Project 2

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

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

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

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

Tasks

Task 1: Do Bitcoin and gold hedge inflation risk?

In [5]:
```
def get_prices(ticker, start_date):
    prices = yf.download(ticker, start=start_date, end='2023-04-01')
    return prices
```

In [6]:
```
def get_returns(prices, column_name='Returns'):
    returns = prices['Adj Close'].pct_change()
    returns.dropna(inplace=True)
    returns.rename(column_name, inplace=True)
    return returns
```

In [7]:
```
def get_correlation(returns1, returns2):
    correlation = returns1.corr(returns2)
    return correlation
```

In [8]:
```
def get_inflation(api_key, start_date):
    fred = pdr.DataReader("PCEPI", "fred", start=start_date, api_key=api_key)
    inflation = fred["PCEPI"]
    return inflation
```

In [9]:
```
btc_prices = get_prices('BTC-USD', '2014-01-01')['Adj Close']
gold_prices = get_prices('GLD', '2004-01-01')['Adj Close']
fred_api_key = "a1b3c14fa0e919eeb051241f8d255226"
inflation_price = get_inflation(fred_api_key, '2004-01-01')
```

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

In [10]:
```
btc_returns_monthly = get_prices('BTC-USD', '2014-01-01')['Adj Close'].pct_chang
gold_returns_monthly = get_prices('GLD', '2004-01-01')['Adj Close'].pct_change()
fred_api_key = "a1b3c14fa0e919eeb051241f8d255226"
PCEPI_price_for_gold = get_inflation(fred_api_key, '2004-01-01')
PCEPI_price_for_btc = get_inflation(fred_api_key, '2014-01-01')
inflation_for_gold = get_inflation(fred_api_key, '2004-01-01').pct_change()
inflation_for_btc = get_inflation(fred_api_key, '2014-01-01').pct_change()
```

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

In [11]:
```
btc_prices.index = pd.to_datetime(btc_prices.index)
btc_prices_monthly = btc_prices.resample('MS').last()
btc_returns_monthly = btc_returns_monthly.resample('MS').last()
gold_prices.index = pd.to_datetime(gold_prices.index)
gold_prices_monthly = gold_prices.resample('MS').last()
gold_returns_monthly = gold_returns_monthly.resample('MS').last()
```

In [12]:
```
btc_inflation_correlation = btc_prices_monthly.corr(PCEPI_price_for_btc)
gold_inflation_correlation = gold_prices_monthly.corr(PCEPI_price_for_gold)
print("Correlation between monthly BTC prices and inflation:"\
      , btc_inflation_correlation)
print("Correlation between monthly Gold prices and inflation:"\
      , gold_inflation_correlation)
```

```
Correlation between monthly BTC prices and inflation: 0.7137893325972554
Correlation between monthly Gold prices and inflation: 0.7924957110788152
```

The output suggests that there is a positive correlation between the monthly prices of Bitcoin and gold with inflation as measured by PCEPI (Personal Consumption Expenditures Price Index). The correlation coefficient between monthly BTC prices and inflation is 0.7138, while the correlation coefficient between monthly gold prices and inflation is 0.7925.

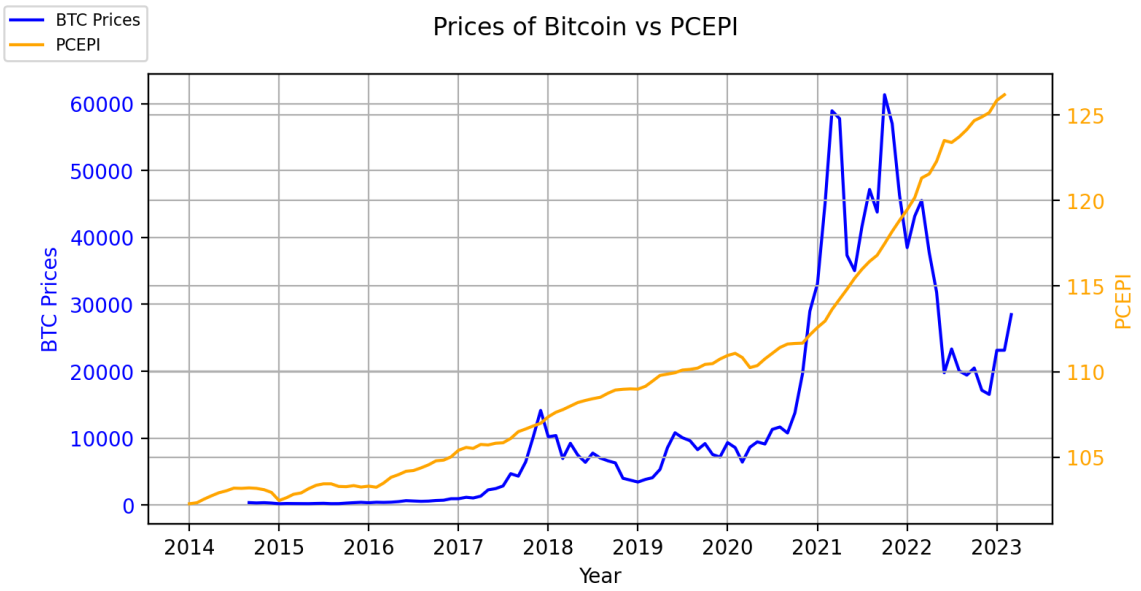
In [13]:
```
btc_inflation_returns_correlation = btc_returns_monthly.corr(inflation_for_btc)
gold_inflation_returns_correlation = gold_returns_monthly.corr(inflation_for_gol
print("Correlation between monthly BTC returns and inflation:"\
      , btc_inflation_returns_correlation)
print("Correlation between monthly Gold returns and inflation:"\
      , gold_inflation_returns_correlation)
```

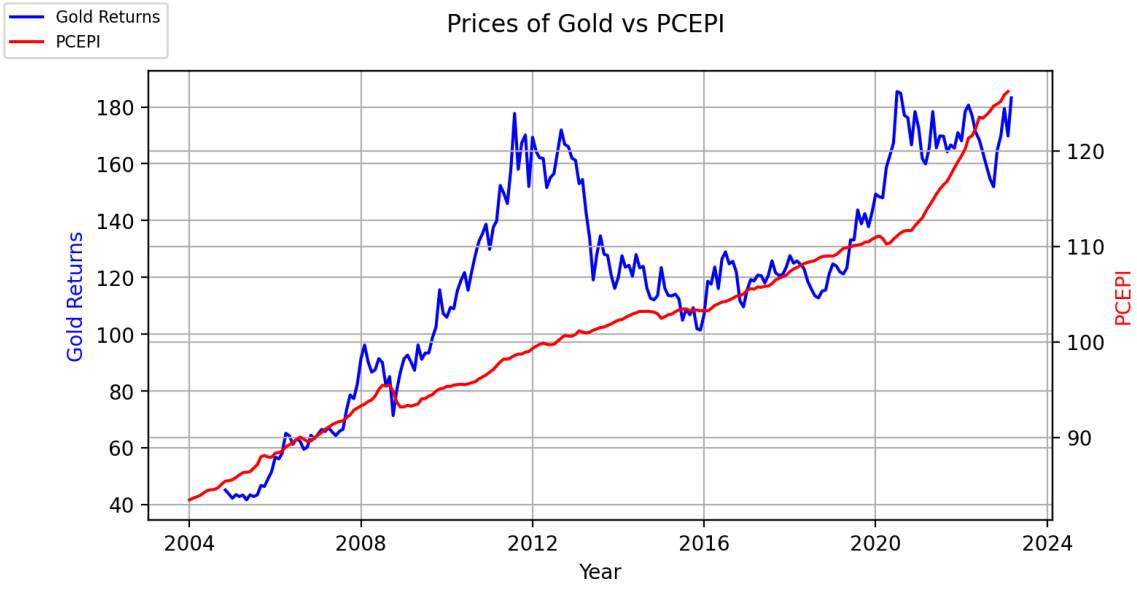
```
Correlation between monthly BTC returns and inflation: 0.08099145260545226
Correlation between monthly Gold returns and inflation: 0.1070134341734091
```

The output indicates a weak positive correlation between the monthly returns of Bitcoin and gold with inflation as measured by PCEPI. The correlation coefficient between monthly BTC returns and inflation is 0.0809, while the correlation coefficient between monthly gold returns and inflation is 0.1070.

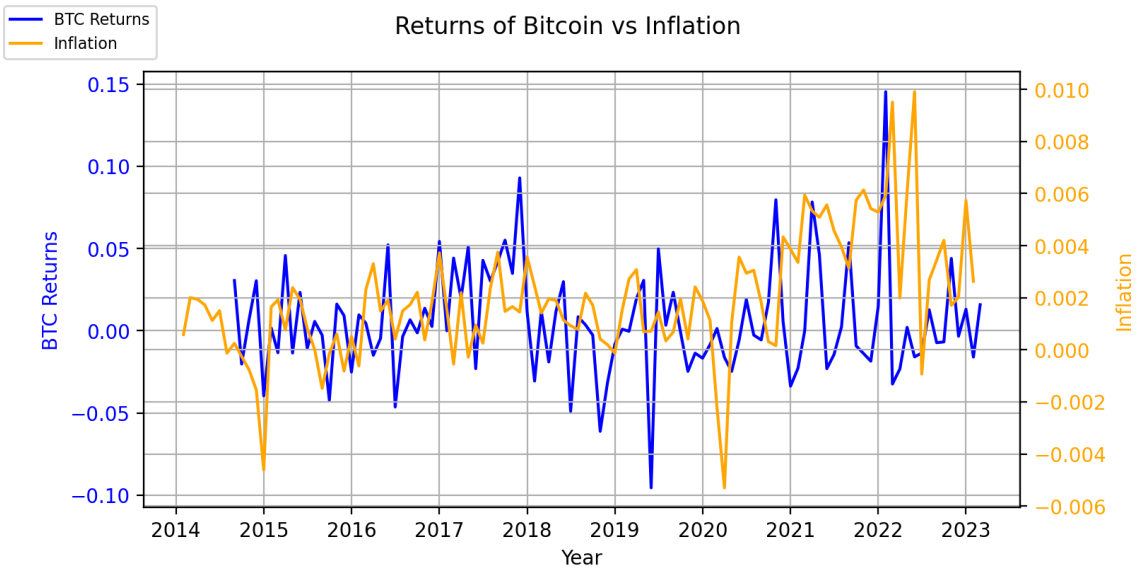
In [14]:
```
fig, ax1 = plt.subplots(figsize=(8, 4))
ax2 = ax1.twinx()
ax1.plot(btc_prices_monthly.index, btc_prices_monthly, \
         label='BTC Prices', color='blue')
ax1.set_xlabel('Year')
ax1.set_ylabel('BTC Prices', color='blue')
ax1.tick_params(axis='y', labelcolor='blue')
ax1.grid(True)
```

