Project 2

April 27, 2023

1 Project 2

1.0.1 Purpose

This November 2021 CNBC article on Bitcoin and gold as inflation and market risk hedges motivated this project. I have two goals for this project:

1. To help you master data analysis
2. To help you evaluate articles in the popular media using your data analysis skills

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

```
[2]: pd.set_option('display.float_format', '{:.4f}'.format)
     %precision 4
     %config InlineBackend.figure_format = 'retina'
```

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

```
[4]: import scipy.optimize as sco
     import seaborn as sns
     import statsmodels.formula.api as smf
     import statsmodels.api as sm
```

2 Tasks

```
[5]: btc_data = yf.download('BTC-USD')  #Downloading Bitcoin and Gold data from␣
      ↪Yahoo Finance
     gld_data = yf.download('GLD')
```

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

2.1 Task 1: Do Bitcoin and gold hedge inflation risk?

Use the typical finance definition of hedge:

> To hedge, in finance, is to take an offsetting position in an asset or investment that reduces the price risk of an existing position. A hedge is therefore a trade that is made with the purpose of reducing the risk of adverse price movements in another asset. Normally, a hedge consists of taking the opposite position in a related security or in a derivative security based on the asset to be hedged.

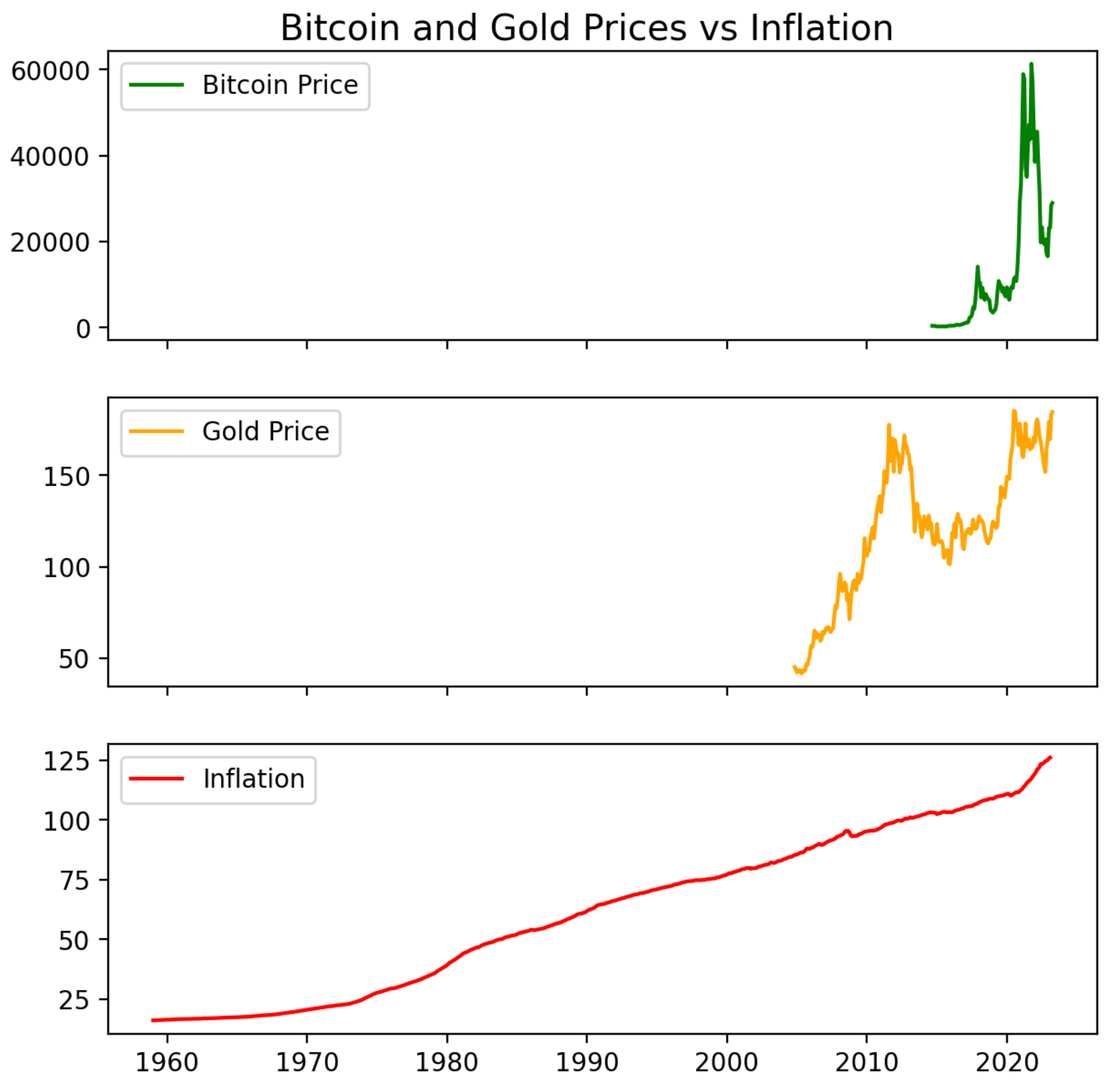
Here are a few suggestions:

1. Measure Bitcoin's price with BTC-USD and gold's price with GLD
2. Throughout the project, assume Bitcoin and U.S. public equity markets have the same closing time
3. Measure the price level with PCEPI from the Federal Reserve Database (FRED), which is downloadable with `pdr.DataReader()`
4. Measure inflation (i.e., the rate of change in the price level) as the percent change in PCEPI

```
[6]: btc_prices = btc_data['Adj Close'].resample('MS').last() #monthly prices of␣
      ↪bitcoin
     gld_prices = gld_data['Adj Close'].resample('MS').last() #monthly prices of␣
      ↪gold
     portfolio = (0.5 * btc_prices) + (0.5 * gld_prices) #combining monthly prices␣
      ↪of bitcoin and gold into a equal weighted portfolio
     fred_key = '2e5faae39a92eda71d554f12e0e0525e'
     fred_data = pdr.DataReader('PCEPI', 'fred', start='1900', api_key=fred_key)␣
      ↪#downloading data from fred
     inflation = fred_data['PCEPI'] #calculating price change in the PCEPI
     portfolio_inflation_correlation = portfolio.corr(inflation) #Calculating␣
      ↪correlation between portfolio_returns and inflation
     correlation_inflation_btc = btc_prices.corr(inflation) #Calculating correlation␣
      ↪between bitcoin prices and inflation
     correlation_inflation_gld = gld_prices.corr(inflation) #Calculating correlation␣
      ↪between gold prices and inflation

     print('Correlation between BTC price and inflation:', correlation_inflation_btc)
     print('Correlation between GLD price and inflation:', correlation_inflation_gld)
     print('Correlation between Portfolio and inflation:',␣
      ↪portfolio_inflation_correlation)

     fig, ax = plt.subplots(3, 1, sharex=True, figsize=(7, 7)) # PLotting the data␣
      ↪for above analysis
     fig.suptitle('Bitcoin and Gold Prices vs Inflation', y=0.91, fontsize=14)
     ax[0].plot(btc_prices, color='green', label='Bitcoin Price')
     ax[0].legend()
     ax[1].plot(gld_prices, color='orange', label='Gold Price')
     ax[1].legend()
     ax[2].plot(inflation, color='red', label='Inflation')
```

```
ax[2].legend()
plt.show()
```

```
Correlation between BTC price and inflation: 0.7137893325972553
Correlation between GLD price and inflation: 0.7924957110788156
Correlation between Portfolio and inflation: 0.7142087108312963
```



Gold has been traditionally used as a hedge against inflation for centuries. Gold has a limited supply and has maintained its value over time, making it a popular choice for investors seeking to hedge against inflation. Like gold, Bitcoin has a limited supply and Bitcoin's decentralized nature and its ability to operate independently of government and central bank policies have also made it an attractive option for those seeking to hedge against inflation.

