib/python3.12/site-packages (from matplotlib->-r
/workspaces/uzh-digfintools-research/requirementsshort.txt (line 5)) (4.54.1)
Requirement already satisfied: kiwisolver>=1.3.1 in
/home/codespace/.local/lib/python3.12/site-packages (from matplotlib->-r
/workspaces/uzh-digfintools-research/requirementsshort.txt (line 5)) (1.4.7)
Requirement already satisfied: packaging>=20.0 in
/home/codespace/.local/lib/python3.12/site-packages (from matplotlib->-r
/workspaces/uzh-digfintools-research/requirementsshort.txt (line 5)) (24.1)
Requirement already satisfied: pillow>=8 in
/home/codespace/.local/lib/python3.12/site-packages (from matplotlib->-r
/workspaces/uzh-digfintools-research/requirementsshort.txt (line 5)) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/home/codespace/.local/lib/python3.12/site-packages (from matplotlib->-r
/workspaces/uzh-digfintools-research/requirementsshort.txt (line 5)) (3.1.4)
Requirement already satisfied: soupsieve>1.2 in
/home/codespace/.local/lib/python3.12/site-packages (from
beautifulsoup4>=4.11.1->yfinance->-r /workspaces/uzh-digfintools-
research/requirementsshort.txt (line 1)) (2.6)
Requirement already satisfied: six>=1.9 in
/home/codespace/.local/lib/python3.12/site-packages (from
html5lib>=1.1->yfinance->-r /workspaces/uzh-digfintools-
research/requirementsshort.txt (line 1)) (1.16.0)
Requirement already satisfied: webencodings in
/home/codespace/.local/lib/python3.12/site-packages (from
html5lib>=1.1->yfinance->-r /workspaces/uzh-digfintools-
research/requirementsshort.txt (line 1)) (0.5.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/home/codespace/.local/lib/python3.12/site-packages (from
requests>=2.31->yfinance->-r /workspaces/uzh-digfintools-
research/requirementsshort.txt (line 1)) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/home/codespace/.local/lib/python3.12/site-packages (from
requests>=2.31->yfinance->-r /workspaces/uzh-digfintools-
research/requirementsshort.txt (line 1)) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/home/codespace/.local/lib/python3.12/site-packages (from
requests>=2.31->yfinance->-r /workspaces/uzh-digfintools-
research/requirementsshort.txt (line 1)) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/home/codespace/.local/lib/python3.12/site-packages (from
requests>=2.31->yfinance->-r /workspaces/uzh-digfintools-
research/requirementsshort.txt (line 1)) (2024.8.30)

[notice] A new release of pip is
available: 24.2 -> 24.3.1
[notice] To update, run:
pip install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

Equity data - D&J 60 from 2019-11-01 to 2020-11-01 (source: finance yahoo)

[77]:

```
import yfinance as yf
import pandas as pd
import os

tickers = ["MMM", "AXP", "AAPL", "BA", "CAT", "CVX", "CSCO", "KO", "DOW",␣
 ↪"XOM", "GS", "HD", "IBM", "INTC", "JNJ", "JPM", "MCD", "MRK", "MSFT", "NKE",␣
 ↪"PFE", "PG", "TRV", "UNH", "VZ", "V", "WBA", "WMT", "DIS", "RTX"]

start_date = '2019-11-01'
end_date = '2020-11-01'

data = yf.download(tickers, start=start_date, end=end_date,␣
 ↪interval='1d')['Close']
```

```
#Gather industry data
industry_data = []
for ticker in tickers:
  stock = (yf.Ticker(ticker)).info
  info = {
      'Ticker': ticker,
      'Industry': stock.get('industry', 'N/A'),
  }
  industry_data.append(info)
industry_df = pd.DataFrame(industry_data)
```

```
[*********************100%***********************]  30 of 30 completed
```

Risk-free data - T-bill 10Y yield for the same time period (source: finance yahoo)

[55]:

```
import yfinance as yf
import pandas as pd
import os

ticker = "^TNX"

y10_data = yf.download(ticker, start=start_date, end=end_date,␣
  ↪interval='1d')['Close']
y10_data.head()
rf_rate = y10_data/100

descriptive_stats = rf_rate.describe()
descriptive_stats.head()
```

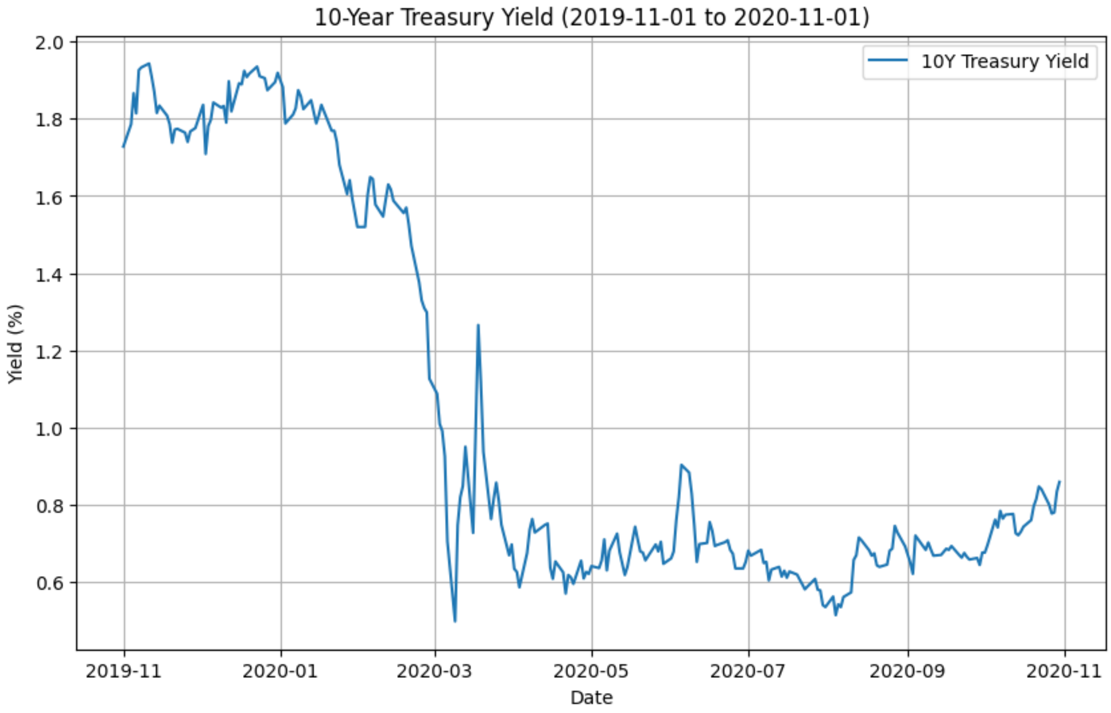
```
[*********************100%***********************]  1 of 1 completed
```