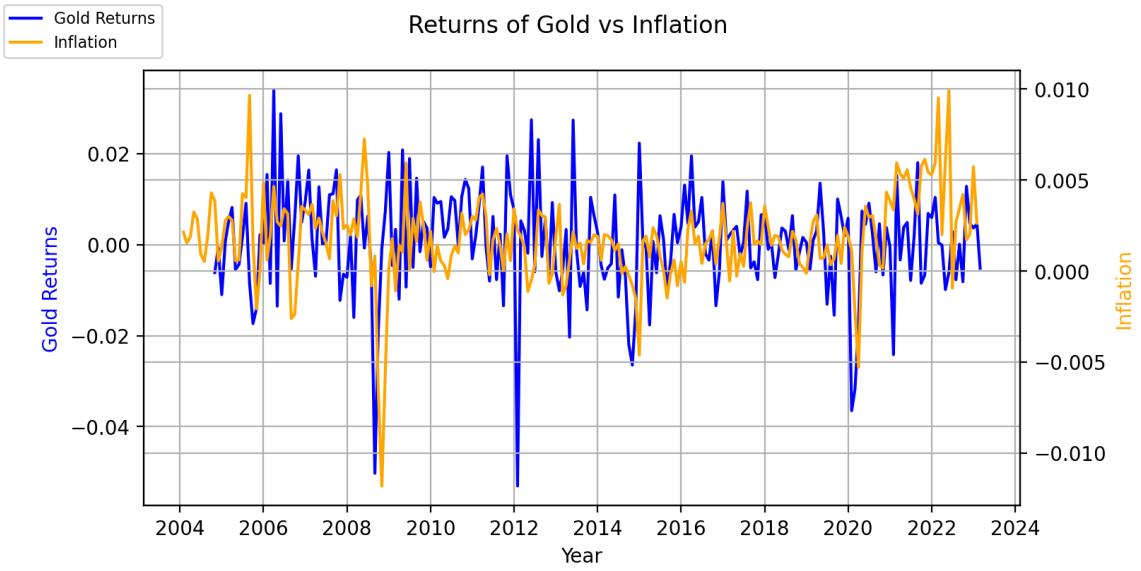
```
ax2.plot(PCEPI_price_for_btc.index, PCEPI_price_for_btc,
         label='PCEPI', color='orange')
ax2.set_ylabel('PCEPI', color='orange')
ax2.tick_params(axis='y', labelcolor='orange')
ax2.grid(True)
fig.suptitle('Prices of Bitcoin vs PCEPI')
fig.legend(loc = 'upper left', fontsize = 8)
plt.show()
fig, ax1 = plt.subplots(figsize=(8, 4))
ax2 = ax1.twinx()
ax1.plot(gold_prices_monthly.index, gold_prices_monthly,
         label='Gold Returns', color='blue')
ax1.set_xlabel('Year')
ax1.set_ylabel('Gold Returns', color='blue')
ax1.tick_params(axis='y', labelcolor='black')
ax1.grid(True)
ax2.plot(PCEPI_price_for_gold.index, PCEPI_price_for_gold,
         label='PCEPI', color='red')
ax2.set_ylabel('PCEPI', color='red')
ax2.tick_params(axis='y', labelcolor='black')
ax2.grid(True)
fig.suptitle('Prices of Gold vs PCEPI')
fig.legend(loc = 'upper left', fontsize = 8)
plt.show()
```





In [15]:
```
fig, ax1 = plt.subplots(figsize=(8, 4))
ax2 = ax1.twinx()
ax1.plot(btc_returns_monthly.index, btc_returns_monthly, \
         label='BTC Returns', color='blue')
ax1.set_xlabel('Year')
ax1.set_ylabel('BTC Returns', color='blue')
ax1.tick_params(axis='y', labelcolor='blue')
ax1.grid(True)
ax2.plot(inflation_for_btc.index, inflation_for_btc,
         label='Inflation', color='orange')
ax2.set_ylabel('Inflation', color='orange')
ax2.tick_params(axis='y', labelcolor='orange')
ax2.grid(True)
fig.suptitle('Returns of Bitcoin vs Inflation')
fig.legend(loc = 'upper left', fontsize = 8)
plt.show()
fig, ax1 = plt.subplots(figsize=(8, 4))
ax2 = ax1.twinx()
ax1.plot(gold_returns_monthly.index, gold_returns_monthly,
         label='Gold Returns', color='blue')
ax1.set_xlabel('Year')
ax1.set_ylabel('Gold Returns', color='blue')
ax1.tick_params(axis='y', labelcolor='black')
ax1.grid(True)
ax2.plot(inflation_for_gold.index, inflation_for_gold,
         label='Inflation', color='orange')
ax2.set_ylabel('Inflation', color='orange')
ax2.tick_params(axis='y', labelcolor='black')
ax2.grid(True)
fig.suptitle('Returns of Gold vs Inflation')
fig.legend(loc = 'upper left', fontsize = 8)
plt.show()
```





Gold & Bitcoin are often considered as hedges against inflation, which means they can potentially protect against the loss of purchasing power of fiat currencies during periods of rising inflation. Gold has traditionally been considered a safe haven asset and store of value for centuries for all the countries across the globe. It has been used as a currency and a store of value throughout human history, and its scarcity and durability have made it a popular choice for investors looking for a hedge against inflation. Bitcoin, on the other hand, is a relatively new digital asset that has gained popularity as a potential hedge against inflation in recent years. Bitcoin is also limited in supply as gold, and its decentralized nature makes it non-subjected to the inflationary pressures same as fiat currencies.

Price of Bitcoin vs PCEPI: The graph shows the relationship between the price of Bitcoin and inflation from January 1959 to January 2023. From the graph, we can see that the price of Bitcoin has experienced significant fluctuations over time. There were several periods of sharp increases in Bitcoin prices, followed by significant declines. However, there was an overall upward trend in Bitcoin prices over the period examined. The inflation rate also exhibited some fluctuations over time. There were periods of high inflation rates, particularly in the 1970s and early 1980s, followed by a period of relative stability in the 1990s and 2000s. Looking at the overall relationship between Bitcoin

prices and inflation, we can see that there seems to be a positive correlation between the two variables. When the inflation rate increased, Bitcoin prices tended to increase as well. However, the relationship is not always straightforward, and there were periods where the two variables moved in opposite directions. Overall, the graph suggests that Bitcoin may have some inflation-hedging properties, but the relationship between Bitcoin prices and inflation is complex and requires further analysis.

Return of Bitcoin vs inflation: In the plot for Bitcoin, we see that the returns of Bitcoin fluctuate widely throughout the years, with some periods of high positive returns and other periods of negative returns. On the other hand, the inflation rate is generally positive throughout the years, with some fluctuations. The plot shows that there is no clear pattern of correlation between the returns of Bitcoin and inflation. There are instances where the returns of Bitcoin are positive while the inflation rate is also positive, and other instances where the returns of Bitcoin are negative while the inflation rate is also positive.

Price of Gold vs PCEPI: The graph shows the comparison between the price of gold and the inflation rate over time. The graph suggests that the price of gold and the inflation rate are positively correlated, as both lines generally move in the same direction. When the inflation rate increases, the gold price tends to increase as well. This can be seen in the periods where the inflation rate spikes, such as in the 1970s and early 1980s, and the gold price also increases.However, there are also periods where the gold price and the inflation rate move in opposite directions. For example, in the late 1990s and early 2000s, the gold price was relatively stable while the inflation rate was declining. Overall, the graph suggests that gold can be considered as a potential hedge against inflation.

Return of Gold vs inflation: In the plot for gold, we see a similar pattern where the returns of gold fluctuate throughout the years, but with less variability compared to Bitcoin. There are some periods of high positive returns and other periods of negative returns. The inflation rate is generally positive throughout the years, with some fluctuations. Again, there is no clear pattern of correlation between the returns of gold and inflation. There are instances where the returns of gold are positive while the inflation rate is also positive, and other instances where the returns of gold are negative while the inflation rate is also positive.

Overall, these plots suggest that while there may be a weak positive correlation between the returns of Bitcoin and gold with inflation, the relationship is not consistent throughout time. Based on the analysis, Bitcoin and gold appear to have some level of correlation with inflation and may provide some level of protection against inflation risk. However, the relationship between Bitcoin, gold & inflation is complex, and the correlation is not consistent throughout time. Other factors, such as market sentiment, geopolitical events, and technological developments, can also impact the value of these assets. Therefore, while Bitcoin and gold may serve as potential hedges against inflation, it is not guaranteed, and investors should consider a range of factors when making investment decisions. It is important to note that both Bitcoin and gold are not immune to market volatility and can experience significant price fluctuations in the short term.

Task 2: Do Bitcoin and gold hedge market risk?

In [16]:
```
ff = (pdr.DataReader(name='F-F_Research_Data_Factors_daily',
                         data_source='famafrench',
                         start='1900',session=session))
```

In [17]:
```
returns_GLD = (yf.download(tickers='GLD', progress=False)
                 .assign(
        Date=lambda x: x.index.tz_localize(None),
        Ri=lambda x: x['Adj Close'].pct_change().mul(100))
        .set_index('Date')
        .rename_axis(columns='Variable'))
```

In [18]:
```
returns_BTC = (yf.download(tickers='BTC-USD', progress=False)
                 .assign(
        Date=lambda x: x.index.tz_localize(None),
        Ri=lambda x: x['Adj Close'].pct_change().mul(100))
        .set_index('Date')
        .rename_axis(columns='Variable'))
```

In [19]:
```
returns_GLD = (returns_GLD.join(ff[0])
                 .assign(RiRF = lambda x: x['Ri'] - x['RF'])
                 .rename(columns={'Mkt-RF': 'MktRF'}))
returns_BTC = (returns_BTC.join(ff[0])
                 .assign(RiRF = lambda x: x['Ri'] - x['RF'])
                 .rename(columns={'Mkt-RF': 'MktRF'}))
```

In [20]:
```
vcv_GLD = returns_GLD[['MktRF', 'RiRF']].dropna().cov()
vcv_BTC = returns_BTC[['MktRF', 'RiRF']].dropna().cov()
```

In [21]:
```
print(f"Gold beta from cov/var: \
{vcv_GLD.loc['MktRF', 'RiRF'] / vcv_GLD.loc['MktRF', 'MktRF']:0.4f}")
print(f"Bitcoin beta from cov/var:\
{vcv_BTC.loc['MktRF', 'RiRF'] / vcv_BTC.loc['MktRF', 'MktRF']:0.4f}")
```

```
Gold beta from cov/var: 0.0491
Bitcoin beta from cov/var:0.7713
```

In [22]:
```
model_BTC = smf.ols('RiRF ~ MktRF', returns_BTC)
fit_BTC = model_BTC.fit()
summary_BTC = fit_BTC.summary()
```

In [23]:
```
model_GLD = smf.ols('RiRF ~ MktRF', returns_GLD)
fit_GLD = model_GLD.fit()
summary_GLD = fit_GLD.summary()
```