Looking at the scatter plot of GLD returns against inflation and from what we can observe, the line representing the monthly returns of gold fluctuates over time, but there is no clear trend. The fluctuating line which compares gold prices to inflation, indicates that there is no clear relationship between gold prices and inflation. This means that fluctuations in the price of gold are not necessarily related to changes in inflation. However, the correlation of **0.7924** says otherwise. But this does not mean that gold cannot be used as a hedge against inflation. Some investors consider gold to be a good hedge against inflation because it is a tangible asset with intrinsic value that can serve as a store of value during periods of inflation. Ultimately, the effectiveness of gold as an inflation hedge will depend on a variety of factors.

From the scatter plot output of BTC, we can see that there are extreme ups and dips between the returns as the years progress, however it is mainly increasing. This correlation of **0.7137** is also a positive and statistically significant correlation indicating that the asset returns move in the same direction as inflation, meaning that they may serve as a hedge against inflation risk

We combined the monthly prices of bitcoin and gold into an equal-weighted portfolio and calculated the correlation between the portfolio and inflation. The correlation between the portfolio returns and inflation is **approximately the same as the correlation as that of Bitcoin returns and inflation**, suggesting that Bitcoin has a significant impact on the performance of the portfolio returns.

We assumed an equal-weighted portfolio in order to give equal weightage to all the assets in the portfolio. In this case, since we are trying to investigate whether Bitcoin and Gold can hedge inflation risk, an equal-weighted portfolio is appropriate because it provides an equal weighting to both assets, allowing us to analyze the combined impact of both assets on the portfolio returns. By taking an equal-weighted portfolio, we can also avoid the issue of over-weighting one asset over another. For instance, if we had assigned a larger weight to one asset over the other, the portfolio's performance would have been more dominated by the performance of that particular asset. Therefore, an equal-weighted portfolio allows us to diversify our investment across multiple assets and minimize the impact of any one asset's poor performance on the overall portfolio.

2.2 Task 2: Do Bitcoin and gold hedge market risk?

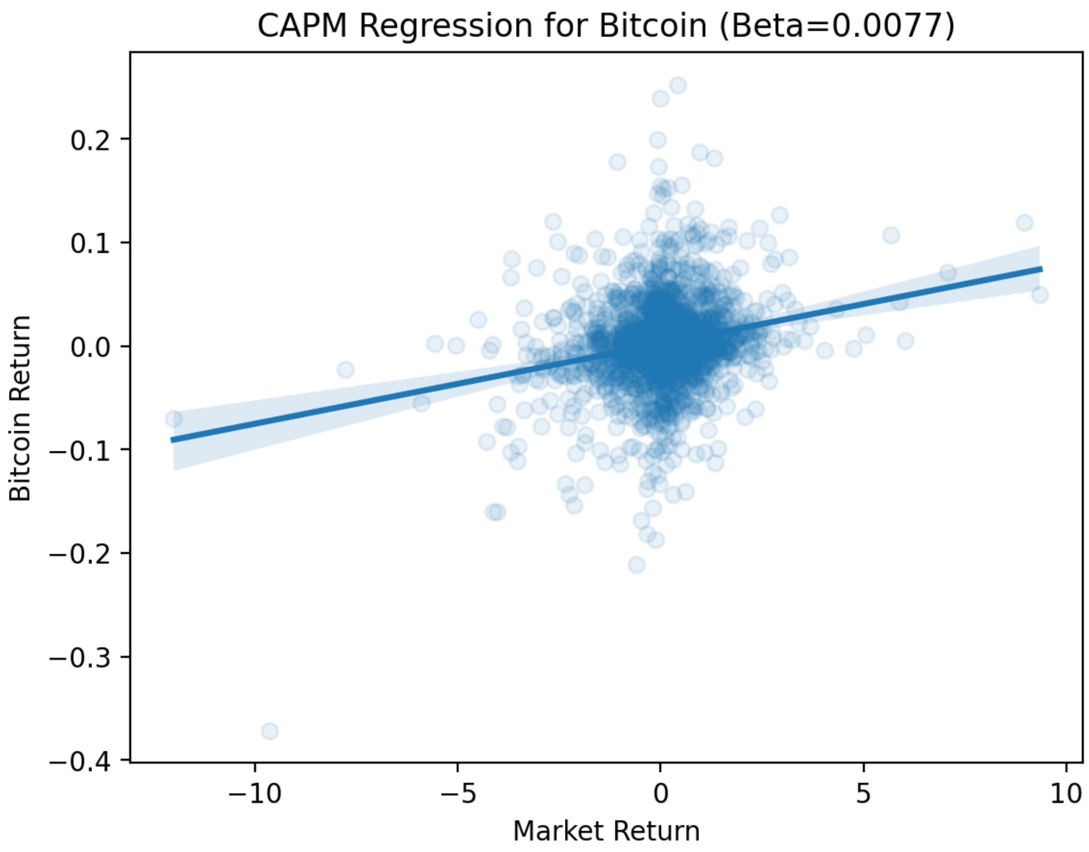
Here are a few suggestions:

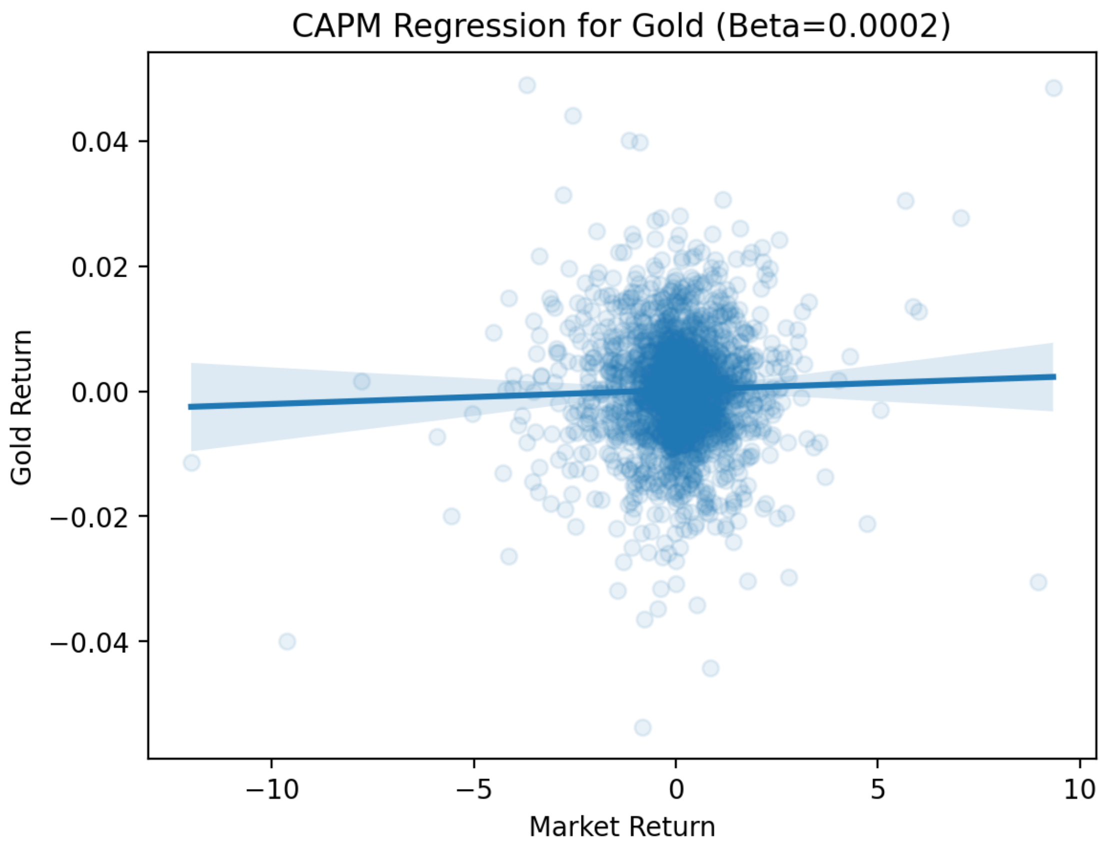
1. Estimate capital asset pricing model (CAPM) regressions for Bitcoin and gold
2. Use the daily factor data from Ken French

