EF_Short_Sell

December 7, 2023

```python
[1]: import matplotlib.pyplot as plt
     import numpy as np
     import pandas as pd
     import scipy.optimize as sco
```

```python
[2]: import yfinance as yf
     import pandas_datareader as pdr
     import requests_cache
     session = requests_cache.CachedSession()
```

```python
[3]: %config InlineBackend.figure_format = 'retina'
     %precision 4
     pd.options.display.float_format = '{:.4f}'.format
```

```python
[4]: # Download real-time stock data from Yahoo! Finance

     rf = 0.0422
     tickers = 'AMZN NVDA TSLA UNH LLY'
     stocks = (
         yf.download(tickers=tickers, start='2018-12-01', end='2023-12-01',␣
      ↪progress=False)
         .resample('M')
         .last()
         .assign(Date=lambda x: x.index.tz_localize(None))
         .set_index('Date')
         .rename_axis(columns=['Variable', 'Ticker'])
     )
     monthly_prices = stocks['Adj Close']
     monthly_prices;
```

```python
[5]: # Calculate monthly log returns

     returns = np.log(monthly_prices / monthly_prices.shift(1)).
      ↪sort_values(by='Date', ascending=False).rename_axis(columns='Tickers').
      ↪dropna()
     returns.head()
```

```
[5]: Tickers       AMZN     LLY     NVDA     TSLA     UNH
     Date
     2023-11-30   0.0932   0.0667   0.1371   0.1785   0.0320
     2023-10-31   0.0459   0.0308  -0.0645  -0.2198   0.0604
     2023-09-30  -0.0822  -0.0313  -0.1262  -0.0309   0.0602
     2023-08-31   0.0319   0.2004   0.0547  -0.0356  -0.0606
     2023-07-31   0.0251  -0.0313   0.0995   0.0214   0.0521
```

```python
[6]: def port_vol(weights, returns):
         return np.sqrt(np.dot(weights.T, np.dot(returns.cov() * 252, weights)))
```

```python
[7]: def port_mean(weights, returns):
         return np.dot(weights, returns.mean()) * 252
```

```python
[8]: def neg_sharpe_ratio(weights, returns, rf):
         return -(port_mean(weights, returns) - rf) / port_vol(weights, returns)
```

```python
[9]: # Find the Global Minimum Variance Portfolio with minimize() function from
     ↪SciPy's optimize module

     res_mv = sco.minimize(
         fun=port_vol,
         x0=np.ones(len(returns.columns)) / len(returns.columns),
         args=(returns),
         method='SLSQP',
         bounds=[(-1,1) for _ in range(len(returns.columns))],
         constraints=({'type': 'eq', 'fun': lambda x: np.sum(x) - 1})
     )

     gmvp_weights = res_mv.x
     gmvp_ret = port_mean(gmvp_weights, returns)
     gmvp_vol = port_vol(gmvp_weights, returns)

     res_mv;
```

```python
[10]: # The Global Minimum Variance Portfolio Statistics

      def print_port_res(w, r, title, ppy=252, tgt=None, rf=rf):
          width = len(title)
          rp = r.dot(w)
          mu = ppy * rp.mean()
          sigma = np.sqrt(ppy) * rp.std()
          if tgt is not None:
              er = rp.sub(tgt)
              sr = np.sqrt(ppy) * er.mean() / er.std()
          elif rf is not None:
              sr = (mu - rf) / sigma
```

```python
        else:
            sr = None
    return print(
        title,
        '=' * width,
        '',
        'Performance',
        '-' * width,
        'Return:'.ljust(width - 6) + f'{mu:0.4f}',
        'Volatility:'.ljust(width - 6) + f'{sigma:0.4f}',
        'Sharpe Ratio:'.ljust(width - 6) + f'{sr:0.4f}\n' if sr is not None␣
 ↪else '',
        'Weights',
        '-' * width,
        '\n'.join([f'{_r}:'.ljust(width - 6) + f'{_w:0.4f}' for _r, _w in zip(r.␣
 ↪columns, w)]),
        sep='\n',
    )

print_port_res(w=res_mv['x'], r=returns, title='Global Minimum Variance␣
 ↪Portfolio', rf=rf)
```

```
Global Minimum Variance Portfolio
=================================

Performance
---------------------------------
Return:                 4.3925
Volatility:             0.8042
Sharpe Ratio:           5.4095

Weights
---------------------------------
AMZN:                   0.2199
LLY:                    0.2707
NVDA:                   0.0386
TSLA:                  -0.0611
UNH:                    0.5319
```

```python
[11]: # Find the Maximum Sharpe Ratio Portfolio with minimize() function from SciPy's␣
      ↪optimize module

res_max_sharpe = sco.minimize(
    fun=neg_sharpe_ratio,
    x0=np.ones(len(returns.columns)) / len(returns.columns),
    args=(returns, rf),
    method='SLSQP',
```

```python
        bounds=[(-1,1) for _ in range(len(returns.columns))],
        constraints=({'type': 'eq', 'fun': lambda x: np.sum(x) - 1})
)

max_sharpe_weights = res_max_sharpe.x
max_sharpe_ret = port_mean(max_sharpe_weights, returns)
max_sharpe_vol = port_vol(max_sharpe_weights, returns)

res_max_sharpe;
```