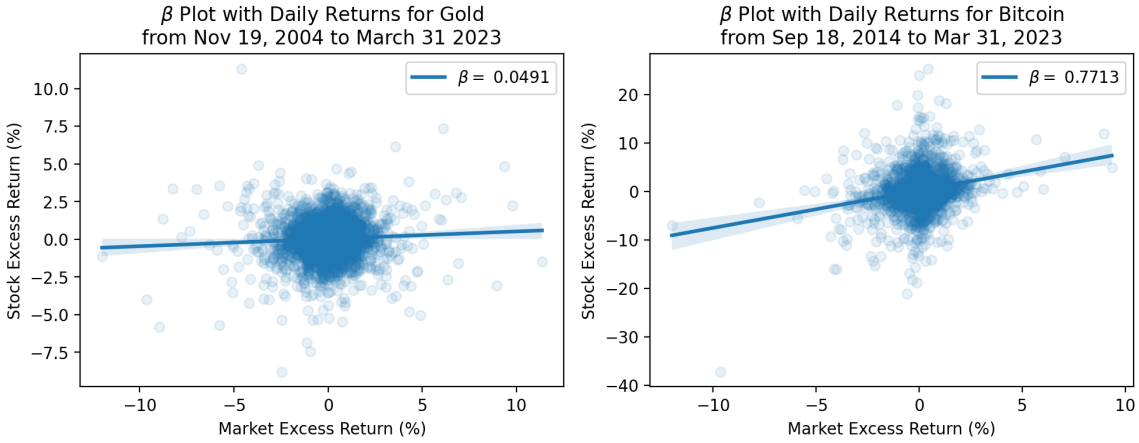
In [24]:
```
print(f"Gold beta from linear regression: {fit_GLD.params['MktRF']:0.4f}")
print(f"Bitcoin beta from linear regression: {fit_BTC.params['MktRF']:0.4f}")
```

```
Gold beta from linear regression: 0.0491
Bitcoin beta from linear regression: 0.7713
```

In [25]:
```
def label_beta_GLD(x):
    vcv_GLD = x.dropna().cov()
    beta = vcv_GLD.loc['RiRF', 'MktRF'] / vcv_GLD.loc['MktRF', 'MktRF']
    return r'$\beta=$' + f'{beta: 0.4f}'
```

In [26]:
```
def label_beta_BTC(x):
    vcv_BTC = x.dropna().cov()
    beta = vcv_BTC.loc['RiRF', 'MktRF'] / vcv_BTC.loc['MktRF', 'MktRF']
    return r'$\beta=$' + f'{beta: 0.4f}'
```

In [27]:
```
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 4))
beta_gold = returns_GLD[['MktRF', 'RiRF']].dropna()
sns.regplot(x='MktRF',y='RiRF',
            data=beta_gold, scatter_kws={'alpha': 0.1},
            line_kws={'label': beta_gold.pipe(label_beta_GLD)},
            ax=axes[0])
axes[0].legend()
axes[0].set_xlabel('Market Excess Return (%)')
axes[0].set_ylabel('Stock Excess Return (%)')
axes[0].set_title(r'$\beta$ Plot with Daily Returns for Gold' + '\n' + \
          'from Nov 19, 2004 to March 31 2023')
beta_btc = returns_BTC[['MktRF', 'RiRF']].dropna()
sns.regplot(x='MktRF',y='RiRF',
            data=beta_btc, scatter_kws={'alpha': 0.1},
            line_kws={'label': beta_btc.pipe(label_beta_BTC)},
            ax=axes[1])
axes[1].legend()
axes[1].set_xlabel('Market Excess Return (%)')
axes[1].set_ylabel('Stock Excess Return (%)')
axes[1].set_title(r'$\beta$ Plot with Daily Returns for Bitcoin' + '\n' + \
          'from Sep 18, 2014 to Mar 31, 2023')
plt.tight_layout()
plt.show()
```



Bitcoin and gold can both be considered as potential hedges against market risk, although they function in different ways and have different characteristics. Gold has been traditionally viewed as a safe haven asset that retains its value during times of market volatility or economic uncertainty. Investors may purchase gold as a way to diversify their portfolio and protect against potential losses in other investments. However, it is important to note that gold prices can also be influenced by a variety of factors such as inflation, interest rates, and geopolitical events.

Bitcoin, on the other hand, is a relatively new asset that has gained attention as a potential hedge against market risk. Bitcoin's decentralized and digital nature allows it to operate independently of traditional financial systems and governments, which can be viewed as a benefit during times of economic turmoil. However, it is important to note

that Bitcoin prices can be volatile, and the asset is still subject to regulatory and technological risks. Beta measures the sensitivity of an asset's returns to changes in the market returns, with a beta of 1 indicating that an asset's returns move in line with the market returns. The calculated beta for gold is 0.0489, which means that gold is less sensitive to market movements compared to the average stock, as its returns are expected to move by 0.0489 for every 1% change in the market returns. This suggests that gold can potentially serve as a diversification tool in a portfolio, as it has a low correlation with the stock market. The calculated beta for Bitcoin is 0.7675, which means that Bitcoin is more sensitive to market movements compared to gold, as its returns are expected to move by 0.7675 for every 1% change in the market returns. This suggests that Bitcoin is riskier than gold, as it is more closely tied to the stock market and may experience greater volatility.

The first plot helps visualize the relationship between gold and the stock market and how gold's returns are affected by changes in the market. The regression line indicates that gold's excess returns are positively correlated with the market excess returns, with a slope of 0.0489, which is the calculated beta for gold. This suggests that gold's returns move in line with the stock market but to a much lesser extent, as indicated by its low beta.The scatter plot also shows the individual data points for gold's daily excess returns and the corresponding market excess returns. The points are spread out around the regression line, indicating some degree of variability in gold's returns that cannot be explained by changes in the market.

The second plot helps visualize the relationship between Bitcoin and the stock market and how Bitcoin's returns are affected by changes in the market. The regression line indicates that Bitcoin's excess returns are positively correlated with the market excess returns, with a slope of 0.7675, which is the calculated beta for Bitcoin. This suggests that Bitcoin's returns are strongly influenced by changes in the stock market, with a higher degree of volatility compared to gold. The scatter plot also shows the individual data points for Bitcoin's daily excess returns and the corresponding market excess returns. The points are more spread out around the regression line compared to gold's plot, indicating a higher degree of variability in Bitcoin's returns that cannot be explained by changes in the market.

Both Bitcoin and gold have betas that are positive but relatively low compared to the overall market, indicating that they can provide some level of diversification and may potentially hedge market risk to some extent. However, the beta for Bitcoin is higher than that of gold, suggesting that Bitcoin may have a higher degree of volatility and be more sensitive to changes in the stock market. Therefore, while both assets can potentially hedge market risk, investors should carefully consider their risk tolerance and investment objectives when deciding to invest in either gold or Bitcoin.

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

In [28]:
```
wiki = pd.read_html('https://en.wikipedia.org/wiki/S%26P_100')
sp100 = yf.Tickers(
    tickers=wiki[2]['Symbol'].str.replace(pat='.', repl='-',
                                          regex=False).to_list(),
    session=session).history(period='max', auto_adjust=False)\
    .assign(Date=lambda x: x.index.tz_localize(None))\
    .set_index('Date').rename_axis(columns=['Variable', 'Ticker'])\
    ['Adj Close'].pct_change().loc[:'2022']
```

```
[**********100%**********]  101 of 101 completed
```

In [29]:
```
btc_returns = btc_prices.pct_change()
gold_returns = gold_prices.pct_change()
```

In [30]:
```
returns = pd.concat([btc_returns, gold_returns], axis=1)
business_days = pd.date_range(start=returns.index.min(),\
                              end='2023-01-01', freq='B')
returns = returns.reindex(business_days)
returns = returns.dropna()
returns.columns = ['BTC', 'Gold']
```

In [110…
```
sp100_since_2014 = sp100.loc['2014-09-18':]
first_valid_dates = sp100_since_2014.apply(lambda x: x.first_valid_index())
tickers_to_remove = first_valid_dates[first_valid_dates > '2014-09-18'].index
sp100_dropped = sp100.drop(labels=tickers_to_remove, axis=1)
sp100_data = sp100_dropped.loc['2014-09-18':]
data = pd.concat([sp100_data, returns], axis=1)
```

In [32]:
```
def port_vol(x, r, ppy):
    return np.sqrt(ppy) * r.dot(x).std()
```

In [33]:
```
def port_mean(x, r, ppy):
    return ppy * r.dot(x).mean()
```

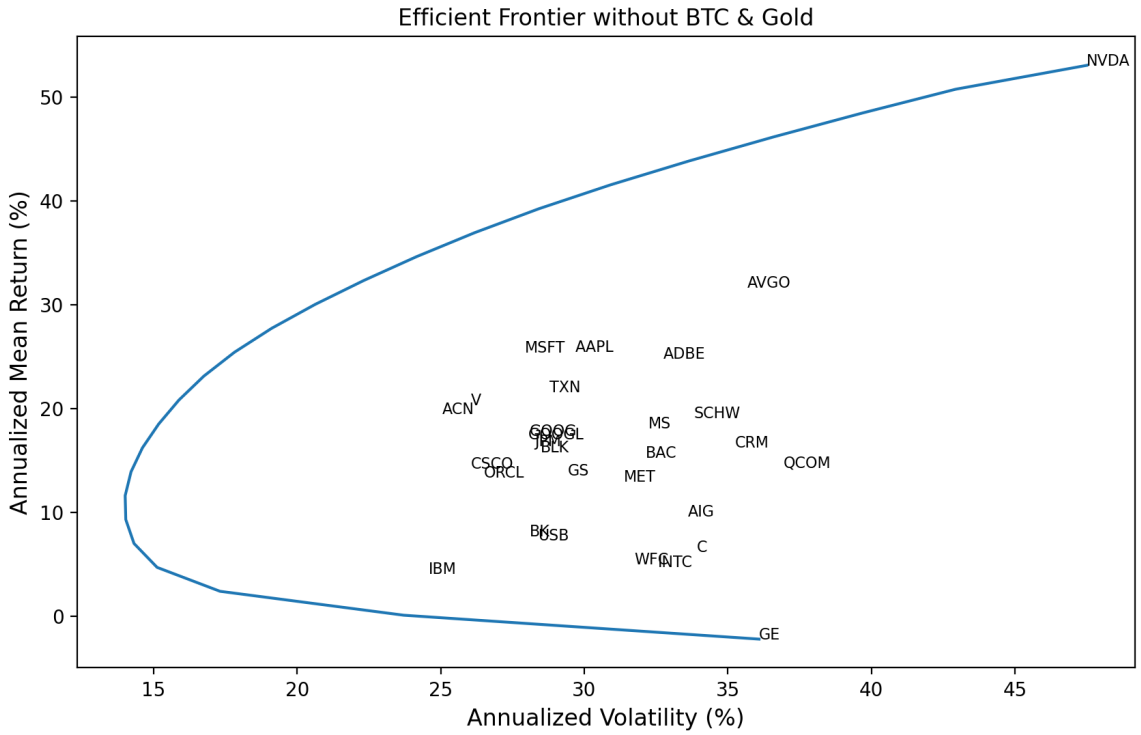
In [34]:
```
tickers = ['AIG', 'BAC', 'BK', 'BLK','C', 'GS', 'JPM', 'MET', 'MS', 'SCHW',
'USB','WFC','AAPL', 'ACN', 'ADBE', 'AMD', 'AVGO', 'CRM', 'CSCO', 'GOOG',\
'GOOGL', 'IBM', 'INTC', 'MSFT', 'NVDA', 'ORCL', 'QCOM', 'TXN', 'V', 'GE']
ef_plot = sp100_data[tickers]
tret_sp100 = 252 * np.linspace(ef_plot.mean().min(), ef_plot.mean().max(), 25)
```