[55]:

```
Ticker        ^TNX
count   252.000000
mean      0.010363
std       0.005018
min       0.004990
25%       0.006627
```

[56]:

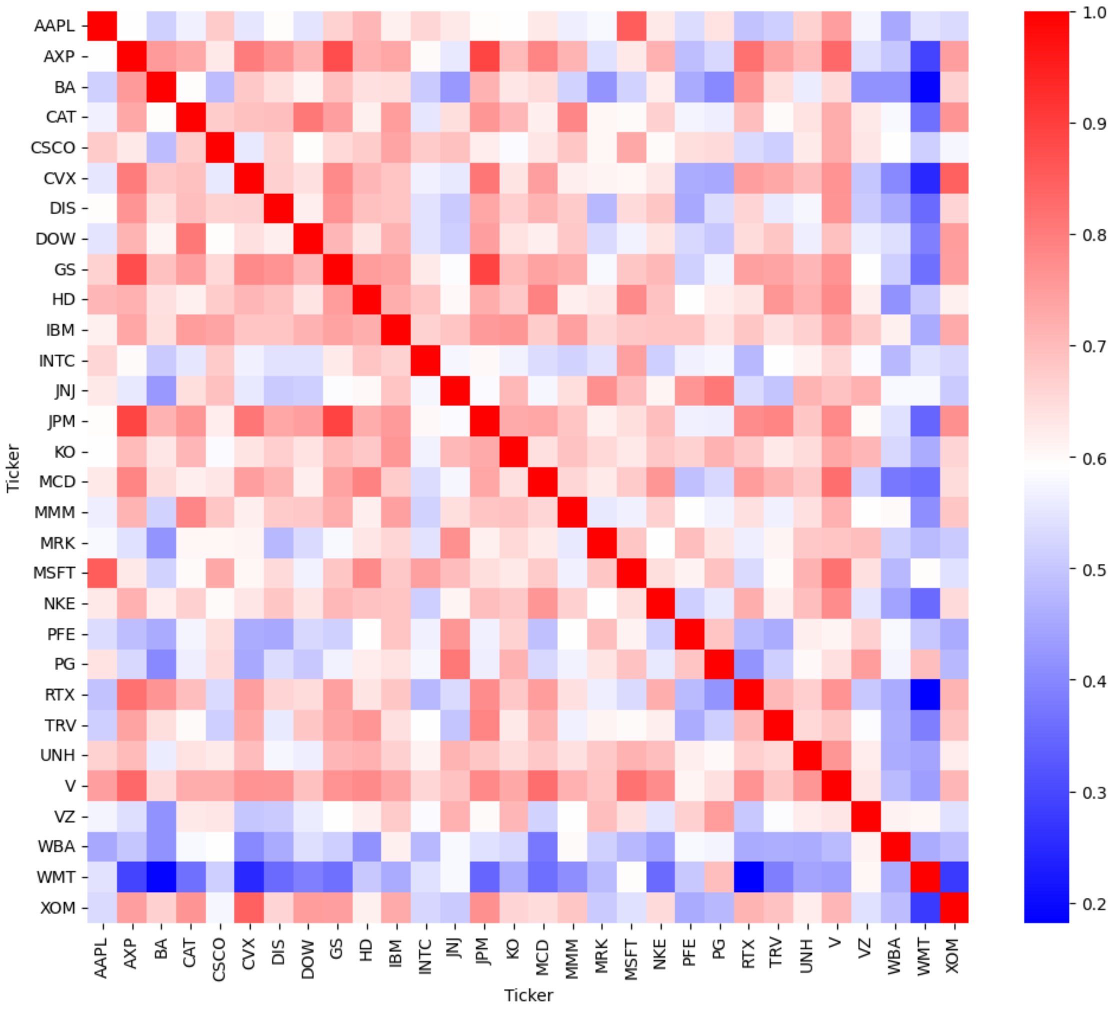
```
import matplotlib.pyplot as plt

plt.figure(figsize=(10,6))
plt.plot(rf_rate*100, label='10Y Treasury Yield')
plt.title('10-Year Treasury Yield (2019-11-01 to 2020-11-01)')
plt.xlabel('Date')
plt.ylabel('Yield (%)')
plt.legend()
plt.grid(True)
plt.show()
```



[86]:
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import os

# Compute log returns
log_returns = np.log(data / data.shift(1)).dropna()

# Descriptive statistics (First 4 moments)
mean_returns = log_returns.mean()
variance = log_returns.var()
skewness = log_returns.skew()
kurtosis = log_returns.kurtosis()

# Combine the descriptive statistics into a DataFrame
descriptive_stats = pd.DataFrame({
    'Mean (%)': mean_returns*100,
    'Variance (%)': variance*100,
    'Skewness': skewness,
    'Kurtosis': kurtosis
})
```

