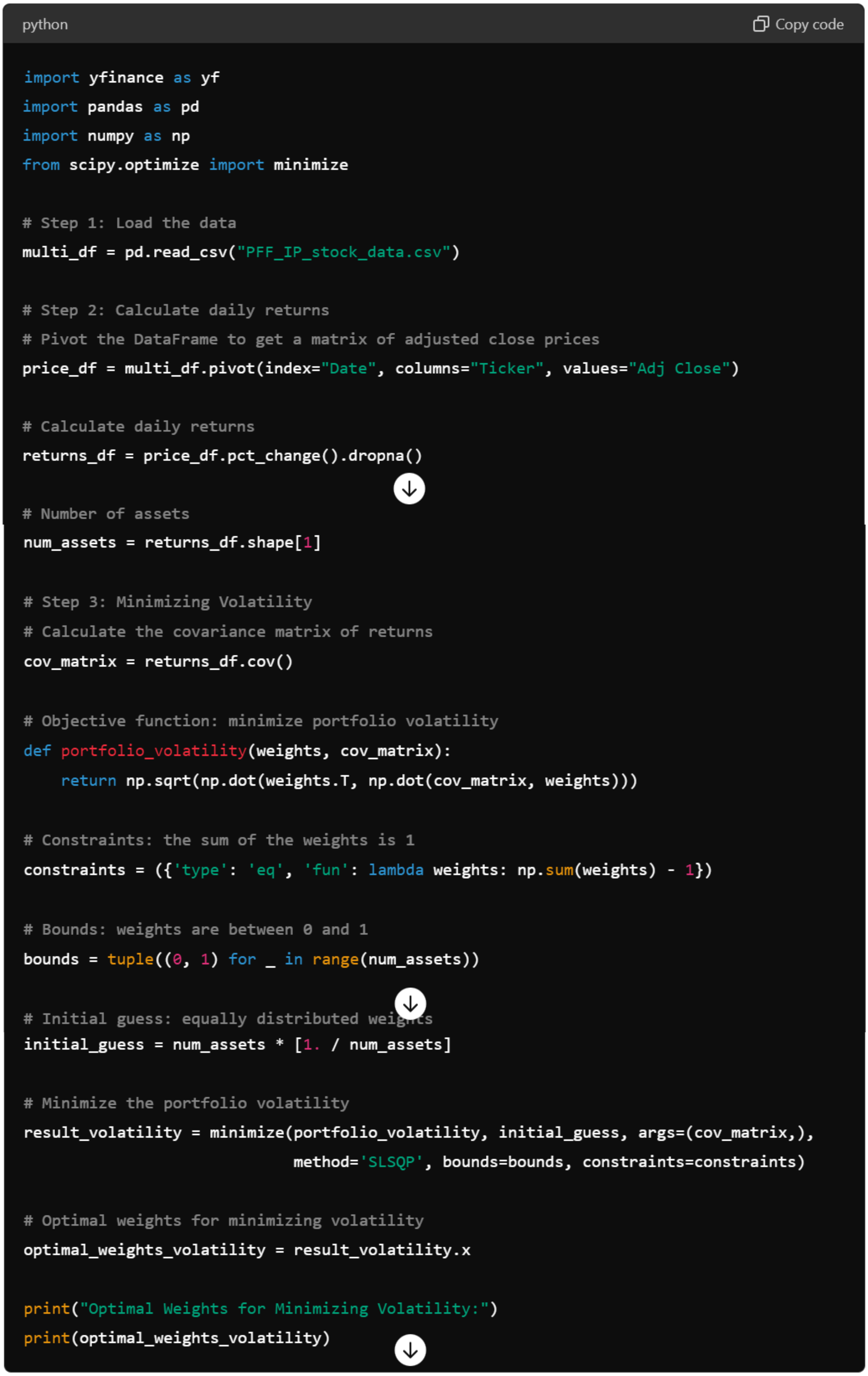
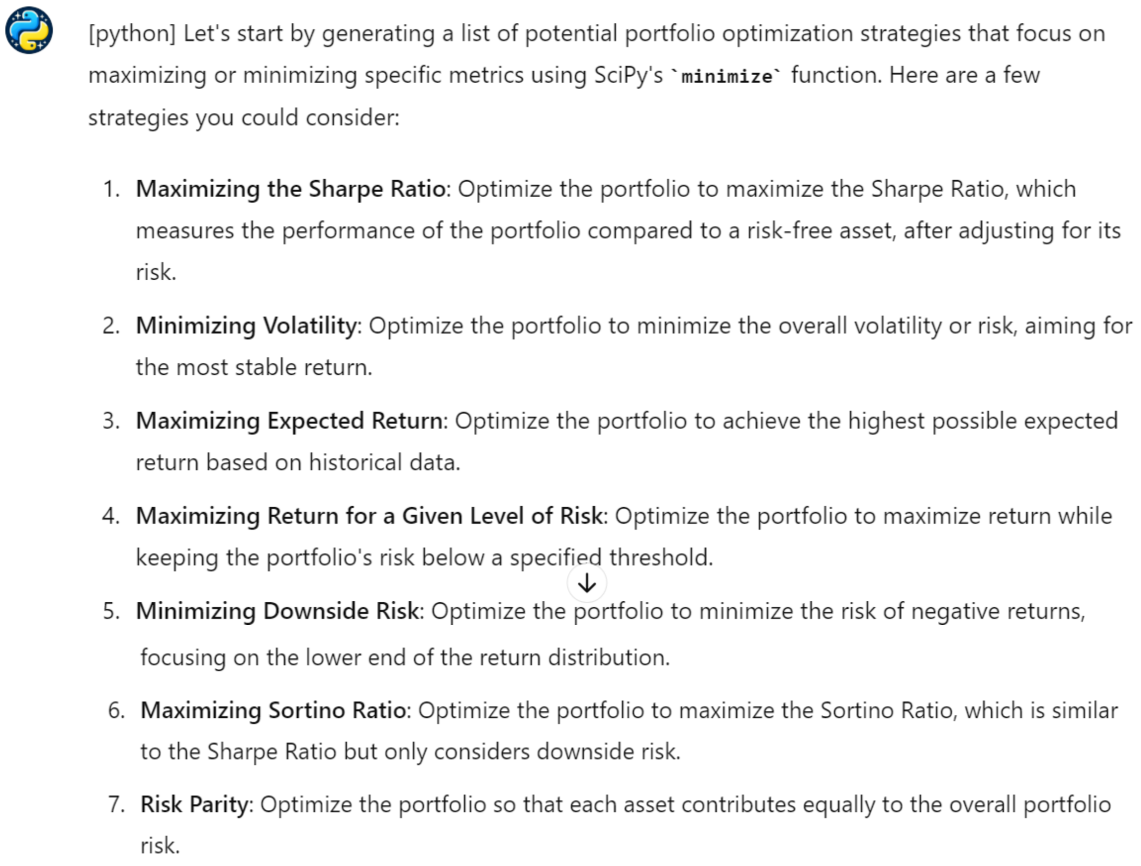
Chat GPT-4o Output for Minimizing Volatility strategy



```python
import yfinance as yf
import pandas as pd
import numpy as np
from scipy.optimize import minimize

# Step 1: Load the data
multi_df = pd.read_csv("PFF_IP_stock_data.csv")

# Step 2: Calculate daily returns
# Pivot the DataFrame to get a matrix of adjusted close prices
price_df = multi_df.pivot(index="Date", columns="Ticker", values="Adj Close")

# Calculate daily returns
returns_df = price_df.pct_change().dropna()