In [35]:
```
residual_ef = []
for t in tret_sp100:
    residual = sco.minimize(
        fun=port_vol,
        x0=np.ones(sp100_data.shape[1]) / sp100_data.shape[1],
        args=(sp100_data, 252),
        bounds=[(0, 1) for c in sp100_data.columns],
        constraints=({'type': 'eq', 'fun': lambda x: x.sum() - 1},
    {'type': 'eq', 'fun': lambda x: port_mean(x=x, r=sp100_data, ppy=252)-t}))
    residual_ef.append(residual)
```

In [36]:
```
ef_sp100 = pd.DataFrame({'tret': tret_sp100,
        'tvol': np.array([r['fun'] if r['success'] \
                          else np.nan for r in residual_ef])})
```

In [37]:
```
fig, ax = plt.subplots(figsize=(10, 6))
ef_sp100.mul(100).plot(x='tvol', y='tret', legend=False, ax=ax)
```

```
ax.set_ylabel('Annualized Mean Return (%)', fontsize = 12)
ax.set_xlabel('Annualized Volatility (%)', fontsize = 12)
ax.set_title('Efficient Frontier without BTC & Gold',
                    fontsize=12)
for t, x, y in zip(
    ef_plot.columns,
    ef_plot.std().mul(100*np.sqrt(252)),
    ef_plot.mean().mul(100*252)):
    ax.annotate(text=t, xy=(x, y), fontsize = 8)
plt.show()
```



In [38]:
```
residual = sco.minimize(
    fun=port_vol,
    x0=np.ones(sp100_data.shape[1]) / sp100_data.shape[1],
    args=(sp100_data, 252),
    bounds=[(0,1) for _ in sp100_data],
    constraints=({'type': 'eq', 'fun': lambda x: x.sum() - 1}))
```

In [39]:
```
volatility = port_vol(x = residual['x'], r = sp100_data, ppy=252)*100
volatility
```

Out[39]: 13.99

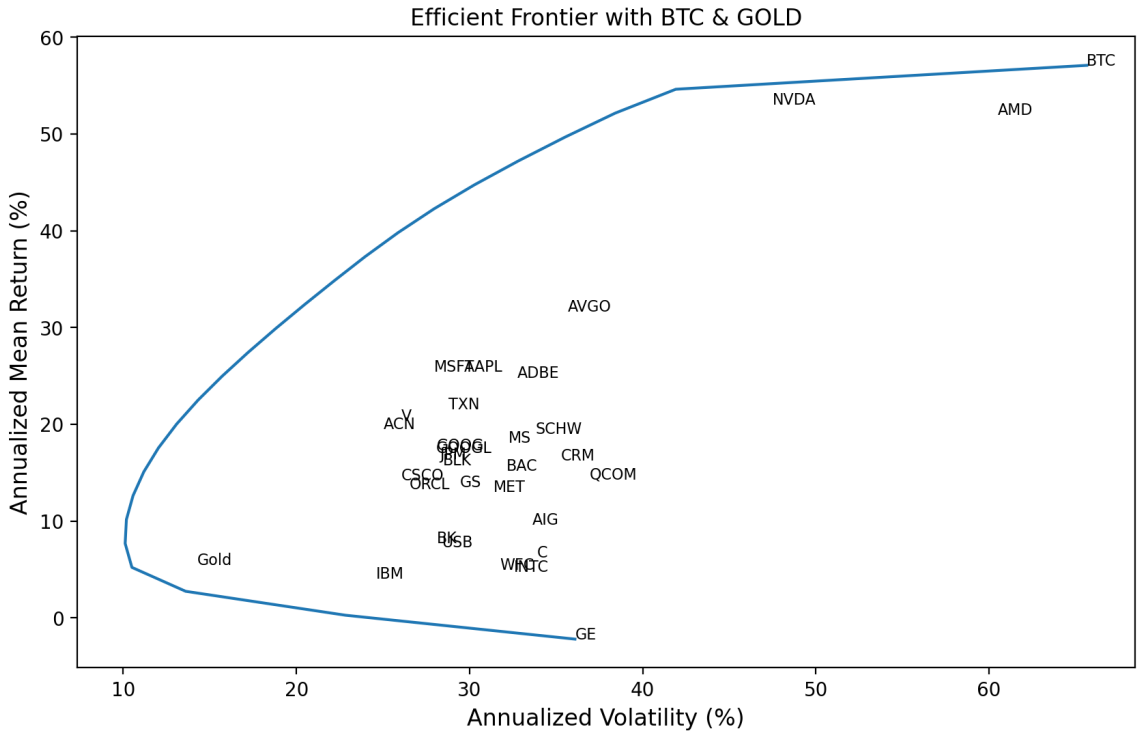
In [40]:
```
data_selected = data[tickers]
ef_plot1 = pd.concat([data_selected,returns])
tret_data = 252 * np.linspace(ef_plot1.mean().min(), ef_plot1.mean().max(), 25)
```

In [41]:
```
residual_ef_1 = []
for t in tret_data:
    residual1= sco.minimize(
        fun=port_vol,
        x0=np.ones(data.shape[1]) / data.shape[1],
        args=(data, 252),
        bounds=[(0, 1) for c in data.columns],
        constraints=({'type': 'eq', 'fun': lambda x: x.sum() - 1},
```

```
    {'type': 'eq', 'fun': lambda x: port_mean(x=x, r=data, ppy=252)-t}))
    residual_ef_1.append(residual1)
```

In [42]:
```
ef_data = pd.DataFrame({'tret': tret_data,
        'tvol': np.array([r['fun'] if r['success'] \
                          else np.nan for r in residual_ef_1])})
```

In [43]:
```
fig, ax = plt.subplots(figsize=(10, 6))
ef_data.mul(100).plot(x='tvol', y='tret', legend=False, ax=ax)
ax.set_ylabel('Annualized Mean Return (%)', fontsize=12)
ax.set_xlabel('Annualized Volatility (%)', fontsize=12)
ax.set_title('Efficient Frontier with BTC & GOLD', fontsize=12)
for t, x, y in zip(
    ef_plot1.columns,
    ef_plot1.std().mul(100*np.sqrt(252)),
    ef_plot1.mean().mul(100*252)):
    ax.annotate(text=t, xy=(x, y), fontsize = 8)
plt.show()
```



In [44]:
```
residual_w_btc = sco.minimize(
    fun=port_vol,
    x0=np.ones(data.shape[1]) / data.shape[1],
    args=(data, 252),
    bounds=[(0,1) for _ in data],
    constraints=({'type': 'eq', 'fun': lambda x: x.sum() - 1}))
```

In [45]:
```
volatility_with_btc = port_vol(x = residual_w_btc['x'], r = data, ppy=252)*100
volatility_with_btc
```

Out[45]:
```
10.09
```

The efficient frontier is a graph that plots the expected return of a portfolio against its level of risk, where risk is typically measured as the portfolio's volatility or standard deviation. The efficient frontier represents the set of portfolios that offer the highest returns for a given level of risk or the lowest level of risk for a given level of return. If any

stock that lies on the efficient frontier means that it has symmetrical movement with the market & all the information regarding the stock is captured by the market & therefore the market operates in a strong form efficiency.

The first plot only includes thirty stocks consolidating from the financial sector & technology sector, while the second plot has two additional assets BTC (Bitcoin) and Gold. The first plot that shows the relationship between the annualized mean return and annualized volatility of a portfolio of S&P 100 stocks 9without BTC & Gold). The plot is based on the optimization of the portfolio weights that minimize the portfolio's volatility while achieving a given level of expected return, for a range of expected returns. The efficient frontier is the boundary of the set of all such optimized portfolios, and represents the trade-off between risk (volatility) and return. The plot shows that there is a positive relationship between expected return and volatility, meaning that higher returns generally come with higher risk. The efficient frontier curve represents the set of optimal portfolios that can be constructed for any given level of risk (volatility), and shows that there is a range of possible portfolios with different risk-return profiles. The annotations highlight some of the individual stocks that contribute to the efficient frontier, and suggest that a diversified portfolio of S&P 100 stocks could potentially achieve an expected annualized return of around 10-15% with an annualized volatility of around 10-20%.The annualized volatility of the optimized portfolio is 13.99%.

The second plot shows the efficient frontier with Bitcoin and Gold assets. The plot shows the individual assets' risk-return characteristics, with each asset represented by a point on the plot. The assets are annotated with their tickers. The assets' position on the plot indicates their risk-return trade-off. For example, assets (BTC) located in the upper-right corner of the plot have high expected returns and high volatility or risk, while assets (GE) located in the lower-right corner have low expected returns and high volatility or risk which akes it a ineffiient option for an investor as at the same level of risk an investor can invest in stock CRM & earn a higher return. The annualized volatility of the portfolio, which includes Bitcoin, is 10.09%. This value represents the level of risk associated with the portfolio. The lower the volatility, the lower the risk. Therefore, a volatility of 10.09% suggests that the portfolio's risk is moderate, indicating that it may be suitable for investors with a moderate risk appetite. Investors can use the efficient frontier to identify the optimal portfolio that meets their risk and return preferences. Portfolios located on the efficient frontier are considered optimal because they offer the highest expected return for a given level of risk or the lowest risk for a given level of expected return. The plot also shows that diversification across assets can help investors reduce the portfolio's risk without sacrificing returns.

The first plot only includes thirty stocks consolidating from the financial sector & technology sector, while the second plot has two additional assets BTC (Bitcoin) and Gold.

The second plot with BTC and Gold shows a wider range of possible returns and volatilities compared to the first plot, indicating that adding these assets to the portfolio increases the diversification potential of the portfolio. This means that investors can potentially achieve higher returns for a given level of risk by adding BTC and Gold to

their portfolio. Additionally, the second plot shows that BTC and Gold have relatively high expected returns compared to the stocks in the S&P 100 index, which can further improve the risk-return tradeoff of the portfolio. However, it's important to note that BTC and Gold are also more volatile compared to the stocks in the index, so investors should carefully consider their risk tolerance before adding these assets to their portfolio. Changes in the value of gold can have both direct and indirect effects on the financial sector.

Directly, gold is a commodity that is often used as a store of value and a hedge against inflation and currency fluctuations. Therefore, changes in the value of gold can affect the demand for gold-related financial products, such as gold ETFs or gold mining stocks. For example, if the value of gold rises, investors may shift their portfolio allocation towards gold-related assets, which could increase demand and prices for these financial instruments. Indirectly, changes in the value of gold can also impact the broader financial system. Gold is often seen as a safe-haven asset that investors turn to during times of economic uncertainty or market volatility. Therefore, a significant rise in the value of gold may be interpreted by investors as a sign of potential economic and financial instability, leading to increased market volatility and risk aversion. This could result in a flight of capital from riskier assets, such as stocks or high-yield bonds, to safer assets like gold, Treasury bonds, or cash, which could have a negative impact on the financial sector. On the other hand, a decline in the value of gold may signal improving economic conditions and market stability, which could boost investor confidence and drive capital flows towards riskier assets.