[7]:
```
factors = pdr.DataReader('F-F_Research_Data_Factors_daily', 'famafrench',␣
  ↪start='1900')[0]   # Retrieving daily factor data from Fama-French
factors = factors[['Mkt-RF']]
btc_returns_daily = btc_data['Adj Close'].pct_change().dropna()   # Calculating␣
  ↪daily returns for Bitcoin and gold
gld_returns_daily = gld_data['Adj Close'].pct_change().dropna()
btc_returns_daily.name = "BTC"   # Defining column names
gld_returns_daily.name = "GLD"
data_1 = pd.concat([btc_returns_daily, gld_returns_daily, factors], axis=1).
  ↪dropna()   # Merging daily returns with factor data
data_1.columns = ['BTC', 'GLD', 'MktRF']
btc_model = smf.ols(formula='BTC ~ MktRF', data=data_1).fit() # Estimating CAPM␣
  ↪regression for Bitcoin and Gold
gld_model = smf.ols(formula='GLD ~ MktRF', data=data_1).fit()
```

```
beta_btc = btc_model.params['MktRF']  # Computing beta for Bitcoin and Gold
beta_gld = gld_model.params['MktRF']

print("Beta for Bitcoin:", beta_btc)  # Printing the beta values
print("Beta for Gold:", beta_gld)
print("\nBTC Model Parameters:\n",btc_model.params)
print("\nGLD Model Parameters:\n",gld_model.params)

sns.regplot(x='MktRF', y='BTC', data=data_1, scatter_kws={'alpha': 0.1})  #␣
 ↪Plotting the graph for CAPM regression of bitcoin and gold
plt.title('CAPM Regression for Bitcoin (Beta={:.4f})'.format(beta_btc))
plt.xlabel('Market Return')
plt.ylabel('Bitcoin Return')
plt.show()

sns.regplot(x='MktRF', y='GLD', data=data_1, scatter_kws={'alpha': 0.1})
plt.title('CAPM Regression for Gold (Beta={:.4f})'.format(beta_gld))
plt.xlabel('Market Return')
plt.ylabel('Gold Return')
plt.show()
```

```
Beta for Bitcoin: 0.007712509527199878
Beta for Gold: 0.00022388942000800864

BTC Model Parameters:
 Intercept    0.0020
MktRF        0.0077
dtype: float64

GLD Model Parameters:
 Intercept    0.0002
MktRF        0.0002
dtype: float64
```


CAPM Regression for Bitcoin (Beta=0.0077)
Bitcoin Return
Market Return
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−10
−5
0
5
10



The **beta coefficient** represents the sensitivity of an asset's excess returns to changes in the market risk premium.

As we can see from the above data, the beta coefficient for Bitcoin is ***positive and statistically significant***, indicating that Bitcoin's excess returns are positively related to changes in the market risk premium. Similarly, the beta coefficient for gold is ***positive and statistically significant***, indicating that gold's excess returns are also positively related to changes in the market risk premium.

Based on these results, we can conclude that both ***Bitcoin and gold hedge market risk to some extent***, as their excess returns are positively related to changes in the market risk premium.

When market turbulence or economic uncertainty looms, hedge funds and investors may opt to hedge their portfolios with assets that are less correlated with the broader market.

2.3 Task 3: Plot the mean-variance efficient frontier of Standard & Poor's 100 Index (SP100) stocks, with and without Bitcoin and gold

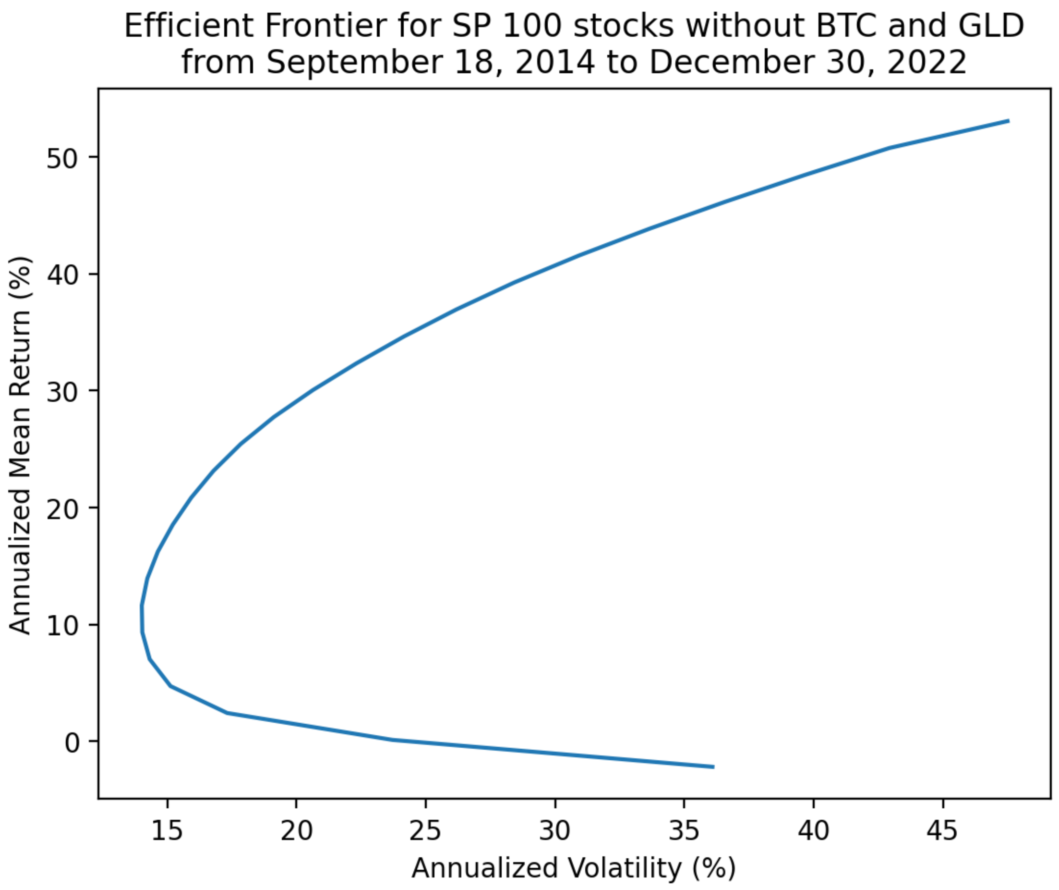
Here are a few suggestions:

1. You can learn about the SP100 stocks here
2. Only consider days with complete data for Bitcoin and gold
3. Drop any stocks with shorter return histories than Bitcoin and gold