```
# Associate with industries
descriptive_stats = descriptive_stats.merge(
    industry_df, how='left', left_on='Ticker', right_on='Ticker'
)

# Sort by Industry and Ticker for organization
descriptive_stats = descriptive_stats.sort_values(by=['Industry', 'Ticker'])

# Calculate the correlation matrix
correlation_matrix = log_returns.corr()

# Display descriptive statistics
#print("\nDescriptive Statistics (First 4 moments):\n", descriptive_stats)

#Export to Latex table
descriptive_latex = descriptive_stats.to_latex(index=False,na_rep='',␣
 ↪float_format="%.2f")
print(descriptive_latex)


correlation_matrix = log_returns.corr()

# Set up the matplotlib figure
plt.figure(figsize=(12, 10))

# Draw the heatmap
sns.heatmap(correlation_matrix, annot=False, cmap='bwr') #account for␣
 ↪colorblind color palette

# Display the plot
plt.show()
```

```
\begin{tabular}{lrrrrl}
\toprule
Ticker & Mean (%) & Variance (%) & Skewness & Kurtosis & Industry \\
\midrule
BA & -0.35 & 0.29 & -0.32 & 6.21 & Aerospace & Defense \\
RTX & -0.21 & 0.12 & -0.17 & 5.10 & Aerospace & Defense \\
JPM & -0.11 & 0.11 & -0.21 & 6.74 & Banks - Diversified \\
KO & -0.05 & 0.05 & -0.77 & 4.14 & Beverages - Non-Alcoholic \\
GS & -0.06 & 0.10 & -0.15 & 6.42 & Capital Markets \\
DOW & -0.06 & 0.15 & -1.13 & 9.67 & Chemicals \\
CSCO & -0.11 & 0.07 & -0.37 & 6.53 & Communication Equipment \\
MMM & -0.02 & 0.06 & -0.19 & 4.95 & Conglomerates \\
AAPL & 0.21 & 0.08 & -0.34 & 4.41 & Consumer Electronics \\
AXP & -0.11 & 0.14 & 0.40 & 7.39 & Credit Services \\
V & 0.00 & 0.07 & -0.14 & 7.46 & Credit Services \\
WMT & 0.07 & 0.04 & 0.95 & 9.24 & Discount Stores \\
JNJ & 0.02 & 0.04 & 0.19 & 5.48 & Drug Manufacturers - General \\
MRK & -0.05 & 0.04 & -0.11 & 4.33 & Drug Manufacturers - General \\
PFE & -0.03 & 0.04 & -0.25 & 3.95 & Drug Manufacturers - General \\
DIS & -0.04 & 0.08 & -0.09 & 5.78 & Entertainment \\
CAT & 0.03 & 0.08 & -0.81 & 5.03 & Farm & Heavy Construction Machinery \\
NKE & 0.12 & 0.07 & -0.19 & 7.83 & Footwear & Accessories \\
UNH & 0.08 & 0.09 & -0.81 & 9.28 & Healthcare Plans \\
HD & 0.05 & 0.08 & -1.93 & 18.19 & Home Improvement Retail \\
PG & 0.04 & 0.04 & 0.12 & 8.23 & Household & Personal Products \\
IBM & -0.08 & 0.07 & -0.51 & 5.60 & Information Technology Services \\
TRV & -0.03 & 0.10 & -2.17 & 16.60 & Insurance - Property & Casualty \\
CVX & -0.20 & 0.14 & -1.12 & 14.37 & Oil & Gas Integrated \\
XOM & -0.30 & 0.10 & -0.21 & 3.27 & Oil & Gas Integrated \\
WBA & -0.21 & 0.08 & -0.01 & 3.31 & Pharmaceutical Retailers \\
MCD & 0.04 & 0.06 & -0.29 & 18.11 & Restaurants \\
INTC & -0.10 & 0.11 & -0.87 & 11.89 & Semiconductors \\
MSFT & 0.14 & 0.07 & -0.46 & 7.64 & Software - Infrastructure \\
VZ & -0.02 & 0.02 & 0.50 & 5.75 & Telecom Services \\
\bottomrule
\end{tabular}
```



[83]:

```
print(descriptive_stats.head())
```

```
  Ticker      Mean  Variance  Skewness  Kurtosis                  Industry
2     BA -0.003472  0.002904 -0.324005  6.213240        Aerospace & Defense
22   RTX -0.002113  0.001225 -0.174238  5.097088        Aerospace & Defense
13   JPM -0.001056  0.001069 -0.205374  6.741553        Banks - Diversified
14    KO -0.000457  0.000458 -0.773735  4.139190  Beverages - Non-Alcoholic
8     GS -0.000557  0.001040 -0.154397  6.422341           Capital Markets
```

[8]:

```
#Define target portfolio return as average daily return of DJ in October 2019

import yfinance as yf
import pandas as pd
import os
import numpy as np
```

```
ticker = "^DJI"
dji_data = yf.download(ticker, start='2019-10-01', end='2019-10-31',␣
  ↪interval='1d')['Close']
dji_log_returns = np.log(dji_data / dji_data.shift(1)).dropna()
p0 = np.mean(dji_log_returns)
print(p0)
```

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

```
0.0010871601183749330
```

Markowtiz optimisation set up

[42]:

```
import numpy as np


fc = 21 #forecasting period: 21 - monthly
rb = 10 #rebalancing: 10 - biweekly
n = log_returns.shape[1] #number of securities
tdays = log_returns.shape[0]
tperiods = int(( tdays - fc) / rb) - 1 #number of forecasting periods
eqw = np.full(n, 1 / n)  # Equally weighted portfolio


results_minvar = {
    'volatility_p': [],
    'return_p': [],
    'sharpe_ratio_p': [],
    'volatility_eqw': [],
    'return_eqw': [],
    'sharpe_ratio_eqw': [],
    'volatility_diff': [],
    'return_diff': [],
    'sharpe_ratio_diff': []
}


results_maxsr = {
    'volatility_p': [],
    'return_p': [],
    'sharpe_ratio_p': [],
    'volatility_eqw': [],
    'return_eqw': [],
    'sharpe_ratio_eqw': [],
    'volatility_diff': [],
    'return_diff': [],
    'sharpe_ratio_diff': []
}

riskfree_rate = rf_rate.values
```

Minimum variance optimisation

[43]:
```
import cvxopt
import numpy as np

cvxopt.solvers.options['show_progress'] = False

for i in range(tperiods):
    # Extract the rolling window
    window = log_returns.iloc[i * rb: fc + i * rb, :]
    window_cov = window.cov()
    log_returns_target = np.mean(window, axis=0)


    # Initialize quadratic programming problem to minimise variance
    P = 2 * cvxopt.matrix(window_cov.values) # Covariance matrix (for variance␣
 ↪minimization)
    q = cvxopt.matrix(np.zeros(n)) # No linear term in minimisation problem

    # Constraints: no short selling (weights >= 0) and minimum portfolio return␣
 ↪constraint
    G = cvxopt.matrix(np.vstack([-np.eye(n), log_returns_target]))
    h = cvxopt.matrix(np.append(np.zeros(n), -p0))

    # Fully invested portfolio: sum of weights = 1
    A = cvxopt.matrix(np.ones([1, n]))
    b = cvxopt.matrix([1.0])

    #Run optimisation problem
    sol = cvxopt.solvers.qp(P,q, G, h, A, b)

    #Check for optimisation failures
    if sol['status'] != 'optimal':
        print(f"Optimisation failed at iteration {i}")
        continue

    #Extract optimised weights
    weights = np.array(sol['x']).flatten()

    #Back-testing

    return_bt = log_returns.iloc[fc + (i + 1) * rb, :].values #Realised return
    variance_bt = log_returns.iloc[i * rb: fc + (i + 1) * rb, :].cov() #Realised covariance

    # Optimal portfolio variance, return and sharpe ratio calculatoin
    variance_p = weights.T @ variance_bt @ weights #Potrfolio variance (daily data)
    volatility_p = np.sqrt(variance_p) #Portfolio volatility (daily data)
    return_p = weights.T @ return_bt #Portfolio return daily
    sr_p = (return_p - riskfree_rate[i])/volatility_p #Portfolio Sharpe ratio

    #Calculate performance of benchmark - equally weighted portfolio
    volatility_bench = np.sqrt(eqw.T @ variance_bt @ eqw) #Benchmark variance (daily data)
    return_bench = eqw.T@return_bt #Benchmark daily returns
    sr_bench = (return_bench - riskfree_rate[i])/volatility_bench #Benchmark Sharpe ratio

    #Calculate differences in performance measures for optimised portfolio against benchmark
    volatility_diff = volatility_p-volatility_bench #Difference in volatility
    return_diff = return_p - return_bench #Difference in daily returns
    sr_diff = sr_p - sr_bench #Difference in Sharpe ratios

    # Store results
    results_minvar['volatility_p'].append(volatility_p)
    results_minvar['return_p'].append(return_p)
    results_minvar['sharpe_ratio_p'].append(sr_p)
    results_minvar['volatility_eqw'].append(volatility_bench)
    results_minvar['return_eqw'].append(return_bench)
    results_minvar['sharpe_ratio_eqw'].append(sr_bench)
    results_minvar['volatility_diff'].append(volatility_diff)
    results_minvar['return_diff'].append(return_diff)
    results_minvar['sharpe_ratio_diff'].append(sr_diff)
```

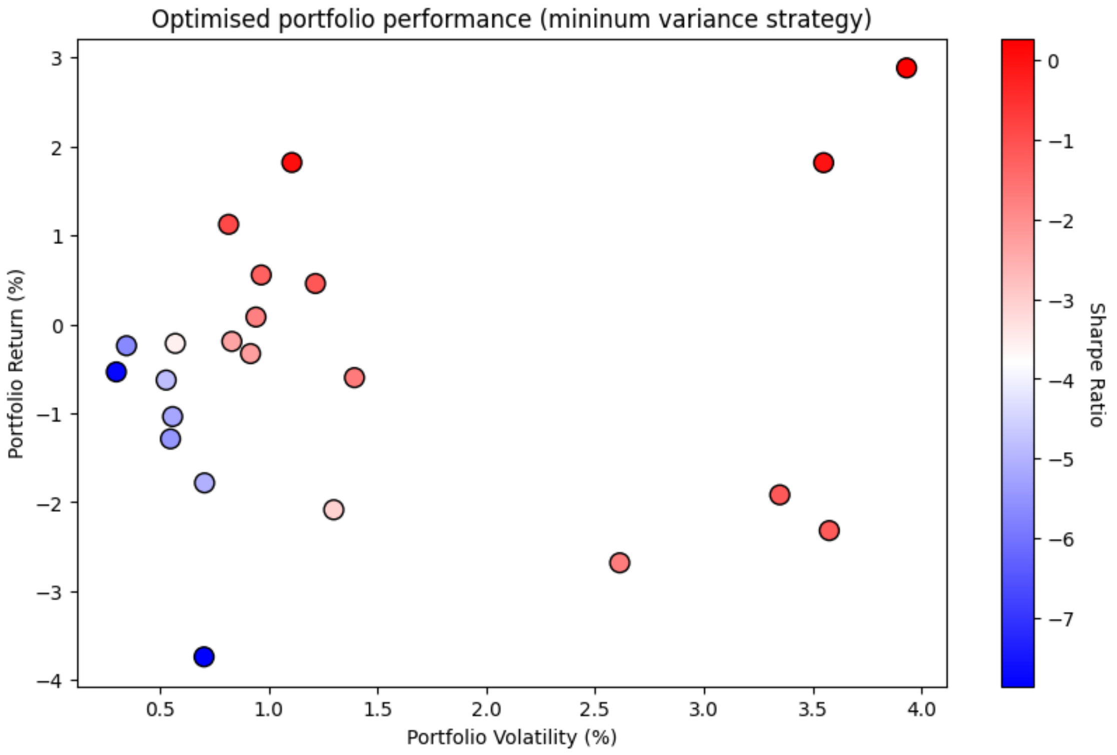
Maximum Sharpe Ratio

[46]:
```
import cvxopt
import numpy as np

cvxopt.solvers.options['show_progress'] = False

for i in range(tperiods):
    # Extract the rolling window
    window = log_returns.iloc[i * rb: fc + i * rb, :]
    window_cov = window.cov()
    log_returns_target = np.mean(window, axis=0)
```