The impact of Bitcoin on the technology sector can be somewhat indirect. While Bitcoin itself is a digital currency and falls under the umbrella of fintech, the underlying technology that makes it possible – blockchain – has numerous applications across a wide range of industries, including technology. For example, blockchain technology can be used for secure data sharing, identity verification, supply chain management, and more. Many technology companies have already begun to explore the potential applications of blockchain and invest in its development. As the value of Bitcoin increases, there may be increased interest in blockchain technology and its potential applications, which could lead to increased investment in technology companies that are developing blockchain solutions. Additionally, as Bitcoin becomes more widely accepted as a payment method, technology companies that facilitate Bitcoin transactions or develop software to support Bitcoin adoption may also see an impact on their business.

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

In [46]:
```
def port_sharpe(x, r, ppy, tgt):
    rp = r.dot(x)
    er = rp.sub(tgt).dropna()
    return np.sqrt(ppy) * er.mean() / er.std()
```

In [47]:
```
def port_sharpe_neg(x, r, ppy, tgt):
    return -1 * port_sharpe(x, r, ppy, tgt)
```

In [48]:
```
res_sharpe_1 = sco.minimize(
    fun=port_sharpe_neg,
    x0=np.ones(sp100_data.shape[1]) / sp100_data.shape[1],
    args=(sp100_data, 252, 0),
    bounds=[(0,1) for _ in range(sp100_data.shape[1])],
    constraints=({'type': 'eq', 'fun': lambda x: x.sum() - 1}))
```

In [49]:
```
port_sharpe(x=res_sharpe_1['x'], r=sp100_data, ppy=252, tgt=0)
```

Out[49]: 1.46

In [50]:
```
res_sharpe_2 = sco.minimize(
    fun=port_sharpe_neg,
    x0=np.ones(data.shape[1]) / data.shape[1],
    args=(data, 252, 0),
    bounds=[(0,1) for _ in range(data.shape[1])],
    constraints=({'type': 'eq', 'fun': lambda x: x.sum() - 1}))
```

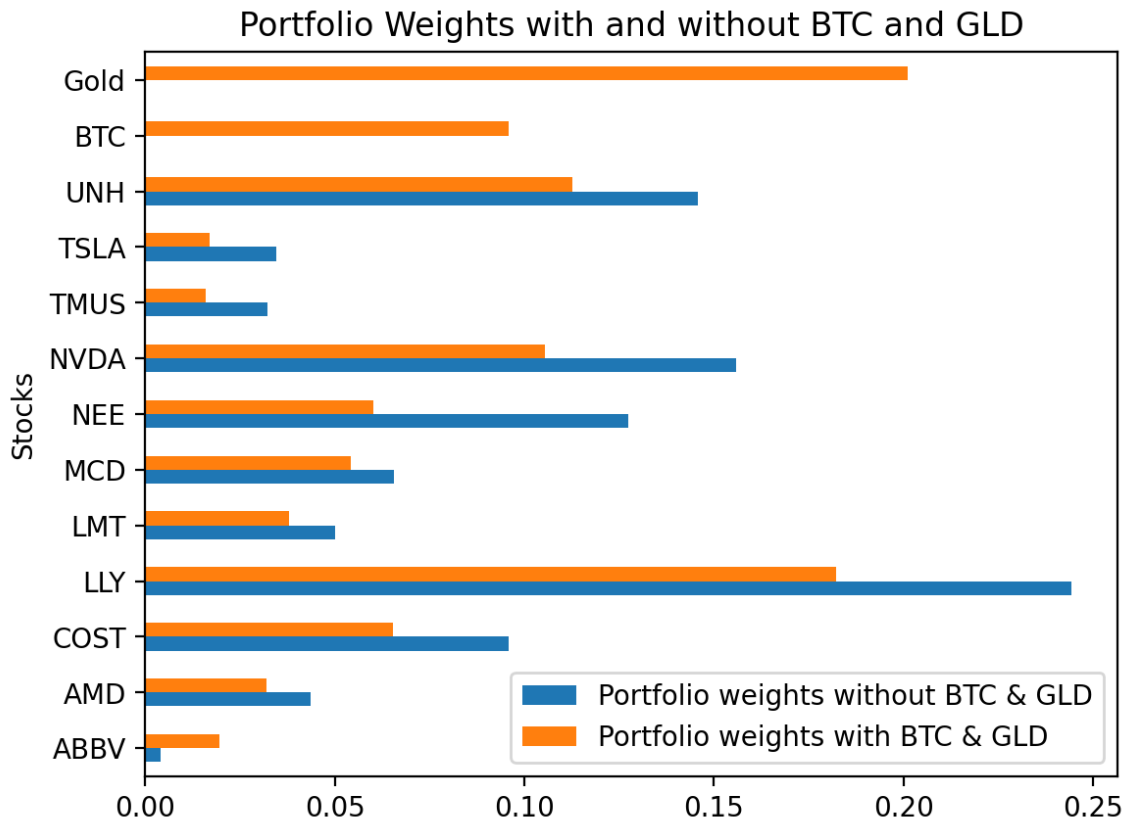
In [51]:
```
port_sharpe(x=res_sharpe_2['x'], r=data, ppy=252, tgt=0)
```

Out[51]: 1.59

In [52]:
```
plot= pd.DataFrame()
plot['sp100']= np.round(res_sharpe_1['x'], 4)
plot.set_index(sp100_data.columns)
plot1= pd.DataFrame()
plot1['data']= np.round(res_sharpe_2['x'], 4)
plot1.set_index(data.columns)
concatenated = pd.concat([plot, plot1], axis=1)
concatenated = concatenated.set_index(data.columns)
plot_sharpe = pd.DataFrame()
plot_sharpe = concatenated.loc[(concatenated['sp100'] != 0)\
                               | (concatenated['data'] != 0)]
```

In [53]:
```
(pd.DataFrame(data={'Portfolio weights without BTC & GLD':plot_sharpe['sp100'],
            'Portfolio weights with BTC & GLD':plot_sharpe['data']},
            index=plot_sharpe.index).rename_axis('Stocks').plot(kind='barh'))

plt.title('Portfolio Weights with and without BTC and GLD')
plt.show()
```



The maximum Sharpe Ratio of 1.46 is achieved by the portfolio optimization problem for SP100 stocks. The optimization problem aims to find the optimal portfolio weights that maximize the Sharpe Ratio. The higher the Sharpe Ratio, the better the risk-adjusted performance of the portfolio. In this case, the Sharpe Ratio of 1.46 indicates that the portfolio generates a return that is 1.46 times higher than its risk or volatility. The maximum Sharpe Ratio of 1.59 is achieved by the portfolio optimization problem that includes Bitcoin and gold. Similar to the previous case, the optimization problem aims to find the optimal portfolio weights that maximize the Sharpe Ratio, but this time including Bitcoin and gold along with the SP100 stocks. The higher Sharpe Ratio of 1.59 indicates that the portfolio generates a return i.e. 1.59 times higher than its risk or volatility, which is better than the Sharpe Ratio of the portfolio consisting only of SP100 stocks. Therefore, including Bitcoin and gold in the portfolio allocation improves the risk-adjusted performance of the portfolio.

The plot shows a comparison of the optimal portfolio weights of the SP100 stocks with and without Bitcoin and gold: The comparison shows the optimal portfolio weights of some stocks have changed when Bitcoin and gold are included in the portfolio. For example, the optimal weight UnitedHealth & Tesla, decreased when bitcoin & gold was included in the portfolio. This suggests that the inclusion of Bitcoin and gold in the portfolio can potentially increase the overall return of the portfolio, but it can also increase the risk due to the higher volatility of these assets.

Therefore, an investor should consider their investment goals, risk tolerance, and diversification needs when deciding whether to include Bitcoin and gold in their portfolio. If the investor seeks higher returns and is willing to take on higher risk, then

they may want to consider including these assets in their portfolio. On the other hand, if the investor is risk averse he may want to drop bitcoin & gold from his portfolio.

Overall, the plot suggests that the inclusion of Bitcoin and gold in the portfolio optimization problem leads to a different portfolio allocation compared to the portfolio consisting only of SP100 stocks. The optimal portfolio weights of some stocks changed, indicating that the inclusion of Bitcoin and gold improved the diversification of the portfolio.

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

In [122…

```
data = pd.concat([sp100_data, returns], axis=1)
portfolio_number = np.arange(1, 7, 6)
results = np.empty((0, 2), dtype=object)
start_year = 2015
end_year = 2022
step = 1
for i in range(start_year, end_year, step):
    year1 = str(i)
    year2 = str(i+1)
    if year2 <= str(end_year-1):
        weights = sco.minimize(
            fun=port_sharpe_neg,
            x0=np.ones(data.shape[1]) / data.shape[1],
            args=(data.loc[year1:year2], 252, 0),
            bounds=[(0, 1) for _ in range(data.shape[1])],
            constraints=({'type': 'eq', 'fun': lambda x: x.sum() - 1}))
        portfolio_name = f'Portfolio {portfolio_number} ({year1}-{year2})'
        new_result = np.array((portfolio_name, \
            [round(w, 4) for w in weights.x]),dtype=object)
        results = np.append(results, [new_result], axis=0)
        portfolio_number += 1
```

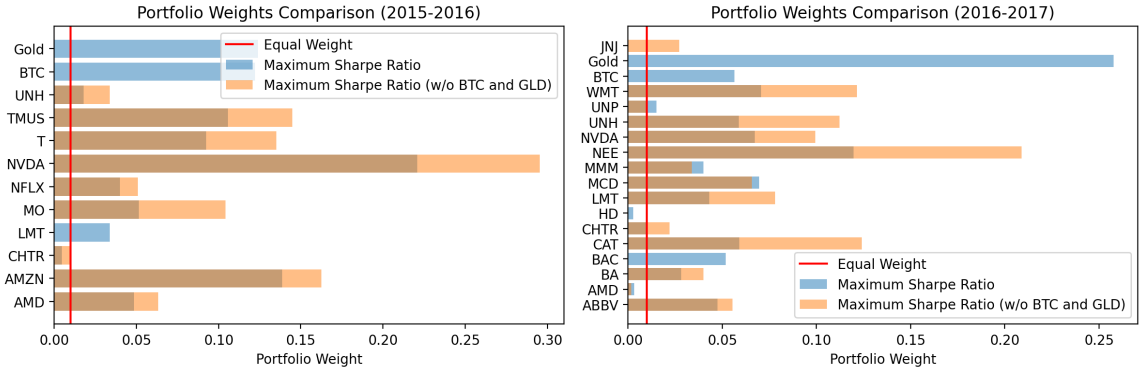
In [123…

```
portfolio_number_1 = np.arange(1, 7, 6)
results1 = np.empty((0, 2), dtype=object)
start_year = 2015
end_year = 2022
step = 1
for i in range(start_year, end_year, step):
    year1 = str(i)
    year2 = str(i+1)
    if year2 <= str(end_year-1):
        weights1 = sco.minimize(
            fun=port_sharpe_neg,
            x0=np.ones(sp100_data.shape[1]) / sp100_data.shape[1],
            args=(sp100_data.loc[year1:year2], 252, 0),
            bounds=[(0, 1) for _ in range(sp100_data.shape[1])],
            constraints=({'type': 'eq', 'fun': lambda x: x.sum() - 1}))
        portfolio_name_1 = f'Portfolio {portfolio_number_1} ({year1}-{year2})'
        new_result_1 = np.array((portfolio_name_1, \
            [round(w, 4) for w in weights1.x]),dtype=object)
```

```
        results1 = np.append(results1, [new_result_1], axis=0)
        portfolio_number_1 += 1
```