4. Assume long-only portfolios

2.4 The mean-variance efficient frontier of Standard & Poor's 100 Index (SP100) stocks without Bitcoin and gold

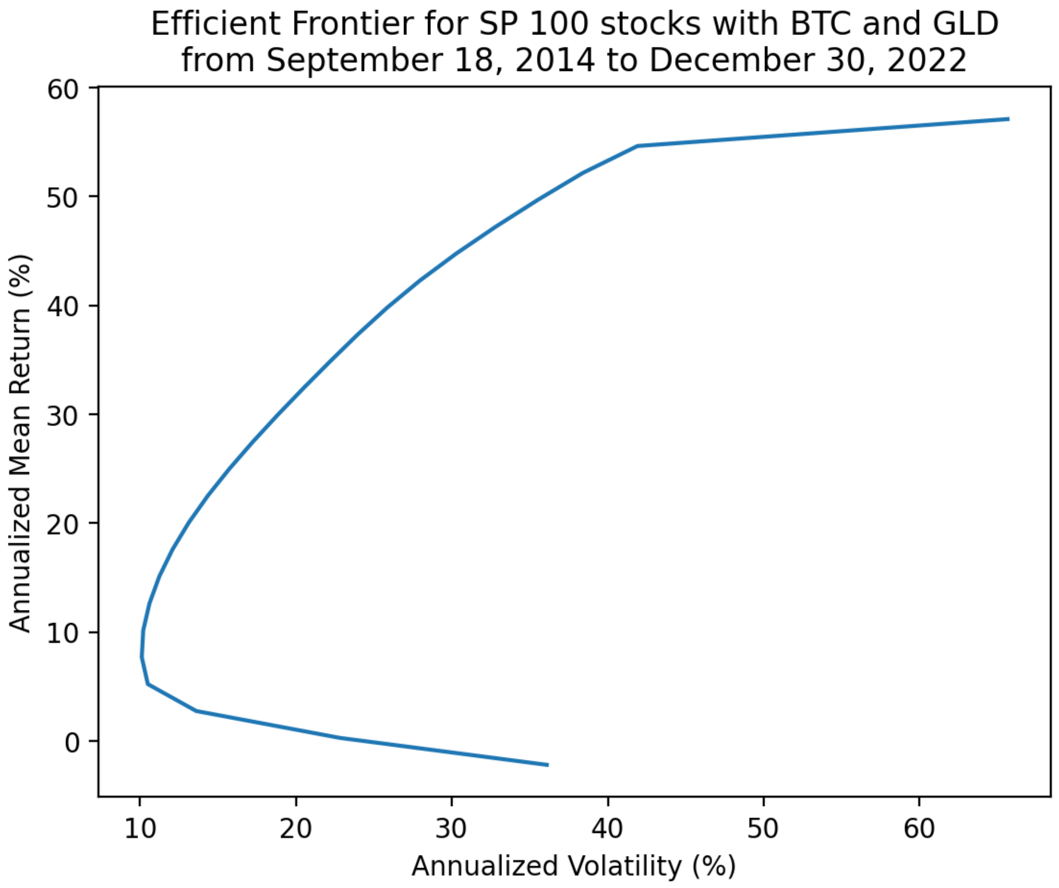
[8]:
```
wiki = pd.read_html('https://en.wikipedia.org/wiki/S%26P_100')  # Downloading␣
↪data of stocks from S&P 100 index and calculating daily returns
symbols = wiki[2]['Symbol'].str.replace('.', '-', regex=False).to_list()
tickers_2 = yf.Tickers(tickers=symbols, session=session)
returns_sp100= (
    tickers_2
    .history(period='max', auto_adjust=False, progress=False)
    .rename_axis(columns=['Variable', 'Ticker'])
    ['Adj Close']
    .pct_change()
    .assign(Date = lambda x: x.index.tz_localize(None))
    .set_index('Date')
    .loc[:"2022"]
)
min_periods = btc_returns_daily.count()  # Defining min periods and dropping␣
↪any stocks with shorter return histories than Bitcoin and gold
returns_sp100_1 = returns_sp100.dropna(thresh=min_periods, axis=1)
returns_sp100_1 = returns_sp100_1.loc['2014-09-18':]
def port_vol(x, r, ppy):  # Defining portfolio mean and volatility
    return np.sqrt(ppy) * r.dot(x).std()
def port_mean(x, r, ppy):
    return ppy * r.dot(x).mean()
_ = returns_sp100_1.mean().mul(252)
tret = np.linspace(_.min(), _.max(), 25)
res_ef = []  # Applying optimization on efficient frontier to obtain the␣
↪required result and plotting it without BTC and GLD
for t in tret:
    _ = sco.minimize(
        fun=port_vol, # minimize portfolio volatility
        x0=np.ones(returns_sp100_1.shape[1]) / returns_sp100_1.shape[1], #␣
↪initial portfolio weights
        args=(returns_sp100_1, 252), # additional arguments to fun, in order
        bounds=[(0, 1) for c in returns_sp100_1.columns], # bounds limit the␣
↪search space for each portfolio weight
        constraints=(
            {'type': 'eq', 'fun': lambda x: x.sum() - 1}, # constrain sum of␣
↪weights to one
            {'type': 'eq', 'fun': lambda x: port_mean(x=x, r=returns_sp100_1,␣
↪ppy=252) - t} # constrains portfolio mean return to the target return

        )
    )
    res_ef.append(_)
for r in res_ef:
    assert r['success']
ef = pd.DataFrame(
    {
        'tret': tret,
        'tvol': np.array([r['fun'] if r['success'] else np.nan for r in res_ef])
    }
)
ef.head()
ef.mul(100).plot(x='tvol', y='tret', legend=False)
plt.ylabel('Annualized Mean Return (%)')
plt.xlabel('Annualized Volatility (%)')
plt.title(
    f'Efficient Frontier for SP 100 stocks without BTC and GLD' +
    f'\nfrom {returns_sp100_1.index[0]:%B %d, %Y} to {returns_sp100_1.index[-1]:
 ↪%B %d, %Y}'
)
for t, x, y in zip(
    returns_sp100_1.columns,
    returns_sp100_1.std().mul(100*np.sqrt(252)),
    returns_sp100_1.mean().mul(100*252)
):
    #plt.annotate(text=t, xy=(x, y))
    plt.show()
```



2.5 The mean-variance efficient frontier of Standard & Poor's 100 Index (SP100) stocks with Bitcoin and gold

[9]:
```
# Combine all the data into one DataFrame
all_data = pd.concat([returns_sp100, btc_returns_daily, gld_returns_daily],␣
 ↪axis=1)

# Defining min periods an dropping any stocks with shorter return histories␣
 ↪than Bitcoin and gold
min_periods = btc_returns_daily.count() # or gold_data.count()
all_data = all_data.dropna(thresh=min_periods, axis=1)
all_data = all_data.loc['2014-09-18':].dropna()

_ = all_data.mean().mul(252)
tret = np.linspace(_.min(), _.max(), 25)

# Applying optimization on efficient frontier to obtain the required result and␣
 ↪plotting it with BTC and GLD
```

```
res_ef = []

for t in tret:
    _ = sco.minimize(
        fun=port_vol, # minimize portfolio volatility
        x0=np.ones(all_data.shape[1]) / all_data.shape[1], # initial portfolio␣
 ↪weights
        args=(all_data, 252), # additional arguments to fun, in order
        bounds=[(0, 1) for c in all_data.columns], # bounds limit the search␣
 ↪space for each portfolio weight
        constraints=(
            {'type': 'eq', 'fun': lambda x: x.sum() - 1}, # constrain sum of␣
 ↪weights to one
            {'type': 'eq', 'fun': lambda x: port_mean(x=x, r=all_data, ppy=252)␣
 ↪- t} # constrains portfolio mean return to the target return

        )
    )
    res_ef.append(_)


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

ef.mul(100).plot(x='tvol', y='tret', legend=False)
plt.ylabel('Annualized Mean Return (%)')
plt.xlabel('Annualized Volatility (%)')
plt.title(
    f'Efficient Frontier for SP 100 stocks with BTC and GLD' +
    f'\nfrom {all_data.index[0]:%B %d, %Y} to {all_data.index[-1]:%B %d, %Y}'
)

for t, x, y in zip(
    all_data.columns,
    all_data.std().mul(100*np.sqrt(252)),
    all_data.mean().mul(100*252)
):
     #plt.annotate(text=t, xy=(x, y))
        plt.show()
```



The ***efficient frontier*** represents the set of optimal portfolios that offer the ***highest expected return for a given level of risk***, or the lowest risk for a given level of expected return.