# Number of assets
num_assets = returns_df.shape[1]

# Step 3: Minimizing Volatility
# Calculate the covariance matrix of returns
cov_matrix = returns_df.cov()

# Objective function: minimize portfolio volatility
def portfolio_volatility(weights, cov_matrix):
    return np.sqrt(np.dot(weights.T, np.dot(cov_matrix, weights)))

# Constraints: the sum of the weights is 1
constraints = ({'type': 'eq', 'fun': lambda weights: np.sum(weights) - 1})

# Bounds: weights are between 0 and 1
bounds = tuple((0, 1) for _ in range(num_assets))

# Initial guess: equally distributed weights
initial_guess = num_assets * [1. / num_assets]

# Minimize the portfolio volatility
result_volatility = minimize(portfolio_volatility, initial_guess, args=(cov_matrix,),
                             method='SLSQP', bounds=bounds, constraints=constraints)

# Optimal weights for minimizing volatility
optimal_weights_volatility = result_volatility.x

print("Optimal Weights for Minimizing Volatility:")
print(optimal_weights_volatility)
```

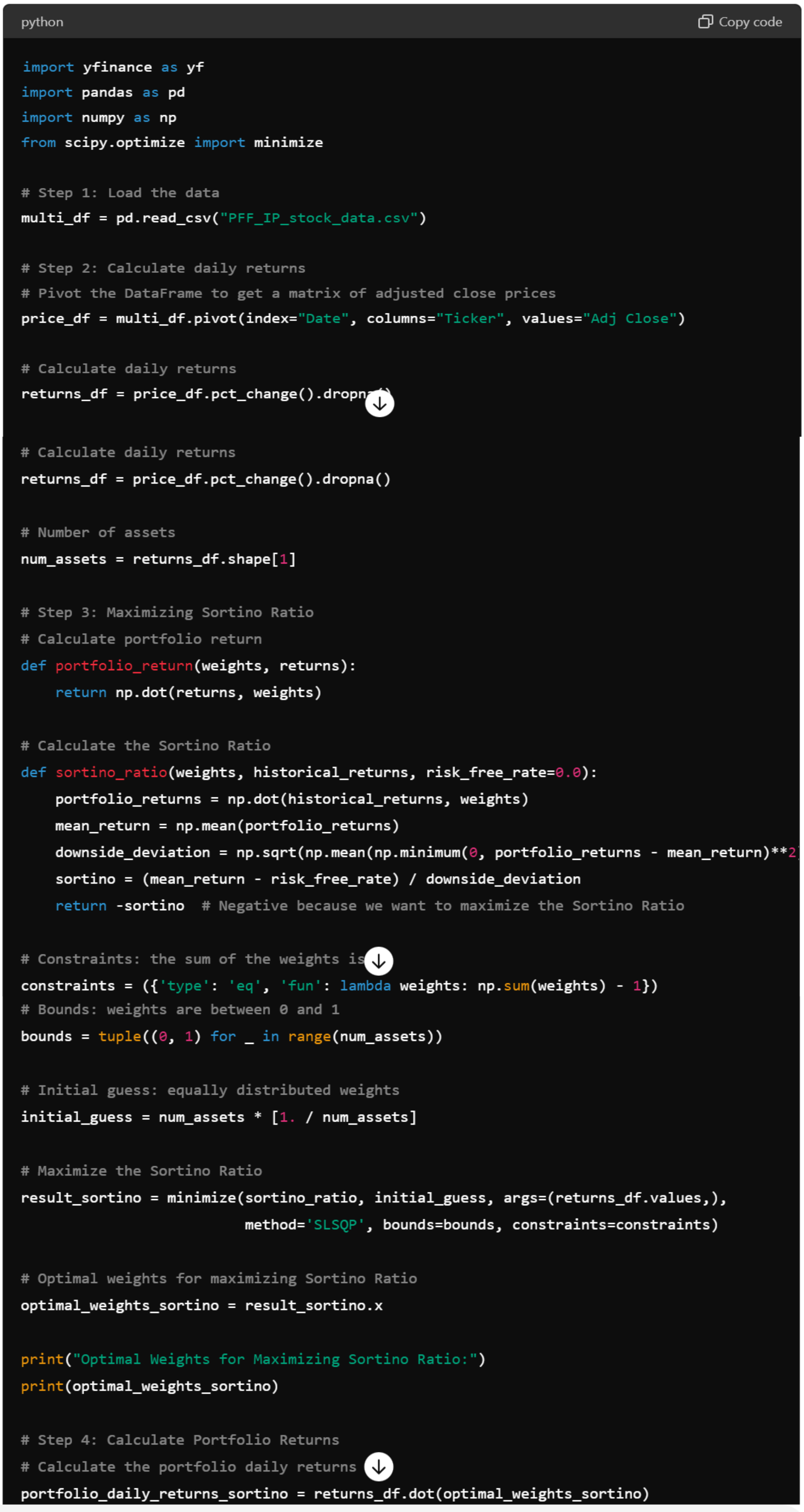