[12]:
```python
# The Maximum Sharpe Ratio Portfolio Statistics

def print_port_res(w, r, title, ppy=252, tgt=None, rf=rf):
    width = len(title)
    rp = r.dot(w)
    mu = ppy * rp.mean()
    sigma = np.sqrt(ppy) * rp.std()
    if tgt is not None:
        er = rp.sub(tgt)
        sr = np.sqrt(ppy) * er.mean() / er.std()
    elif rf is not None:
        sr = (mu - rf) / sigma
    else:
        sr = None
    return print(
        title,
        '=' * width,
        '',
        'Performance',
        '-' * width,
        'Return:'.ljust(width - 6) + f'{mu:0.4f}',
        'Volatility:'.ljust(width - 6) + f'{sigma:0.4f}',
        'Sharpe Ratio:'.ljust(width - 6) + f'{sr:0.4f}\n' if sr is not None␣
  ↪else '',
        'Weights',
        '-' * width,
        '\n'.join([f'{_r}:'.ljust(width - 6) + f'{_w:0.4f}' for _r, _w in zip(r.
  ↪columns, w)]),
        sep='\n',
    )

print_port_res(w=res_max_sharpe['x'], r=returns, title='Maximum Sharpe Ratio␣
  ↪Portfolio', rf=rf)
```

Maximum Sharpe Ratio Portfolio
==============================

```
Performance
-------------------------------
Return:              9.6393
Volatility:          1.1940
Sharpe Ratio:        8.0380

Weights
-------------------------------
AMZN:               -0.4308
LLY:                 0.5612
NVDA:                0.4226
TSLA:                0.0502
UNH:                 0.3968
```

[13]:
```python
# Define target return

tret = 252 * np.linspace(returns.mean().min(), returns.mean().max(), 100)
tret;
```

[14]:
```python
# Find the Efficient Frontier with minimize() function from SciPy's optimize
 ↪module

res_ef = []
for t in tret:
    _ = sco.minimize(
        fun=port_vol,
        x0=np.ones(len(returns.columns)) / len(returns.columns),
        args=(returns,),
        bounds=[(-1,1) for _ in range(len(returns.columns))],
        constraints=(
            {'type': 'eq', 'fun': lambda x: np.sum(x) - 1},
            {'type': 'eq', 'fun': lambda x: port_mean(x, returns) - t}
        )
    )
    res_ef.append(_)
```

[15]:
```python
# Create the Efficient Frontier Dataframe

ef = pd.DataFrame(
    {
        'tret': tret,
        'tvol': np.array([r['fun'] if r['success'] else np.nan for r in res_ef])
    }
)

ef.head();
```

```
[16]: # Plot the Efficient Frontier with target return

      ef.plot(x='tvol', y='tret', legend=False, figsize=(7, 6))
      plt.ylabel('Annualized Mean Return (%)')
      plt.xlabel('Annualized Volatility (%)')
      plt.title(
          f'Efficient Frontier' +
          f'\nfor {", ".join(returns.columns)}' +
          f'\nfrom {returns.index[-1]:%B %d, %Y} to {returns.index[0]:%B %d, %Y}'
      )

      for t, x, y in zip(
          returns.columns,
          returns.std().mul(np.sqrt(252)),
          returns.mean().mul(252)
      ):
          plt.annotate(text=t, xy=(x, y), xytext=(5, 5), textcoords='offset points')

      # Plot the GMVP
      plt.scatter(gmvp_vol, gmvp_ret, color='black', marker='o', s=50)
      plt.annotate('GMVP', xy=(gmvp_vol, gmvp_ret), xytext=(5, 5), textcoords='offset␣
       ↪points', color='black')

      # Plot the Maximum Sharpe Ratio Portfolio
      plt.scatter(max_sharpe_vol, max_sharpe_ret, color='black', marker='o', s=50)
      plt.annotate('Max Sharpe', xy=(max_sharpe_vol, max_sharpe_ret), xytext=(5, 5),␣
       ↪textcoords='offset points', color='black')
      plt.show()
```

Efficient Frontier
for AMZN, LLY, NVDA, TSLA, UNH
from January 31, 2019 to November 30, 2023