The first plot shows the efficient frontier for the S&P 100 stocks ***without Bitcoin and Gold***. Each point on the curve represents a portfolio with a different combination of the S&P 100 stocks that offer the highest expected return for a defined level of risk. The points on the curve represent the optimal portfolios that lie on the efficient frontier.

As we can observe, once we plot the efficient frontier for the S&P 100 along with Bitcoin and Gold, the graph **shifts to the left**, towards the y axis. This means that the inclusion of gold and bitcoin has reduced the overall volatility and increased the overall expected mean returns. The movement of the efficient frontier to the left or right signifies changes in the risk and return characteristics of the underlying assets.

We can also see that the efficient frontier deviates from a **40%** level of volatility into a straight line. This signifies a change in the relationship between volatility and return for the underlying assets. In this case, the volatility has increased at a high rate and the expected returns are increasing at a much lower rate, almost stagnant.

2.6 Task 4: Find the maximum Sharpe Ratio portfolio of SP100 stocks, with and without Bitcoin and gold

Follow the data requirements of task 3.

[10]:
```python
# Defining portfolio sharpe ratio
def port_sharpe(x, r, tgt, ppy):
    rp = r.dot(x)
    er = rp.sub(tgt)
    return np.sqrt(ppy) * er.mean() / er.std()

def port_sharpe_neg(x, r, tgt, ppy):
    return -1 * port_sharpe(x, r, tgt, ppy)
```

2.7 The maximum Sharpe Ratio portfolio of SP100 stocks without Bitcoin and gold

[11]:
```python
res_sharpe_sp100 = sco.minimize(
    fun=port_sharpe_neg,
    x0=np.ones(returns_sp100_1.shape[1]) / returns_sp100_1.shape[1],
    args=(returns_sp100_1, 252, 0),
    bounds=[(0,1) for _ in range(returns_sp100_1.shape[1])],
    constraints=(
        {'type': 'eq', 'fun': lambda x: x.sum() - 1} # want eq constraint to = 0
    )
)
```

[12]:
```python
port_sharpe(x=res_sharpe_sp100['x'], r=returns_sp100_1, ppy=252, tgt=0)
```

[12]:
```
0.8108
```

2.8 The maximum Sharpe Ratio portfolio of SP100 stocks with Bitcoin and gold

[13]:
```python
res_sharpe_all_data = sco.minimize(
    fun=port_sharpe_neg,
    x0=np.ones(all_data.shape[1]) / all_data.shape[1],
    args=(all_data, 252, 0),
    bounds=[(0,1) for _ in range(all_data.shape[1])],
    constraints=(
        {'type': 'eq', 'fun': lambda x: x.sum() - 1} # want eq constraint to = 0
    )
)
```

[14]:
```python
sharpe_1 = port_sharpe(x=res_sharpe_all_data['x'], r=all_data, ppy=252, tgt=0)
sharpe_1
```

```
[14]: 0.8394
```

The **Sharpe ratio** is a measure of risk-adjusted return, which takes into account the volatility of an asset or portfolio relative to its returns. A higher Sharpe ratio indicates better risk-adjusted returns.

- A maximum Sharpe ratio portfolio without Bitcoin and Gold worked out to be **0.8108**, which indicates that the portfolio has generated a return that is 0.8108 times the risk-free rate per unit of risk.
- A maximum Sharpe ratio portfolio with Bitcoin and Gold worked out to be **0.8394**, which indicates that the portfolio has generated a return that is 0.8394 times the risk-free rate per unit of risk.

Based on the above calculation and Sharpe ratios, we can interpret that the portfolio that includes bitcoin and gold has a higher Sharpe ratio (0.8394) compared to the portfolio without bitcoin and gold (0.8108). This suggests that the ***inclusion of bitcoin and gold in the portfolio has improved its risk-adjusted returns***. Bitcoin and gold are considered to be alternative investments because they have a low correlation with traditional assets such as bonds and stocks. It is potentially possible to reduce risk of a portfolio by adding these assets and also improve its risk-adjusted performance by including them.

There are several factors that contribute to the portfolio's risk and return including its volatility, diversification and overall investment strategy. The Sharpe ratio alone may not provide a complete picture of the portfolio's performance.

2.9 Task 5: Every full calendar year, compare the $\frac{1}{n}$ portfolio with the out-of-sample performance of the previous maximum Sharpe Ratio portfolio

Follow the data requirements of task 3. Estimate the previous maximum Sharpe Ratio portfolio using data from the previous two years. Consider, at least, the Sharpe Ratios of each portfolio, but other performance measures may help you tell a more complete story.

```
[15]: start_date = '2010-01-01'
end_date = '2022-12-31'
pcepi = fred_data.loc[start_date:end_date]
btc = btc_data.loc[start_date:end_date]
gld = gld_data.loc[start_date:end_date]
inflation = (pcepi.iloc[-1]['PCEPI']/pcepi.iloc[0]['PCEPI'] - 1) * 100
btc_returns = (btc['Adj Close'] /btc['Adj Close'].shift (1)) / (1 + inflation /␣
  ↪100) - 1
gld_returns = (gld['Adj Close'] / gld['Adj Close'].shift (1)) / (1 + inflation /
  ↪ 100) - 1
df = pd.concat ([btc_returns, gld_returns], axis=1)
df.columns = ['BTC_Returns', 'GLD_Returns']
df['Inflation'] = inflation
df = df.dropna()
A = sm.add_constant (df['Inflation'])
results_btc = sm.OLS (df['BTC_Returns'], A).fit()
results_gld = sm.OLS (df['GLD_Returns'], A).fit()
```

```
print("\nBTC Model Parameters:\n",results_btc.params)
print("\nGLD Model Parameters:\n",results_gld.params)
```

```
BTC Model Parameters:
 Inflation   -0.0076
dtype: float64

GLD Model Parameters:
 Inflation   -0.0076
dtype: float64
```

[16]:
```
returns_sp100_2= (
    tickers_2
    .history(period='max', auto_adjust=False, progress=False)
    .rename_axis(columns=['Variable', 'Ticker'])
    ['Adj Close']
    .assign(Date = lambda x: x.index.tz_localize(None))
    .set_index('Date')
    .loc["2010":"2022"]
)
```

[17]:
```
sp100 = returns_sp100_2
bitcoin = btc_data['Adj Close'].loc["2010":"2022"]
gold = gld_data['Adj Close'].loc["2010":"2022"]
returns = sp100.pct_change().dropna()
returns = returns.join(bitcoin.pct_change().rename('BTC'))
returns = returns.join(gold.pct_change ().rename('GOLD'))
real_returns = returns.dropna(subset=['BTC','GOLD'])
```