Chat GPT-4o Output in response to A2 Optimisation Brief Contents



[python] Let's start by generating a list of potential portfolio optimization strategies that focus on maximizing or minimizing specific metrics using SciPy's `minimize` function. Here are a few strategies you could consider:

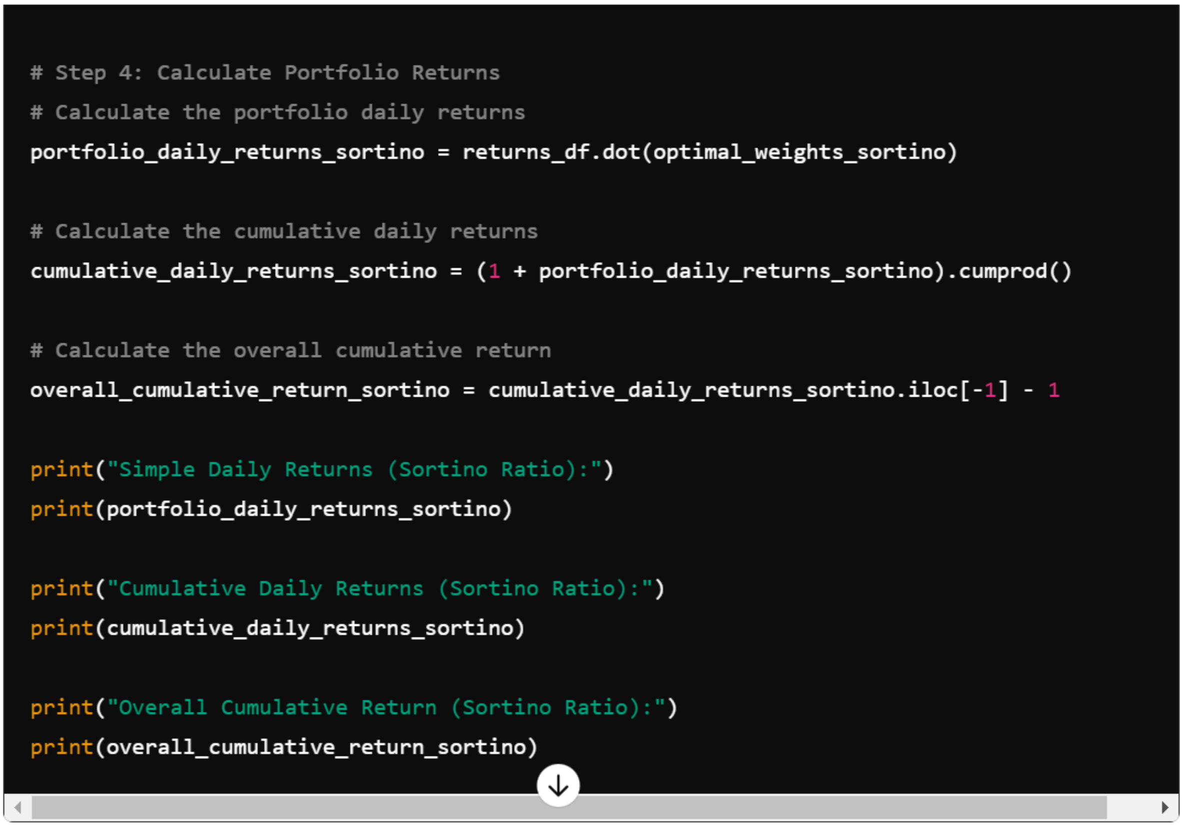
1. **Maximizing the Sharpe Ratio**: Optimize the portfolio to maximize the Sharpe Ratio, which measures the performance of the portfolio compared to a risk-free asset, after adjusting for its risk.