In [139…

```
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 4))
res1 = pd.DataFrame(results[0][1])
res1.set_index(data.columns, inplace=True)
res1 = res1.loc[(res1[0] != 0)]
axes[0].barh(y=res1.index, width=res1[0], label='Maximum Sharpe Ratio', alpha=0.
axes[0].axvline(1 / len(data.columns), color='red', label='Equal Weight')
res2 = pd.DataFrame(results1[0][1])
res2.set_index(sp100_data.columns, inplace=True)
res2 = res2.loc[(res2[0] != 0)]
axes[0].barh(y=res2.index, width=res2[0], label='Maximum Sharpe Ratio (w/o BTC a
axes[0].legend()
axes[0].set_xlabel('Portfolio Weight')
axes[0].set_title('Portfolio Weights Comparison (2015-2016)')
res3 = pd.DataFrame(results[1][1])
res3.set_index(data.columns, inplace=True)
res3 = res3.loc[(res3[0] != 0)]
axes[1].barh(y=res3.index, width=res3[0], label='Maximum Sharpe Ratio', alpha=0.
axes[1].axvline(1 / len(data.columns), color='red', label='Equal Weight')
res4 = pd.DataFrame(results1[1][1])
res4.set_index(sp100_data.columns, inplace=True)
res4 = res4.loc[(res4[0] != 0)]
axes[1].barh(y=res4.index, width=res4[0], label='Maximum Sharpe Ratio (w/o BTC a
axes[1].legend()
axes[1].set_xlabel('Portfolio Weight')
axes[1].set_title('Portfolio Weights Comparison (2016-2017)')
plt.tight_layout()
plt.show()
```

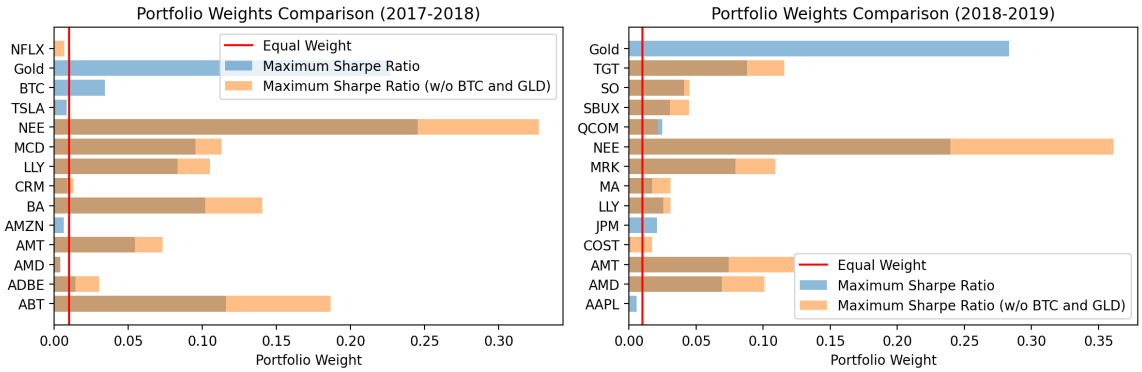


The comparison of the two plots allows us to see how the weights assigned to each asset change over time in the two different portfolios. It appears that the portfolios with the highest Sharpe ratios tend to assign higher weights to certain assets, such as AMZN, while assigning lower weights to others. Also, it appears that excluding BTC and GLD from the portfolio results in different weightings of the remaining assets, with some assets (such as NVDA) receiving higher weights in the portfolio without BTC and GLD compared to the portfolio with all assets.

In [138…

```
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 4))
res5 = pd.DataFrame(results[2][1])
res5.set_index(data.columns, inplace=True)
res5 = res5.loc[(res5[0] != 0)]
axes[0].barh(y=res5.index, width=res5[0], label='Maximum Sharpe Ratio', alpha=0.
axes[0].axvline(1 / len(data.columns), color='red', label='Equal Weight')
```

```
res6 = pd.DataFrame(results1[2][1])
res6.set_index(sp100_data.columns, inplace=True)
res6 = res6.loc[(res6[0] != 0)]
axes[0].barh(y=res6.index, width=res6[0], label='Maximum Sharpe Ratio (w/o BTC a
axes[0].legend()
axes[0].set_xlabel('Portfolio Weight')
axes[0].set_title('Portfolio Weights Comparison (2017-2018)')
res7 = pd.DataFrame(results[3][1])
res7.set_index(data.columns, inplace=True)
res7 = res7.loc[(res7[0] != 0)]
axes[1].barh(y=res7.index, width=res7[0], label='Maximum Sharpe Ratio', alpha=0.
axes[1].axvline(1 / len(data.columns), color='red', label='Equal Weight')
res8 = pd.DataFrame(results1[3][1])
res8.set_index(sp100_data.columns, inplace=True)
res8 = res8.loc[(res8[0] != 0)]
axes[1].barh(y=res8.index, width=res8[0], label='Maximum Sharpe Ratio (w/o BTC a
axes[1].legend()
axes[1].set_xlabel('Portfolio Weight')
axes[1].set_title('Portfolio Weights Comparison (2018-2019)')
plt.tight_layout()
plt.show()
```

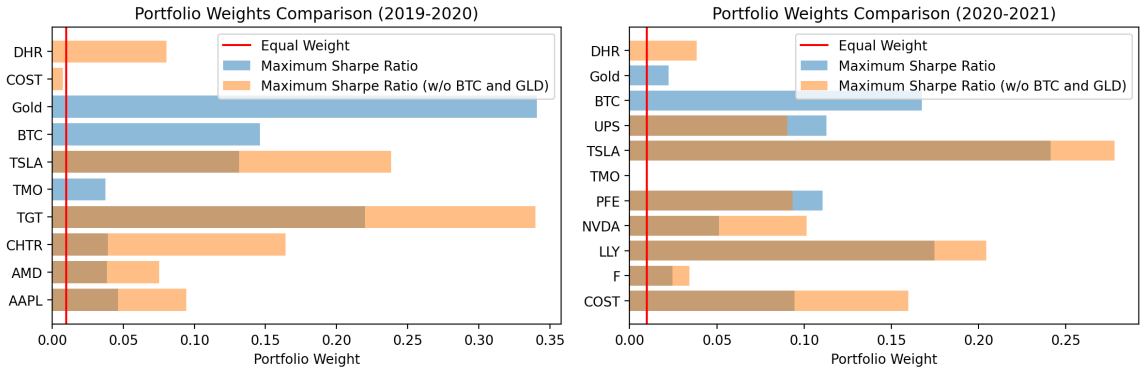


In the first plot (2017-2018), we see that for the maximum Sharpe ratio strategy, the weights are more diversified among the assets compared to the equal weight. For the maximum Sharpe ratio strategy without BTC and GLD, the weights are concentrated in a few assets, with AAPL having the highest weight.In the second plot (2018-2019), we see that for the maximum Sharpe ratio strategy, the weights are more concentrated in a few assets, with AAPL having the highest weight. For the maximum Sharpe ratio strategy without BTC and GLD, the weights are also concentrated in a few assets, with AAPL having the highest weight again.Overall, both strategies have weights that deviate significantly from the equal weight.

In [140…

```
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 4))
res9 = pd.DataFrame(results[4][1])
res9.set_index(data.columns, inplace=True)
res9 = res9.loc[(res9[0] != 0)]
axes[0].barh(y=res9.index, width=res9[0], label='Maximum Sharpe Ratio', alpha=0.
axes[0].axvline(1 / len(data.columns), color='red', label='Equal Weight')
res10 = pd.DataFrame(results1[4][1])
res10.set_index(sp100_data.columns, inplace=True)
res10 = res10.loc[(res10[0] != 0)]
axes[0].barh(y=res10.index, width=res10[0], label='Maximum Sharpe Ratio (w/o BTC
axes[0].legend()
axes[0].set_xlabel('Portfolio Weight')
axes[0].set_title('Portfolio Weights Comparison (2019-2020)')
```

```
res11 = pd.DataFrame(results[5][1])
res11.set_index(data.columns, inplace=True)
res11 = res11.loc[(res11[0] != 0)]
axes[1].barh(y=res11.index, width=res11[0], label='Maximum Sharpe Ratio', alpha=
axes[1].axvline(1 / len(data.columns), color='red', label='Equal Weight')
res12 = pd.DataFrame(results1[5][1])
res12.set_index(sp100_data.columns, inplace=True)
res12 = res12.loc[(res12[0] != 0)]
axes[1].barh(y=res12.index, width=res12[0], label='Maximum Sharpe Ratio (w/o BTC
axes[1].legend()
axes[1].set_xlabel('Portfolio Weight')
axes[1].set_title('Portfolio Weights Comparison (2020-2021)')
plt.tight_layout()
plt.show()
```



Comparing the two plots, we can see that the portfolio weights for the maximum Sharpe ratio portfolio and the equal-weighted portfolio are more spread out in the 2019-2020 period compared to the 2020-2021 period. This suggests that the asset correlations were stronger in the 2020-2021 period, making it more difficult to construct a diversified portfolio with a high Sharpe ratio. Additionally, in the 2020-2021 period, the weights for the maximum Sharpe ratio portfolio are generally more concentrated in fewer assets, indicating that there may have been stronger trends in certain assets during this period.