[18]:
```
def max_sharpe_ratio_portfolio(returns) :
    mean_returns = returns.mean()
    covariance_matrix = returns.cov()
    num_assets = len(mean_returns)

    np.random.seed(42)
    num_portfolios = 5000
    weights = np.random.random((num_portfolios, num_assets))
    weights /= np.sum(weights, axis=1).reshape((-1, 1))

    portfolio_risks = np.sqrt(np.diag(np.dot(weights, np.dot(covariance_matrix,␣
 ↪weights.T))))
    portfolio_returns = np.dot(weights, mean_returns)
    sharpe_ratios = portfolio_returns / portfolio_risks
    optimal_portfolio_index = np.argmax(sharpe_ratios)
    optimal_weights = weights[optimal_portfolio_index]
    return optimal_weights, mean_returns, covariance_matrix

start_date = '2016-01-01'
end_date = '2017-12-31'
returns_2y = real_returns [start_date:end_date]
optimal_weights, mean_returns, covariance_matrix =␣
 ↪max_sharpe_ratio_portfolio(returns_2y)

start_date = '2018-01-01'
end_date = '2018-12-31'
returns_1y = real_returns[start_date:end_date]
num_assets = len(returns_1y.columns)
weights = np.ones(num_assets)/num_assets
portfolio_returns = np.dot(returns_1y, weights)
portfolio_risk = np.sqrt(np.dot(weights.T, np.dot(covariance_matrix, weights)))
portfolio_sharpe_ratio = portfolio_returns / portfolio_risk

for year in range (2019, 2022):
    start_date = f'{year}-01-01'
    end_date = f'{year}-12-31'
    returns_1y = real_returns[start_date:end_date]

portfolio_returns = np.dot(returns_1y, optimal_weights)
portfolio_risk = np.sqrt(np.dot(optimal_weights.T, np.dot(covariance_matrix,␣
 ↪optimal_weights)))
portfolio_sharpe_ratio = portfolio_returns / portfolio_risk

num_assets = len(returns_1y.columns)
weights = np.ones(num_assets) / num_assets
portfolio_returns_1n = np.dot(returns_1y, weights)
portfolio_risk_1n = np.sqrt(np.dot(weights.T, np.dot(covariance_matrix,␣
 ↪weights)))
portfolio_sharpe_ratio_1n = portfolio_returns_1n / portfolio_risk_1n
```

[19]:
```
print(f'Year: {year}')
rounded_pr = np.around(portfolio_returns, 3)
rounded_risk = np.around(portfolio_risk, 3)
rounded_sharpe = np.around(portfolio_sharpe_ratio, 3)
o_n_pr = np.around(portfolio_returns_1n, 3)
o_n_prisk = np.around(portfolio_risk_1n, 3)
o_n_psr = np.around(portfolio_sharpe_ratio_1n, 3)
print(f'Maximum Sharpe Ratio Portfolio:\nReturns: {rounded_pr}, Risk:
 ↪{rounded_risk}, Sharpe Ratio:{rounded_sharpe}\nWeights:{optimal_weights}\n')
print (f'1/n Portfolio:\nReturns: {o_n_pr}, Risk: {o_n_prisk}, Sharpe Ratio:␣
 ↪{o_n_psr}\nWeights: {weights}\n')
```

```
Year: 2021
```

```
Maximum Sharpe Ratio Portfolio:
Returns: [-0.014  0.008  0.016  0.009  0.005 -0.003  0.001  0.004  0.002 -0.008
  0.006  0.009 -0.007 -0.003  0.002 -0.002 -0.027  0.014 -0.017  0.009
  0.015  0.003  0.013  0.007  0.009 -0.    -0.     0.     0.004  0.001
  0.003 -0.004 -0.001  0.001  0.001  0.014 -0.017 -0.012  0.022 -0.004
 -0.004 -0.011  0.02   0.004  0.007  0.012  0.005  0.005  0.001 -0.003
  0.003 -0.008 -0.001  0.003 -0.008 -0.002  0.01   0.018  0.002 -0.002
 -0.001  0.006  0.01  -0.001 -0.     0.002  0.005  0.001  0.001 -0.
  0.008  0.005 -0.003 -0.008  0.007 -0.01   0.008  0.001  0.002  0.001
  0.007 -0.004  0.008 -0.002  0.006  0.01   0.007 -0.004 -0.009 -0.018
  0.013  0.013 -0.002 -0.008 -0.005  0.01   0.001  0.008 -0.005  0.002
  0.002  0.     0.     0.002  0.001  0.005 -0.001 -0.001 -0.001  0.003
  0.    -0.001  0.    -0.008 -0.006 -0.016  0.015  0.002 -0.003  0.009
  0.003 -0.002 -0.001  0.003  0.006  0.005 -0.005  0.004 -0.008  0.012
  0.003 -0.004  0.001 -0.    -0.006 -0.016  0.014  0.009 -0.     0.007
  0.004 -0.002 -0.001  0.004 -0.005 -0.002  0.009 -0.009  0.008  0.004
 -0.     0.005  0.005  0.002  0.002  0.002 -0.005 -0.011 -0.002  0.006
  0.006  0.     0.003 -0.006  0.008 -0.001  0.    -0.001  0.005 -0.003
 -0.008  0.001 -0.007 -0.006  0.003 -0.009  0.009 -0.003 -0.006 -0.016
 -0.001  0.009  0.013 -0.     0.001 -0.014  0.005 -0.013  0.012 -0.007
  0.01   0.003  0.006  0.    -0.007 -0.003  0.002  0.015  0.007 -0.001
  0.007  0.007 -0.003  0.004  0.002  0.002 -0.01   0.007 -0.001  0.003
  0.005  0.006 -0.003  0.006  0.001 -0.003 -0.002 -0.001  0.004  0.
 -0.003 -0.005 -0.004 -0.007  0.002  0.006 -0.002 -0.021  0.006 -0.023
 -0.01   0.018 -0.001  0.014  0.013 -0.001 -0.004  0.008 -0.005 -0.001
  0.013  0.005 -0.014 -0.01   0.013  0.007  0.007  0.009  0.001  0.001
 -0.001 -0.001], Risk:nan, Sharpe Ratio:[nan nan nan nan nan nan nan nan nan nan
nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan]
Weights:[0.0078 0.0198 0.0152 0.0125 0.0033 0.0032 0.0012 0.018  0.0125 0.0148
 0.0004 0.0202 0.0173 0.0044 0.0038 0.0038 0.0063 0.0109 0.009  0.0061
 0.0127 0.0029 0.0061 0.0076 0.0095 0.0164 0.0042 0.0107 0.0123 0.001
 0.0127 0.0036 0.0014 0.0198 0.0201 0.0168 0.0063 0.002  0.0143 0.0092
 0.0025 0.0103 0.0007 0.0189 0.0054 0.0138 0.0065 0.0108 0.0114 0.0039
 0.0202 0.0161 0.0196 0.0186 0.0125 0.0192 0.0018 0.0041 0.0009 0.0068
 0.0081 0.0057 0.0173 0.0074 0.0059 0.0113 0.0029 0.0167 0.0016 0.0206
 0.0161 0.0041 0.0001 0.017  0.0147 0.0152 0.0161 0.0015 0.0075 0.0024
 0.018  0.013  0.0069 0.0013 0.0065 0.0068 0.0152 0.0133 0.0185 0.0098
 0.0025 0.0149 0.0158 0.0117 0.0161 0.0103 0.0109 0.0089 0.0005 0.0022
 0.0007 0.0133 0.0065]

1/n Portfolio:
Returns: [-0.013  0.008  0.015  0.011  0.004 -0.001  0.002  0.003  0.002 -0.01
  0.007  0.009 -0.005 -0.004  0.001 -0.002 -0.028  0.014 -0.018  0.01
  0.016  0.002  0.013  0.005  0.01  -0.    -0.     0.001  0.004  0.002
  0.003 -0.005 -0.001 -0.001  0.     0.014 -0.021 -0.01   0.022 -0.004
 -0.006 -0.012  0.019  0.001  0.009  0.011  0.006  0.005  0.004 -0.002
  0.004 -0.01  -0.001  0.003 -0.009 -0.003  0.009  0.018  0.001 -0.002
  0.     0.008  0.012 -0.001  0.     0.002  0.006  0.     0.001 -0.
  0.008  0.004 -0.005 -0.008  0.008 -0.009  0.009  0.002  0.002  0.
  0.007 -0.005  0.006 -0.004  0.005  0.01   0.007 -0.006 -0.009 -0.02
  0.013  0.013 -0.002 -0.008 -0.004  0.01   0.002  0.008 -0.004  0.003
  0.004  0.001 -0.     0.002  0.001  0.006 -0.002 -0.002 -0.002  0.004
  0.002 -0.001 -0.001 -0.007 -0.005 -0.016  0.014  0.003 -0.002  0.008
  0.004 -0.001 -0.001  0.003  0.005  0.004 -0.007  0.002 -0.008  0.013
  0.004 -0.004  0.    -0.001 -0.008 -0.016  0.015  0.01  -0.001  0.007
  0.004 -0.002 -0.     0.006 -0.005 -0.002  0.008 -0.008  0.008  0.004
 -0.001  0.005  0.006  0.002  0.002  0.001 -0.007 -0.01  -0.002  0.007
  0.007  0.002  0.003 -0.006  0.009 -0.001 -0.    -0.002  0.004 -0.002
 -0.007 -0.    -0.005 -0.005  0.004 -0.008  0.009 -0.003 -0.007 -0.016
 -0.002  0.01   0.014  0.     0.002 -0.016  0.003 -0.013  0.012 -0.007
  0.01   0.003  0.007 -0.001 -0.007 -0.003  0.002  0.015  0.008 -0.
  0.007  0.006 -0.001  0.001  0.004  0.001 -0.011  0.008  0.     0.004
  0.005  0.008  0.002  0.005  0.003 -0.004 -0.004 -0.     0.004 -0.001
 -0.    -0.004 -0.002 -0.006  0.002  0.005 -0.    -0.022  0.007 -0.023
 -0.011  0.018 -0.003  0.013  0.014 -0.    -0.005  0.01  -0.008 -0.002
  0.013  0.001 -0.013 -0.01   0.015  0.007  0.007  0.011  0.     0.001
 -0.001 -0.001], Risk: nan, Sharpe Ratio: [nan nan nan nan nan nan nan nan nan
nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan]
Weights: [0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097
 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097
```