```python
    # Calculate the excess returns (numerator in Sharpe ratio formula)
    excess_returns = log_returns_target - riskfree_rate[i]

    # Initialize quadratic programming problem to maximize Sharpe ratio
    P = cvxopt.matrix(window_cov.values)  # Covariance matrix (for variance minimization)
    q = cvxopt.matrix(-excess_returns)    # Negative of the excess returns (we want to maximize returns)

    # Constraints: no short selling (weights >= 0)
    G = cvxopt.matrix(np.vstack([-np.eye(n)]))
    h = cvxopt.matrix(np.zeros(n))

    # Fully invested portfolio: sum of weights = 1
    A = cvxopt.matrix(np.ones([1, n]))
    b = cvxopt.matrix([1.0])

    # Run optimization problem
    sol = cvxopt.solvers.qp(P, q, G, h, A, b)

    if sol['status'] != 'optimal':
        print(f"Optimization failed at iteration {i}")
        continue

    weights = np.array(sol['x']).flatten()  # Optimal weights

    #Back-testing

    return_bt = log_returns.iloc[fc + (i + 1) * rb, :].values #Realised return
    variance_bt = log_returns.iloc[i * rb: fc + (i + 1) * rb, :].cov() #Realised covariance

    # Optimal portfolio variance, return and sharpe ratio calculatoin
    variance_p = weights.T @ variance_bt @ weights #Potrfolio variance (daily data)
    volatility_p = np.sqrt(variance_p) #Portfolio volatility (daily data)
    return_p = weights.T @ return_bt #Portfolio return daily
    sr_p = (return_p - riskfree_rate[i])/volatility_p #Portfolio Sharpe ratio

    #Calculate performance of benchmark - equally weighted portfolio
    volatility_bench = np.sqrt(eqw.T @ variance_bt @ eqw) #Benchmark variance (daily data)
    return_bench = eqw.T@return_bt #Benchmark daily returns
    sr_bench = (return_bench - riskfree_rate[i])/volatility_bench #Benchmark Sharpe ratio
    #Calculate differences in performance measures for optimised portfolio␣
  ↪against benchmark
    volatility_diff = volatility_p-volatility_bench #Difference in volatility
    return_diff = return_p - return_bench #Difference in daily returns
    sr_diff = sr_p - sr_bench #Difference in Sharpe ratios

    # Store results
    results_maxsr['volatility_p'].append(volatility_p)
    results_maxsr['return_p'].append(return_p)
    results_maxsr['sharpe_ratio_p'].append(sr_p)
    results_maxsr['volatility_eqw'].append(volatility_bench)
    results_maxsr['return_eqw'].append(return_bench)
    results_maxsr['sharpe_ratio_eqw'].append(sr_bench)
    results_maxsr['volatility_diff'].append(volatility_diff)
    results_maxsr['return_diff'].append(return_diff)
    results_maxsr['sharpe_ratio_diff'].append(sr_diff)
```

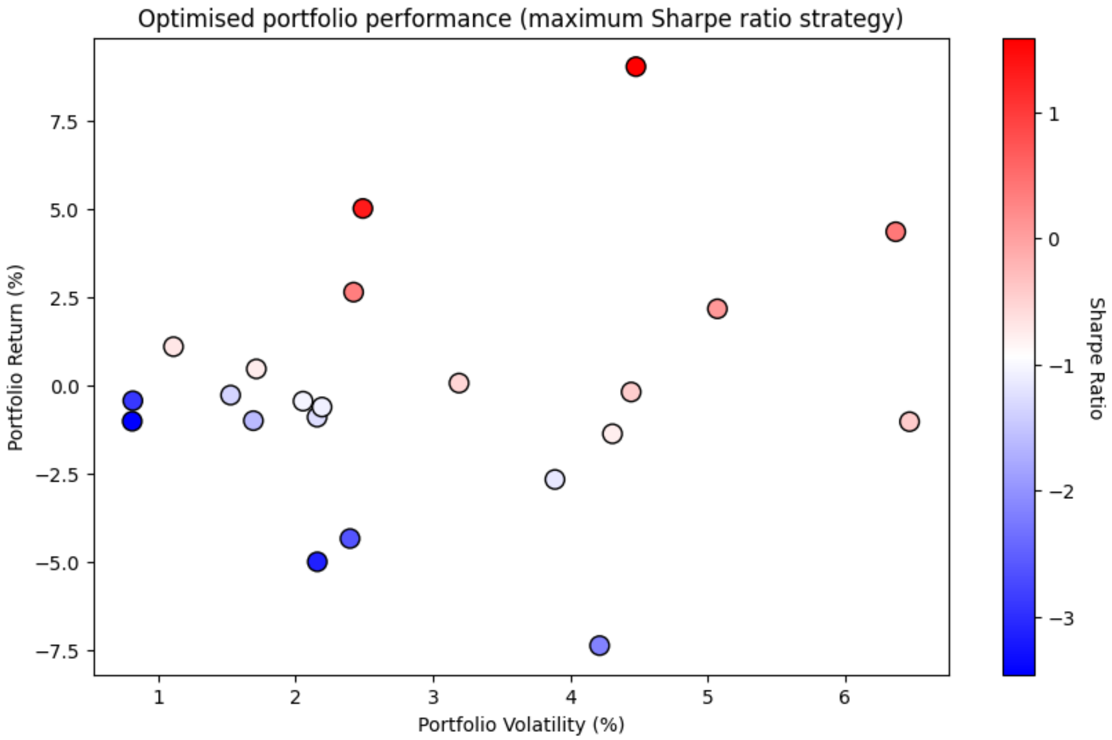
Plots

[47]:
```
# Scatter plot for Minimum Variance Strategy
plt.figure(figsize=(10, 6))