2. **Minimizing Volatility**: Optimize the portfolio to minimize the overall volatility or risk, aiming for the most stable return.
3. **Maximizing Expected Return**: Optimize the portfolio to achieve the highest possible expected return based on historical data.
4. **Maximizing Return for a Given Level of Risk**: Optimize the portfolio to maximize return while keeping the portfolio's risk below a specified threshold.
5. **Minimizing Downside Risk**: Optimize the portfolio to minimize the risk of negative returns, focusing on the lower end of the return distribution.
6. **Maximizing Sortino Ratio**: Optimize the portfolio to maximize the Sortino Ratio, which is similar to the Sharpe Ratio but only considers downside risk.
7. **Risk Parity**: Optimize the portfolio so that each asset contributes equally to the overall portfolio risk.

Chat GPT-4o Output for Maximizing Sortino Ratio Pt 1 of 2



```python
import yfinance as yf
import pandas as pd
import numpy as np
from scipy.optimize import minimize

# Step 1: Load the data
multi_df = pd.read_csv("PFF_IP_stock_data.csv")

# Step 2: Calculate daily returns
# Pivot the DataFrame to get a matrix of adjusted close prices
price_df = multi_df.pivot(index="Date", columns="Ticker", values="Adj Close")

# Calculate daily returns
returns_df = price_df.pct_change().dropna()

# Calculate daily returns
returns_df = price_df.pct_change().dropna()

# Number of assets
num_assets = returns_df.shape[1]

# Step 3: Maximizing Sortino Ratio
# Calculate portfolio return
def portfolio_return(weights, returns):
    return np.dot(returns, weights)

# Calculate the Sortino Ratio
def sortino_ratio(weights, historical_returns, risk_free_rate=0.0):
    portfolio_returns = np.dot(historical_returns, weights)
    mean_return = np.mean(portfolio_returns)
    downside_deviation = np.sqrt(np.mean(np.minimum(0, portfolio_returns - mean_return)**2
    sortino = (mean_return - risk_free_rate) / downside_deviation
    return -sortino  # Negative because we want to maximize the Sortino Ratio

# Constraints: the sum of the weights is
constraints = ({'type': 'eq', 'fun': lambda weights: np.sum(weights) - 1})
# Bounds: weights are between 0 and 1
bounds = tuple((0, 1) for _ in range(num_assets))

# Initial guess: equally distributed weights
initial_guess = num_assets * [1. / num_assets]

# Maximize the Sortino Ratio
result_sortino = minimize(sortino_ratio, initial_guess, args=(returns_df.values,),
                          method='SLSQP', bounds=bounds, constraints=constraints)

# Optimal weights for maximizing Sortino Ratio
optimal_weights_sortino = result_sortino.x

print("Optimal Weights for Maximizing Sortino Ratio:")
print(optimal_weights_sortino)

# Step 4: Calculate Portfolio Returns
# Calculate the portfolio daily returns
portfolio_daily_returns_sortino = returns_df.dot(optimal_weights_sortino)
```

Chat GPT-4o Output for Maximizing Sortino Ratio Pt 2 of 2



```
# Step 4: Calculate Portfolio Returns
# Calculate the portfolio daily returns
portfolio_daily_returns_sortino = returns_df.dot(optimal_weights_sortino)

# Calculate the cumulative daily returns
cumulative_daily_returns_sortino = (1 + portfolio_daily_returns_sortino).cumprod()

# Calculate the overall cumulative return
overall_cumulative_return_sortino = cumulative_daily_returns_sortino.iloc[-1] - 1

print("Simple Daily Returns (Sortino Ratio):")
print(portfolio_daily_returns_sortino)

print("Cumulative Daily Returns (Sortino Ratio):")
print(cumulative_daily_returns_sortino)

print("Overall Cumulative Return (Sortino Ratio):")
print(overall_cumulative_return_sortino)
```