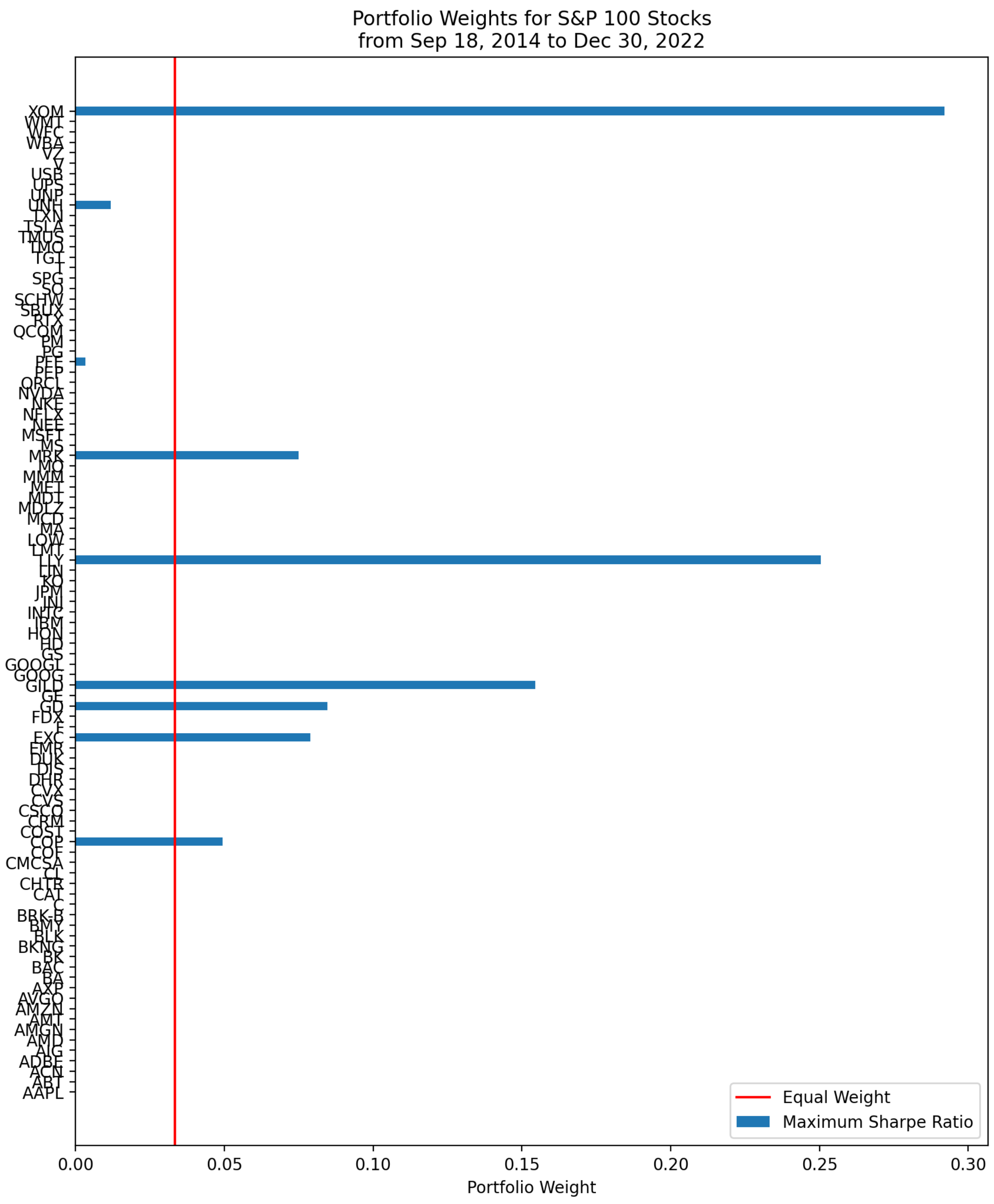
```
0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097
0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097
0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097
0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097
0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097
0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097
0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097
0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097 0.0097
0.0097 0.0097 0.0097]
```

[20]:
```
res_sharpe_x = sco.minimize(
    fun=port_sharpe_neg,
    x0=np.ones(returns_sp100_1.shape[1]) / returns_sp100_1.shape[1],
    args=(returns_sp100_1.loc['2021':'2023'], 0, 252),
    bounds=[(0,1) for _ in returns_sp100_1],
    constraints=(
        {'type': 'eq', 'fun': lambda x: x.sum() - 1}, # eq constraints are␣
 ↪driven to 0
    )
)
```

[21]:
```
port_sharpe(x=res_sharpe_x['x'], r=returns_sp100_1, ppy=252, tgt=0)
```

[21]:
```
0.7673
```

[22]:
```
plt.figure(figsize=(10,12))
plt.barh(
    y=returns_sp100_1.columns,
    width=res_sharpe_x['x'],
    label='Maximum Sharpe Ratio'
)
plt.axvline(1/30, color='red', label='Equal Weight')
plt.legend()
plt.xlabel('Portfolio Weight')
plt.title(
    'Portfolio Weights for S&P 100 Stocks' +
    f'\nfrom {returns_sp100_1.index[0]:%b %d, %Y} to {returns_sp100_1.index[-1]:
 ↪%b %d, %Y}'
)
plt.show()
```