# Extracting data for the minimum variance strategy
returns_minvar = np.array(results_minvar['return_p'])*100  # Portfolio returns
volatility_minvar = np.array(results_minvar['volatility_p'])*100  # Portfolio␣
  ↪volatility
sharpe_ratios_minvar = np.array(results_minvar['sharpe_ratio_p'])  # Sharpe␣
  ↪ratios

# Scatter plot with Sharpe ratio as color
sc_minvar = plt.scatter(volatility_minvar, returns_minvar,␣
  ↪c=sharpe_ratios_minvar, cmap='bwr', s=100, edgecolor='k')
cbar_minvar = plt.colorbar(sc_minvar)
cbar_minvar.set_label('Sharpe Ratio', rotation=270, labelpad=15)

plt.xlabel('Portfolio Volatility (%)')
plt.ylabel('Portfolio Return (%)')
plt.title('Optimised portfolio performance (mininum variance strategy)')
plt.show()
```

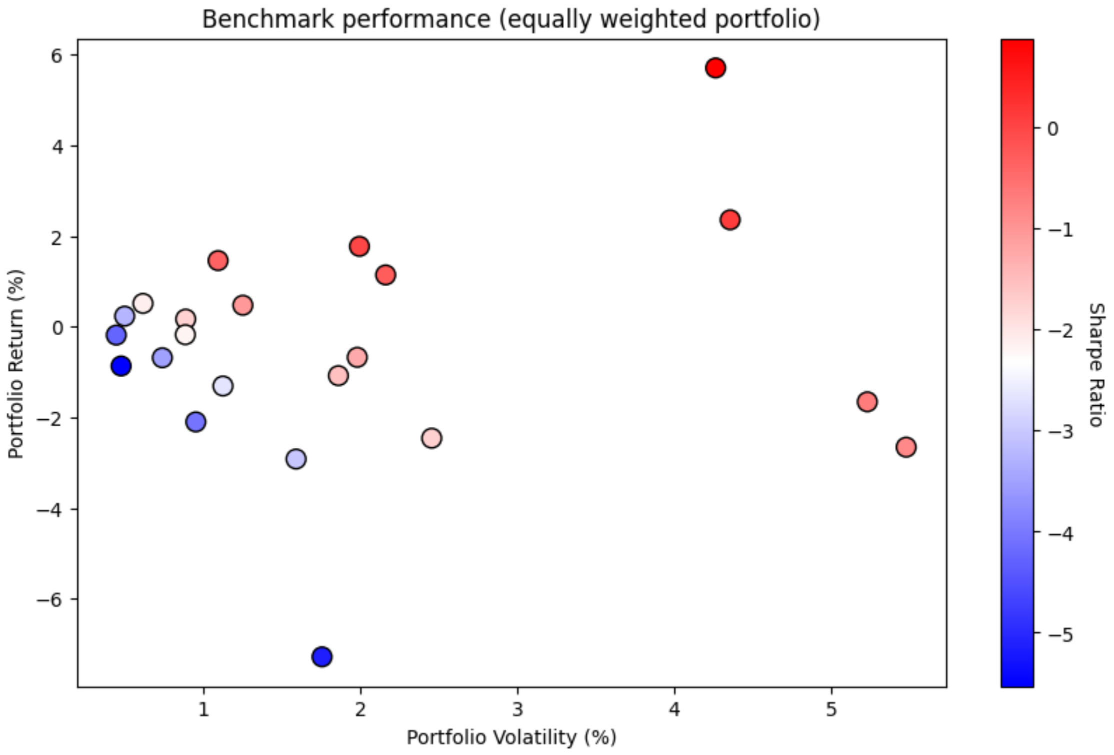


[48]:

```
# Scatter plot for Maximum Sharpe Ratio Strategy
plt.figure(figsize=(10, 6))

# Extracting data for the maximum Sharpe ratio strategy
returns_maxsr = np.array(results_maxsr['return_p'])*100  # Portfolio returns
volatility_maxsr = np.array(results_maxsr['volatility_p'])*100  # Portfolio␣
  ↪volatilities (already sqrt of variance)
sharpe_ratios_maxsr = np.array(results_maxsr['sharpe_ratio_p'])  # Sharpe ratios

# Scatter plot with Sharpe ratio as color
sc_maxsr = plt.scatter(volatility_maxsr, returns_maxsr, c=sharpe_ratios_maxsr,␣
  ↪cmap='bwr', s=100, edgecolor='k')
cbar_maxsr = plt.colorbar(sc_maxsr)
cbar_maxsr.set_label('Sharpe Ratio', rotation=270, labelpad=15)

plt.xlabel('Portfolio Volatility (%)')
plt.ylabel('Portfolio Return (%)')
plt.title('Optimised portfolio performance (maximum Sharpe ratio strategy)')
plt.show()
```



[49]:
```
# Scatter plot for benchmark portfolio
plt.figure(figsize=(10, 6))

# Extracting data for benchmark
returns_b = np.array(results_maxsr['return_eqw'])*100  # Portfolio returns
volatility_b = np.array(results_maxsr['volatility_eqw'])*100  # Portfolio␣
 ↪volatilities (already sqrt of variance)
sharpe_ratios_b = np.array(results_maxsr['sharpe_ratio_eqw'])  # Sharpe ratios