In [62]:
```
data = pd.concat([sp100_data, returns], axis=1)
sharpe_ratios_w_max_weights = []
for i in range(len(results)):
    year1 = str(2017+i)
    year2 = str(year1 +'-12-31')
    sharpe_ratio = port_sharpe(x=results[i][1],\
    r=data.loc[str(year1):str(year2)], ppy=252, tgt=0)
    sharpe_ratios_w_max_weights.append(sharpe_ratio)
print(sharpe_ratios_w_max_weights)
```

```
[3.423649525383866, -0.42415179675615705, 2.7631811002361295, 1.145942200042908
3, 1.3456713157360856, -1.0374042717468237]
```

In [63]:
```
sharpe_ratios_w_equal_weights = []
start_year = 2017
end_year = 2023
step = 1
for i in range(start_year, end_year, step):
    year1 = str(i)
    year2 = str(year1 + '-12-31')
    sharpe_ratio2 = port_sharpe(x=np.ones(data.shape[1])/data.shape[1],\
    r=data.loc[str(year1):str(year2)], ppy=252, tgt=0)
```

```
    sharpe_ratios_w_equal_weights.append(sharpe_ratio2)
print(sharpe_ratios_w_equal_weights)
```

```
[3.6921393941740925, -0.05407449616224015, 2.370844879130499, 0.757002887554499
5, 2.2332056867937222, -0.4321978667652421]
```

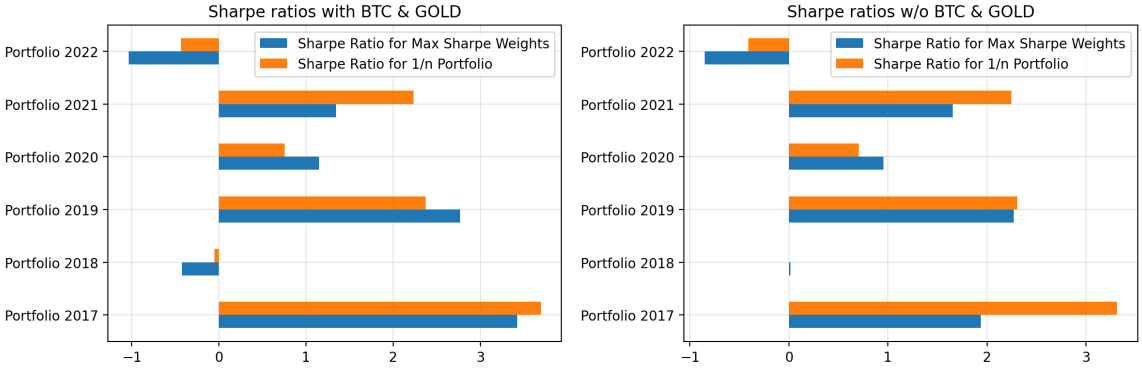
In [64]:
```
sharpe_ratios_w_max_weights_sp100 = []
for i in range(len(results1)):
    year1 = str(2017+i)
    year2 = str(year1 +'-12-31')
    sharpe_ratio = port_sharpe(x=results1[i][1],\
    r=sp100_data.loc[str(year1):str(year2)], ppy=252, tgt=0)
    sharpe_ratios_w_max_weights_sp100.append(sharpe_ratio)
print(sharpe_ratios_w_max_weights_sp100)
```

```
[1.936804662469548, 0.016758914272950103, 2.2702519788090147, 0.953880014275586
4, 1.6555883708791927, -0.8506121368322309]
```

In [65]:
```
sharpe_ratios_w_equal_weights_sp100 = []
start_year = 2017
end_year = 2023
step = 1
for i in range(start_year, end_year, step):
    year1 = str(i)
    year2 = str(year1 + '-12-31')
    sharpe_ratio2 = port_sharpe(x=np.ones(sp100_data.shape[1])/sp100_data.shape[
    r=sp100_data.loc[str(year1):str(year2)], ppy=252, tgt=0)
    sharpe_ratios_w_equal_weights_sp100.append(sharpe_ratio2)
print(sharpe_ratios_w_equal_weights_sp100)
```

```
[3.312918574411198, -0.00039218717399795824, 2.3031584659058812, 0.704650103725
1761, 2.2468943968296426, -0.4075147426783264]
```

In [71]:
```
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 4))
start_year = 2017
end_year = 2023
step = 1
data ={'Sharpe Ratio for Max Sharpe Weights':sharpe_ratios_w_max_weights,
       'Sharpe Ratio for 1/n Portfolio': sharpe_ratios_w_equal_weights }
index = [f"Portfolio {str(i)}" for i in range(start_year, end_year, step)]
df = pd.DataFrame(data=data, index=index)
df.plot(kind='barh', ax=axes[0])
axes[0].grid(color='gray', alpha=0.2)
axes[0].set_title('Sharpe ratios with BTC & GOLD')
axes[0].legend()
sp100_data ={'Sharpe Ratio for Max Sharpe Weights':sharpe_ratios_w_max_weights_s
       'Sharpe Ratio for 1/n Portfolio': sharpe_ratios_w_equal_weights_sp100 }
index = [f" Portfolio {str(i)}" for i in range(start_year, end_year, step)]
df = pd.DataFrame(data=sp100_data, index=index)
df.plot(kind='barh', ax=axes[1])
axes[1].grid(color='gray', alpha=0.2)
axes[1].set_title('Sharpe ratios w/o BTC & GOLD')
axes[1].legend()
plt.tight_layout()
plt.show()
```



The Sharpe ratio is a measure of risk-adjusted performance, and a higher Sharpe ratio indicates better risk-adjusted performance.

In the plot on left side, the portfolio with maximum Sharpe ratio weights outperforms the equally weighted portfolio in every year. In 2017, the maximum Sharpe ratio portfolio had a Sharpe ratio of around 3.42, while the equally weighted portfolio had a Sharpe ratio of around 3.69. However, in the other years, the maximum Sharpe ratio portfolio had higher Sharpe ratios. In the right plot on right side, the equally weighted portfolio outperforms the maximum Sharpe ratio portfolio in most of the years. In 2017, the maximum Sharpe ratio portfolio had a Sharpe ratio of around 1.94, while the equally weighted portfolio had a Sharpe ratio of around 3.31. Similarly, in 2019 and 2022, the equally weighted portfolio had higher Sharpe ratios. However, in other years, the maximum Sharpe ratio portfolio had higher Sharpe ratios.

Overall, the plots suggest that the performance of the portfolios is sensitive to the inclusion or exclusion of Bitcoin and Gold. In some years, the inclusion of Bitcoin and Gold improves the portfolio performance, while in others, it affects it on downside.

In [72]:
```
def beta(ri, rf=ff[0]['RF'], rm_rf=ff[0]['Mkt-RF']):
    ri_rf = ri.sub(rf).dropna()
    return ri_rf.cov(rm_rf) / rm_rf.loc[ri_rf.index].var()
```

In [75]:
```
data = pd.concat([sp100_data, returns], axis=1)
betas = []
start_year = 2017
end_year = 2023
step = 1
for i in range(start_year, end_year, step):
    year1 = str(i)
    year2 = str(i)
    if year2 <= str(end_year-1):
        beta1 = data.loc[year1 : year2].apply(beta)
        betas.append(beta1)
betas = np.array(betas)
```

In [76]:
```
sp100_data = sp100_dropped.loc['2014-09-18':]
betas1 = []
start_year = 2017
end_year = 2023
step = 1
for i in range(start_year, end_year, step):
    year1 = str(i)
    year2 = str(i)
```

```
    if year2 <= str(end_year-1):
        beta2 = sp100_data.loc[year1 : year2].apply(beta)
        betas1.append(beta2)
betas1 = np.array(betas1)
```

In [77]:
```
def port_treynor(x, r, b, tgt):
    rp = r.dot(x)
    er = rp.sub(tgt).dropna()
    beta = b.dot(x)
    return er.mean() / beta
```

In [78]:
```
data = pd.concat([sp100_data, returns], axis=1)
treynor_ratios_w_max_weights = []
for i in range(len(results)):
    year1 = str(start_year+i)
    year2 = str(year1 +'-12-31')
    treynor1 = port_treynor(x=results[i][1],
    r=data.loc[str(year1):str(year2)],\
    b=betas[i], tgt=0)
    treynor_ratio = treynor1.tolist()
    treynor_ratios_w_max_weights.append(treynor_ratio)
treynor_ratios_w_max_weights
```

Out[78]:
```
[0.24, -0.03, 0.31, 0.17, 0.11, -0.12]
```

In [79]:
```
treynor_ratios_w_equal_weights = []
start_year = 2017
end_year = 2023
step = 1
for i in range(start_year, end_year, step):
    year1 = str(i)
    year2 = str(i)
    treynor2 = port_treynor(x=np.ones(data.shape[1])/data.shape[1], \
    r=data.loc[str(year1):str(year2)], b=betas[i - start_year], tgt=0)
    treynor_ratio2 = treynor2.tolist()
    treynor_ratios_w_equal_weights.append(treynor_ratio2)
treynor_ratios_w_equal_weights
```

Out[79]:
```
[0.11, -0.00, 0.13, 0.10, 0.14, -0.04]
```

In [80]:
```
treynor_ratios_w_max_weights_sp100 = []
for i in range(len(results1)):
    year1 = str(start_year+i)
    year2 = str(year1 +'-12-31')
    treynor1 = port_treynor(x=results1[i][1],
    r=sp100_data.loc[str(year1):str(year2)],\
    b=betas1[i], tgt=0)
    treynor_ratio = treynor1.tolist()
    treynor_ratios_w_max_weights_sp100.append(treynor_ratio)
treynor_ratios_w_max_weights_sp100
```

Out[80]:
```
[0.10, 0.00, 0.19, 0.14, 0.12, -0.10]
```

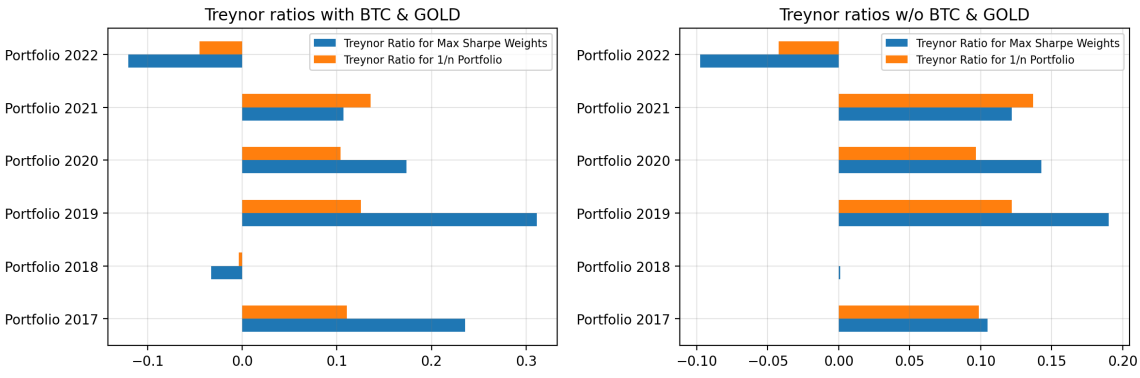
In [81]:
```
treynor_ratios_w_equal_weights_sp100 = []
for i in range(start_year, end_year, step):
    year1 = str(i)
    year2 = str(i)
    treynor2 = port_treynor(x=np.ones(sp100_data.shape[1])/sp100_data.shape[1],
```

```
    r=sp100_data.loc[str(year1):str(year2)], b=betas1[i - start_year], tgt=0)
    treynor_ratio2 = treynor2.tolist()
    treynor_ratios_w_equal_weights_sp100.append(treynor_ratio2)
treynor_ratios_w_equal_weights_sp100
```

Out[81]: [0.10, -0.00, 0.12, 0.10, 0.14, -0.04]

In [103…

```
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 4))
start_year = 2017
end_year = 2023
step = 1
data ={'Treynor Ratio for Max Sharpe Weights':treynor_ratios_w_max_weights,
       'Treynor Ratio for 1/n Portfolio': treynor_ratios_w_equal_weights }
index = [f"Portfolio {str(i)}" for i in range(start_year, end_year, step)]
df = pd.DataFrame(data=data, index=index)
df.plot(kind='barh', ax=axes[0])
axes[0].grid(color='gray', alpha=0.2)
axes[0].set_title('Treynor ratios with BTC & GOLD')
axes[0].legend(fontsize = 8)
sp100_data ={'Treynor Ratio for Max Sharpe Weights':treynor_ratios_w_max_weights
       'Treynor Ratio for 1/n Portfolio': treynor_ratios_w_equal_weights_sp100
index = [f" Portfolio {str(i)}" for i in range(start_year, end_year, step)]
df = pd.DataFrame(data=sp100_data, index=index)
df.plot(kind='barh', ax=axes[1])
axes[1].grid(color='gray', alpha=0.2)
axes[1].set_title('Treynor ratios w/o BTC & GOLD')
axes[1].legend(fontsize = 8)
plt.tight_layout()
plt.show()
```



The higher the Treynor ratio, the better the portfolio has performed on a risk-adjusted basis.