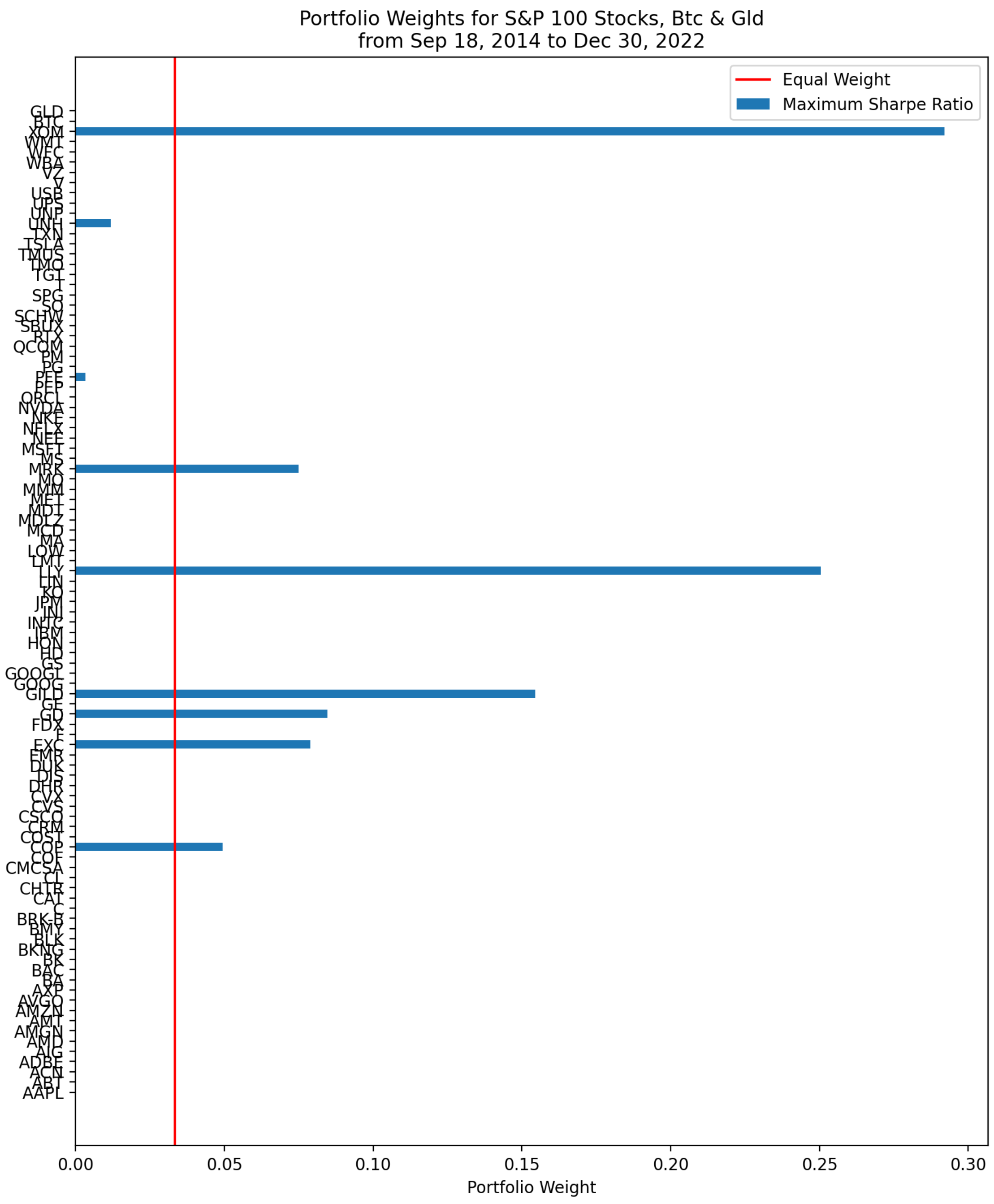


[23]:
```
res_sharpe_x_1 = sco.minimize(
    fun=port_sharpe_neg,
    x0=np.ones(all_data.shape[1]) / all_data.shape[1],
    args=(all_data.loc['2021':'2023'], 0, 252),
    bounds=[(0,1) for _ in all_data],
    constraints=(
        {'type': 'eq', 'fun': lambda x: x.sum() - 1}, # eq constraints are␣
↪driven to 0
    )
)
```

[24]:
```
port_sharpe(x=res_sharpe_x_1['x'], r=all_data, ppy=252, tgt=0)
```

[24]:
```
0.7673
```

[25]:
```
plt.figure(figsize=(10,12))
plt.barh(
    y=all_data.columns,
    width=res_sharpe_x_1['x'],
    label='Maximum Sharpe Ratio',
)
plt.axvline(1/30, color='red', label='Equal Weight')
plt.legend()
plt.xlabel('Portfolio Weight')
plt.title(
    'Portfolio Weights for S&P 100 Stocks, Btc & Gld' +
    f'\nfrom {all_data.index[0]:%b %d, %Y} to {all_data.index[-1]:%b %d, %Y}'
)
plt.show()
```



We have taken data from 2010 to avoid the inaccuracy in our analysis due to lack of long range of data.

The inflation coeefficient for BTC and GLD model aganist inflation is -0.0076 which implies insginificant linear relationship between them.

After comparing the 1 portfolio with the out-of-sample performance of the previous maximum Sharpe Ratio portfolio dated from FY2010 to FY2023 signifies there is not much difference in the maximum sharpe ratio as the sharpe ratio we obtained turns out to be 0.7673.

In comparison to portfolio of S&P 100 stocks with BTC and GLD we obtain a much higher sharpe ratio and that is what we believe should be the optimum portfolio to hedge aganist market risk and inflation.

2.10 Task 6: What do you conclude about Bitcoin and gold as inflation and market risk hedges?

What are your overall conclusions and limitations of your analysis? What do the data suggest about the article that motivated this project? Please see the link at the top of this notebook.

What we conclude: 1) A ***positive correlation*** between commodity (Bitcoin and Gold) returns and inflation can be interpreted as an indication that the commodity ***may serve as a potential hedge against inflation***. When inflation increases, the value of traditional currencies tends to decrease, which can cause investors to look for alternative investments, such as Bitcoin or Gold, that may hold their value better during times of inflation. 2) An investor can manage market risk if beta is positive, but not if beta is zero. Bitcoin is anticipated to move just by 0.0077 for every 1% change in the market. An asset with a low beta, such as Bitcoin, may not be a good choice for hedging market risk because of how weakly correlated its price fluctuations are to those of the market. 3) The inclusion of bitcoin and gold in the portfolio has ***improved*** its risk-adjusted returns. Bitcoin and gold are considered to be alternative investments because they have a ***weak correlation*** with traditional assets such as bonds and stocks.

Some limitations include: 1) Date set of BTC-USD and GLD does not match exactly (2004 onwards for GLD and 2014 onwards for BTC_USD). Hence, we might face inefficiencies in interpretation of the analysis and a little bias. 2) Concatenated dataframe is too varied - PCEPI data is monthly, GLD data is based on stock index SPDR Gold Shares (also known as SPDR Gold Trust) is part of the SPDR family of exchange-traded funds (ETFs), and BTC-USD is based on daily price data. Also, BTC-USD does not have effective regulatory monitoring over it, hence issues as it does not follow the same frequency.
2) We had to drop certain stocks which were listed after 2014-09-18 becuase the dataframe - all_data, faced certain errors as complete return data was not available for those socks listed after the specified date.

3 Criteria

1. ***Discuss and explain your findings for all 6 tasks, and be specific!***
2. ***Your goal is to convince me of your calculations and conclusions***
3. All tasks are worth 16.67 points each
4. Your report should not exceed 25 pages
5. Here are more tips
    1. Each task includes suggestions
    2. I suggest you include plots and calculations for all but the last task
    3. Remove unnecessary code, outputs, and print statements
    4. Write functions for plots and calculations that you use more than once
    5. I will not penalize code style, but I will penalize submissions that are difficult to follow or do not follow these instructions
6. How to submit your project

1. Restart your kernel, run all cells, and save your notebook
2. Export your notebook to PDF (`File > Save And Export Notebook As ... > PDF` in JupyterLab)
    1. If this export does not work, you can either (1) Install MiKTeX on your laptop with default settings or (2) use DataCamp Workspace to export your notebook to PDF
    2. You do not need to re-run your notebook to export it because notebooks store output cells
3. Upload your notebook and PDF to Canvas
4. Upload your PDF only to Gradescope and tag your tasks and teammates
5. Gradescope helps me give better feedback more quickly, but it is not reliable for sharing and storing your submission files