# Scatter plot with Sharpe ratio as color
sc_b = plt.scatter(volatility_b, returns_b, c=sharpe_ratios_b, cmap='bwr',␣
 ↪s=100, edgecolor='k')
cbar_b = plt.colorbar(sc_b)
cbar_b.set_label('Sharpe Ratio', rotation=270, labelpad=15)

plt.xlabel('Portfolio Volatility (%)')
plt.ylabel('Portfolio Return (%)')
plt.title('Benchmark performance (equally weighted portfolio)')
plt.show()
```

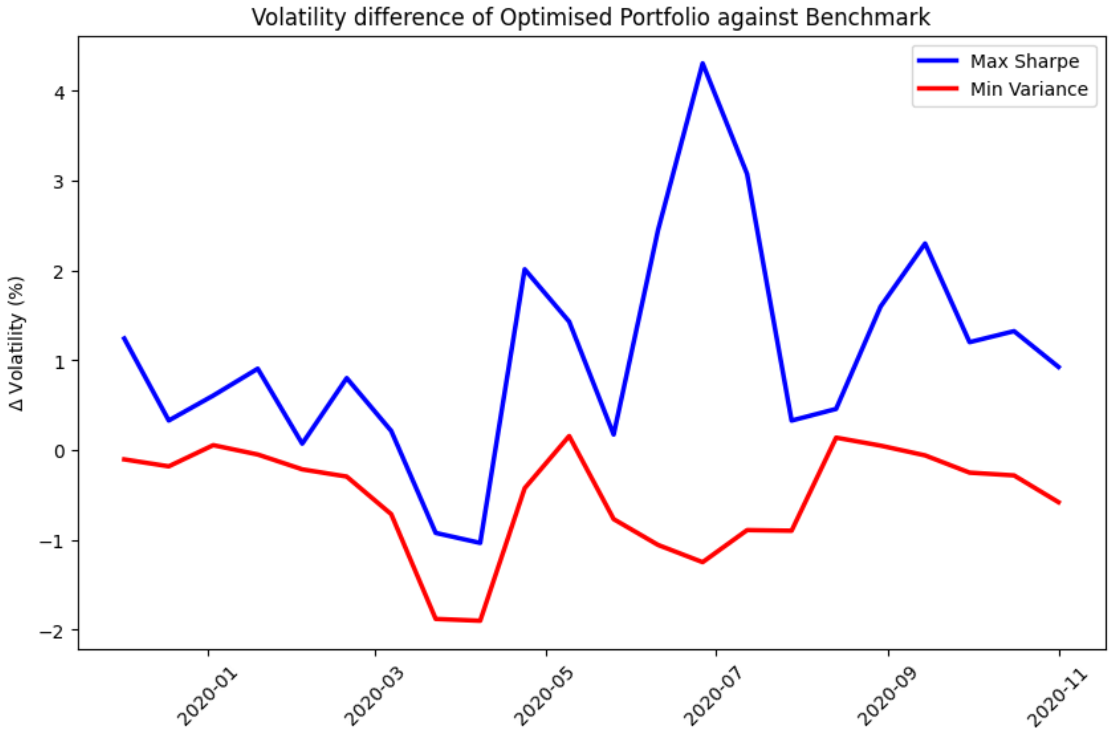


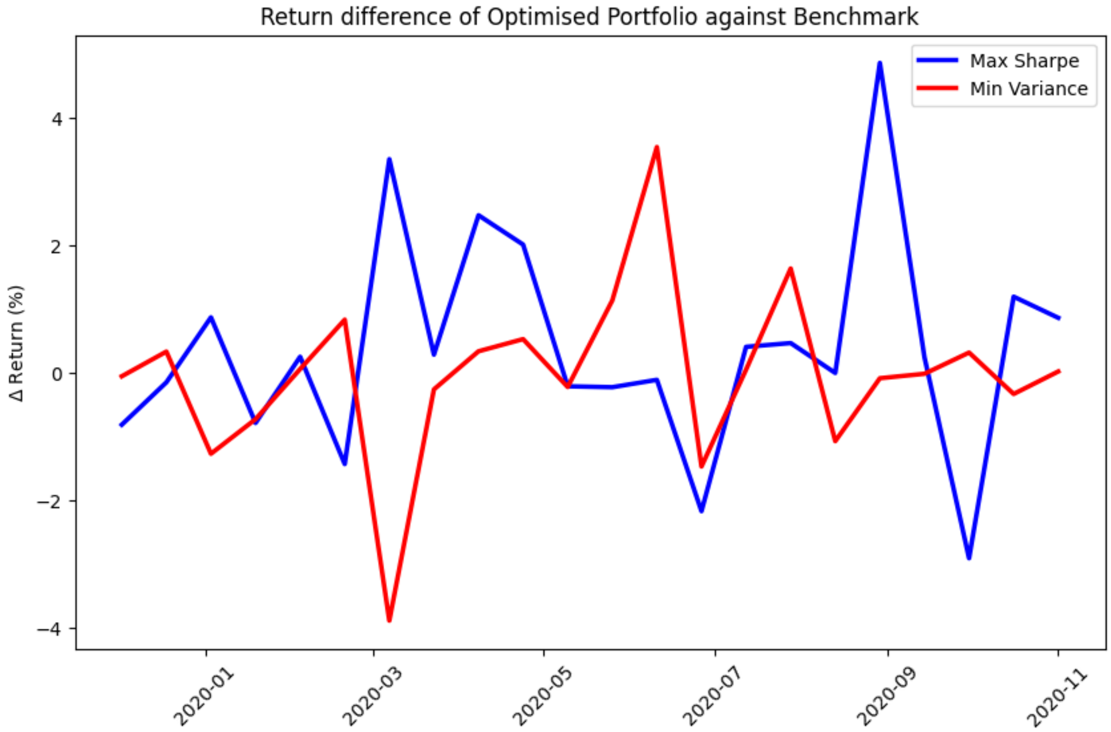
[50]:
```
#Generate dates