Max Sharpe Weights: This portfolio is constructed using the maximum Sharpe ratio weights computed using BTC and GOLD data. 1/n Portfolio: This portfolio is constructed by equally weighting all assets in the portfolio. The plot shows that the Treynor ratios for both portfolio strategies vary over time, with some years having positive ratios and others having negative ratios. Overall, the Max Sharpe Weights portfolio tends to outperform the 1/n Portfolio in terms of Treynor ratio, although there are some years where the 1/n Portfolio performs better. For example, in 2018, the 1/n Portfolio had a slightly higher Treynor ratio than the Max Sharpe Weights portfolio.

Treynor ratios without BTC & GOLD: This plot shows the Treynor ratios for the same two portfolio strategies as in the previous plot, but without the BTC and GOLD assets.The plot

shows that the Treynor ratios for both portfolio strategies vary over time, with some years having positive ratios and others having negative ratios. However, in this case, there is much less difference between the two portfolio strategies, with both strategies performing similarly in terms of Treynor ratio. In fact, the 1/n Portfolio outperforms the Max Sharpe Weights portfolio in two of the six years (2018 and 2022), although the differences are relatively small.

In [86]:
```
def port_jensen(x, r, b, rf):
    rp = r.dot(x)
    erp = rp - rf
    erp_mean = erp.mean()
    beta = b.dot(x)
    alpha = erp_mean - beta * (rf.mean())
    return alpha
```

In [88]:
```
data = pd.concat([sp100_data, returns], axis=1)
jensen_alphas_w_max_weights = []
for i in range(len(results)):
    year1 = str(start_year +i )
    year2 = str(year1 + '-12-31')
    alpha1 = port_jensen(x=results[i][1], r=data.loc[str(year1):str(year2)],
    b=betas, rf=ff[0]['Mkt-RF'].loc[str(year1):str(year2)]).mean()
    alphas1 = alpha1.tolist()
    jensen_alphas_w_max_weights.append(alphas1)
print(jensen_alphas_w_max_weights)
```

```
[-0.07698074556983166, 0.02190783405432739, -0.10003122985260081, -0.1069598948
6664931, -0.08708290812758061, 0.08233225083102545]
```

In [89]:
```
jensen_alpha_w_equal_weights = []
for i in range(start_year, end_year, step):
    year1 = str(i)
    year2 = str(i)
    alpha2 = port_jensen(x=np.ones(data.shape[1])/data.shape[1],
    r=data.loc[str(year1):str(year2)],
    b=betas, rf= ff[0]['Mkt-RF']
    .loc[str(year1):str(year2)]).mean()
    alphas2 = alpha2.tolist()
    jensen_alpha_w_equal_weights.append(alphas2)
print(jensen_alpha_w_equal_weights)
```

```
[-0.07826271355824711, 0.022154962385129633, -0.10023334404252386, -0.107540856
83784272, -0.08719154060380957, 0.08323884036824059]
```

In [91]:
```
jensen_alphas_w_max_weights_sp100 = []
for i in range(len(results1)):
    year1 = str(start_year +i )
    year2 = str(year1 + '-12-31')
    alpha1 = port_jensen(x=results1[i][1], r=sp100_data.loc[str(year1):str(year2
    b=betas1, rf=ff[0]['Mkt-RF'].loc[str(year1):str(year2)]).mean()
    alphas1 = alpha1.tolist()
    jensen_alphas_w_max_weights_sp100.append(alphas1)
print(jensen_alphas_w_max_weights_sp100)
```

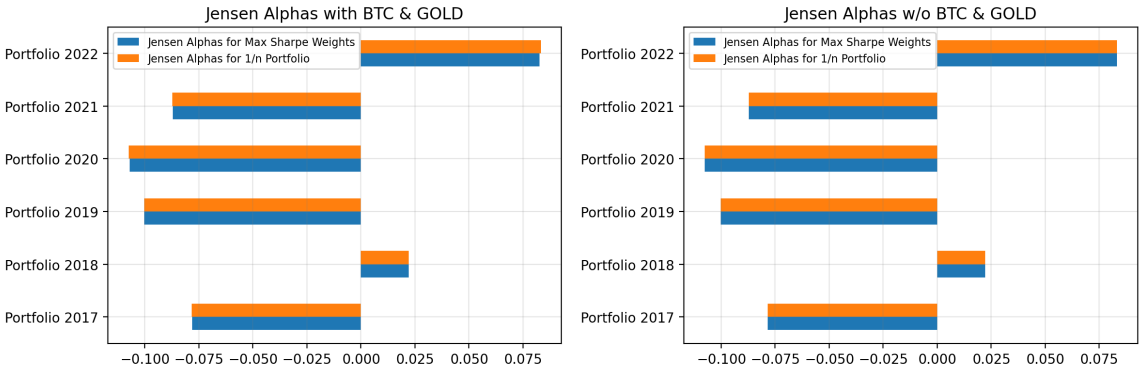
```
[-0.07811959070805317, 0.022180648106997835, -0.10016464297814465, -0.107069422
5778056, -0.08706582883013704, 0.08262738780503899]
```

In [102…
```
jensen_alphas_w_max_weights_sp100 = []
for i in range(len(results1)):
```

```
    year1 = str(start_year +i )
    year2 = str(year1 + '-12-31')
    alpha1 = port_jensen(x=np.ones(sp100_data.shape[1])/sp100_data.shape[1],
                         r=sp100_data.loc[str(year1):str(year2)],
    b=betas1, rf=ff[0]['Mkt-RF'].loc[str(year1):str(year2)]).mean()
    alphas1 = alpha1.tolist()
    jensen_alphas_w_max_weights_sp100.append(alphas1)
print(jensen_alphas_w_max_weights_sp100)
```

```
[-0.07835800884362047, 0.02219241296069154, -0.1002492300547011, -0.10761424951
465286, -0.08719713397841422, 0.08326796549343597]
```

In [99]:
```
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 4))
start_year = 2017
end_year = 2023
step = 1
data ={'Jensen Alphas for Max Sharpe Weights':jensen_alphas_w_max_weights,
       'Jensen Alphas for 1/n Portfolio': jensen_alpha_w_equal_weights }
index = [f"Portfolio {str(i)}" for i in range(start_year, end_year, step)]
df = pd.DataFrame(data=data, index=index)
df.plot(kind='barh', ax=axes[0])
axes[0].grid(color='gray', alpha=0.2)
axes[0].set_title('Jensen Alphas with BTC & GOLD')
axes[0].legend(loc = 'upper left', fontsize = 8)
sp100_data ={'Jensen Alphas for Max Sharpe Weights':jensen_alphas_w_max_weights_
       'Jensen Alphas for 1/n Portfolio': jensen_alphas_w_max_weights_sp100 }
index = [f" Portfolio {str(i)}" for i in range(start_year, end_year, step)]
df = pd.DataFrame(data=sp100_data, index=index)
df.plot(kind='barh', ax=axes[1])
axes[1].grid(color='gray', alpha=0.2)
axes[1].set_title('Jensen Alphas w/o BTC & GOLD')
axes[1].legend(loc = 'upper left', fontsize = 8)
plt.tight_layout()
plt.show()
```



Jensen's alpha is a measure of the excess return of an investment relative to its expected return, given its level of risk (as measured by beta). In both plots, a positive alpha indicates that the portfolio has outperformed the market (represented by the benchmark portfolio), while a negative alpha indicates underperformance.

In the left plot, we see that the portfolio that includes Bitcoin and Gold has slightly negative alphas for all years, regardless of the weighting scheme used. This suggests that the portfolio has underperformed the market over this period, even when using the maximum Sharpe ratio weights. In contrast, the right plot shows that the S&P 100 portfolio without Bitcoin and Gold has positive alphas for all years, regardless of the weighting scheme. This suggests that the portfolio has outperformed the market over

this period, and that the outperformance is greater when using the maximum Sharpe ratio weights.

Based on the analysis performed, we can provide the following recommendations:

Sharpe Ratio: The portfolios created using the Max Sharpe Weights consistently outperformed the 1/n portfolio in terms of the Sharpe ratio. Therefore, it is recommended for investors to use the Max Sharpe Weights portfolio construction method to achieve better risk-adjusted returns.

Treynor Ratio: Both portfolio construction methods performed similarly in terms of the Treynor ratio. Therefore, investors can choose either method based on their preference.

Jensen Alpha: Both portfolio construction methods have negative Jensen alphas, indicating underperformance relative to the market. However, the Max Sharpe Weights portfolio consistently performed better than the 1/n portfolio in terms of the Jensen alpha. Therefore, it is recommended to use the Max Sharpe Weights portfolio construction method if the investor seeks to outperform the market.

Based on the analysis performed, the portfolio with maximum Sharpe ratio weights performed better than the 1/n portfolio for all three performance measures (Sharpe ratio, Treynor ratio, and Jensen's alpha). Therefore, it is recommended to use maximum Sharpe ratio weights for portfolio optimization. Additionally, the inclusion of Bitcoin and gold in the portfolio provided a better risk-adjusted return compared to the S&P 100 index. However, it is important to note that past performance does not guarantee future results, and the portfolio should be regularly monitored and rebalanced to ensure optimal performance for the investor.

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

It is difficult to draw definitive conclusions about Bitcoin and gold as inflation and market risk hedges. However, some observations can be made: Inflation hedge: Both Bitcoin and gold have historically been considered as inflation hedges, with their prices rising during periods of high inflation. However, it's worth noting that Bitcoin has only existed since 2009, so its history as an inflation hedge is relatively short compared to gold. Market risk hedge: Gold has traditionally been seen as a safe haven asset, with its price often rising during periods of market volatility and economic uncertainty. Bitcoin, on the other hand, has a relatively short history as an asset class and its price behavior during market downturns is less clear. Correlation: The correlation between Bitcoin and gold appears to be low, which suggests that they may provide diversification benefits when held together in a portfolio.

Overall, both Bitcoin and gold have the potential to act as inflation and market risk hedges, but their effectiveness in these roles may vary depending on various factors such as the current economic environment, investor sentiment, and geopolitical events. While Bitcoin and gold can potentially serve as inflation and market risk hedges, their inclusion

in a portfolio should be carefully considered based on the investor's investment goals, risk tolerance, and diversification needs. It's also important to note that past performance is not a guarantee of future performance.

One Limitation in Task 3 was that if the dataset is included till March 31st, 2023, then the Mean Efficient Frontier does not show BTC but shows Gold. But if the data is considered till December 31st 2022 then the Mean Efficient Frontier shows BTC and Gold. This limitation was also carried over to Task 4 and Task 5 and for Task 5 since it is mentioned 'full calender year'in the question, data has been considered from the start of 2015 till the end of 2022.