import pandas as pd

import numpy as np

data\_AAPL = pd.read\_csv('AAPL\_historical\_data.csv')

data\_SPY = pd.read\_csv('SPY\_history.csv')

data\_AAPL.head()

data\_SPY.head()

aapl\_df\_clean = data\_AAPL.iloc[2:].copy()

spy\_df\_clean = data\_SPY.iloc[2:].copy()

aapl\_df\_clean

spy\_df\_clean

aapl\_df\_clean.columns = ['Date', 'Close', 'High', 'Low', 'Open', 'Volume']

spy\_df\_clean.columns = ['Date', 'Close', 'High', 'Low', 'Open', 'Volume']

aapl\_df\_clean['Date'] = pd.to\_datetime(aapl\_df\_clean['Date'])

aapl\_df\_clean['Close'] = aapl\_df\_clean['Close'].astype(float)

spy\_df\_clean['Date'] = pd.to\_datetime(spy\_df\_clean['Date'])

spy\_df\_clean['Close'] = spy\_df\_clean['Close'].astype(float)

aapl\_df\_clean

spy\_df\_clean

merged\_df = pd.merge(aapl\_df\_clean[['Date', 'Close']], spy\_df\_clean[['Date', 'Close']], on='Date', suffixes=('\_AAPL', '\_SPY'))

merged\_df

# Normalize the closing prices (start each at 1.0 for fair comparison)

normalized\_prices = merged\_df[['Close\_AAPL', 'Close\_SPY']] / merged\_df[['Close\_AAPL', 'Close\_SPY']].iloc[0]

daily\_returns = normalized\_prices.pct\_change().dropna()

daily\_returns

# Define Sharpe Ratio function (negative for minimization)

def negative\_sharpe\_ratio(weights, returns):

portfolio\_returns = (returns \* weights).sum(axis=1)

mean\_return = portfolio\_returns.mean()

std\_return = portfolio\_returns.std()

sharpe\_ratio = mean\_return / std\_return

return -sharpe\_ratio # Negative for minimization

constraints = ({'type': 'eq', 'fun': lambda x: np.sum(x) - 1})

bounds = ((0, 1), (0, 1))

initial\_guess = [0.5, 0.5]

# Optimize using scipy

from scipy.optimize import minimize

result = minimize(negative\_sharpe\_ratio,

initial\_guess,

args=(daily\_returns,),

method='SLSQP',

bounds=bounds,

constraints=constraints)

optimal\_allocations = result.x

optimal\_portfolio\_returns = (daily\_returns \* optimal\_allocations).sum(axis=1)

cumulative\_return = (normalized\_prices \* optimal\_allocations).sum(axis=1).iloc[-1] - 1

avg\_daily\_return = optimal\_portfolio\_returns.mean()

std\_daily\_return = optimal\_portfolio\_returns.std()

sharpe\_ratio = avg\_daily\_return / std\_daily\_return

performance\_summary = pd.DataFrame({

'Metric': ['AAPL Allocation', 'SPY Allocation', 'Cumulative Return', 'Average Daily Return',

'Standard Deviation of Returns', 'Sharpe Ratio'],

'Value': [optimal\_allocations[0], optimal\_allocations[1], cumulative\_return,

avg\_daily\_return, std\_daily\_return, sharpe\_ratio]

})

performance\_summary

import matplotlib.pyplot as plt

# Calculate portfolio value over time using optimal allocations

portfolio\_value = (normalized\_prices \* optimal\_allocations).sum(axis=1)

spy\_value = normalized\_prices['Close\_SPY']

# Plot the optimized portfolio value vs. Benchmark SPY

plt.figure(figsize=(10, 6))

plt.plot(merged\_df['Date'], portfolio\_value, label='Optimized Portfolio')

plt.plot(merged\_df['Date'], spy\_value, label='SPY (Benchmark)')

plt.title('Optimized Portfolio Value over Time vs. Benchmark SPY')

plt.xlabel('Date')

plt.ylabel('Normalized Value')

plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.savefig('Portfolio Optimization Graph.png', bbox\_inches= 'tight')

plt.show()