import pandas as pd
# The initial start date
start_date = pd.to_datetime("2019-11-01")
end_date = pd.to_datetime("2020-11-01")
start_f_date = start_date + pd.tseries.offsets.BDay(fc) #forecasting period␣
 ↪using as step
dates_df = pd.date_range(start=start_f_date, end=end_date, periods=tperiods)
len(dates_df)
```

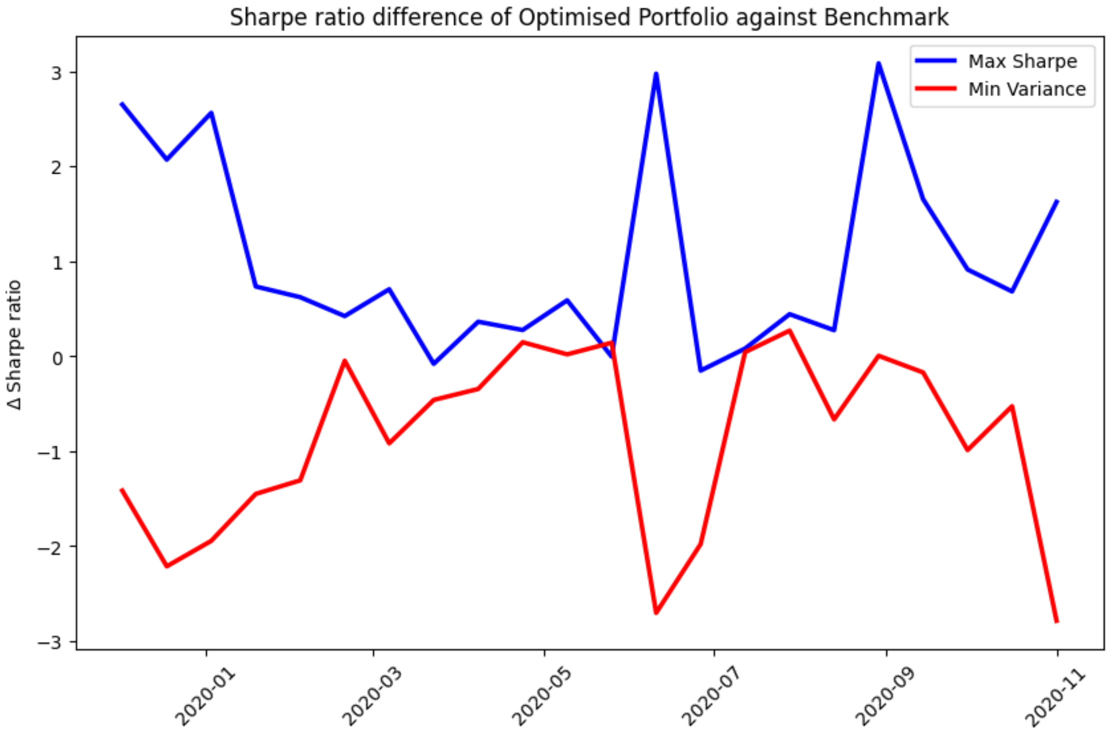
[50]:
```
22
```

[53]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# Plotting volatility for both portfolios
plt.figure(figsize=(10, 6))
plt.plot(dates_df, np.array(results_maxsr['volatility_diff'])*100, label="Max␣
 ↪Sharpe", color='blue', linewidth=2.5)
plt.plot(dates_df, np.array(results_minvar['volatility_diff'])*100, label="Min␣
 ↪Variance", color='red', linewidth=2.5)
```

```
plt.ylabel("$\Delta$ Volatility (%)")
plt.title("Volatility difference of Optimised Portfolio against Benchmark")
plt.xticks(rotation=45)
plt.legend()
plt.show()


# Plotting returns for both portfolios
plt.figure(figsize=(10, 6))
plt.plot(dates_df, np.array(results_maxsr['return_diff'])*100, label="Max␣
 ↪Sharpe", color='blue', linewidth=2.5)
plt.plot(dates_df, np.array(results_minvar['return_diff'])*100, label="Min␣
 ↪Variance", color='red', linewidth=2.5)
plt.ylabel("$\Delta$ Return (%)")
plt.title("Return difference of Optimised Portfolio against Benchmark")
plt.xticks(rotation=45)
plt.legend()
plt.show()

# Plotting Sharpe ratio difference for both portfolios
plt.figure(figsize=(10, 6))
plt.plot(dates_df, results_maxsr['sharpe_ratio_diff'], label="Max Sharpe",␣
 ↪color='blue', linewidth=2.5)
plt.plot(dates_df, results_minvar['sharpe_ratio_diff'], label="Min Variance",␣
 ↪color='red', linewidth=2.5)
plt.ylabel("$\Delta$ Sharpe ratio")
plt.title("Sharpe ratio difference of Optimised Portfolio against Benchmark")
plt.xticks(rotation=45)
plt.legend()
plt.show()
```


Volatility difference of Optimised Portfolio against Benchmark
Max Sharpe
Min Variance
Δ Volatility (%)
4
3
2
1
0
−1
−2
2020-01
2020-03
2020-05
2020-07
2020-09
2020-11


Return difference of Optimised Portfolio against Benchmark
Max Sharpe
Min Variance
Δ Return (%)
4
2
0
−2
−4
2020-01
2020-03
2020-05
2020-07
2020-09
2020-11


Sharpe ratio difference of Optimised Portfolio against Benchmark
Max Sharpe
Min Variance
Δ Sharpe ratio
3
2
1
0
−1
−2
−3
2020-01
2020-03
2020-05
2020-07
2020-09
2020-11