University of St. Gallen

School of Management, Economics, Law, Social Sciences,

International Affairs and Computer Science

Group Project

Investment Analysis Tool:

Capital Allocation across a Risky and Risk-Free Asset

Programming Language: Python

Students:

Robin Weiss | robin.weiss@student.unisg.ch | 19-608-470

Nuria Wildermuth | nuria.wildermuth@student.unisg.ch | 20-608-238

Lukas Piffeteau | lukasalexis.piffeteau@student.unisg.ch | 21-621-487

Aleksander Vannebo | aleksander.vannebo@student.unisg.ch | 23-626-401

Spring Semester 2024

Skills: Programming with Advanced Computer Languages

Lecturer: Dr. Mario Silic

Submitted on May 12th, 2024

Table of Contents

1 Description of Task and Code

Our program is a Python-based investment analysis tool that helps users identify their investor profile, select stocks, analyze financial metrics, and allocate their investment across a portfolio of stocks (risky asset) and a risk-free asset to mitigate risk according to their risk preferences. The program progresses as follows:

1. Investor Type Identification

Interactive Quiz: The program starts with an interactive quiz where users answer questions to determine their degree of risk aversion. Points are awarded based on the risk tolerance reflected in their answers.

Risk Profile Calculation: Depending on the total points accumulated from the quiz, the program assigns a risk aversion coefficient to the user, categorizing them into different investor types, from extremely conservative to extremely aggressive.

2. Stock Analysis and Visualization

Stock Data Retrieval: The program allows users to enter stock ticker symbols and fetches general information and financial ratios using the yfinance API.

Data Visualization: It provides visualizations of stock prices along with their 50-day and 200-day moving averages. Additional graphs display financial ratios to aid in analysis.

3. Portfolio Optimization and Allocation Adjustment Based on Risk Aversion

Optimization Setup: User inputs multiple stock tickers to form a "Risky Asset". The program then performs portfolio optimization to maximize the Sharpe Ratio.

Optimization Execution: Utilizing constraints (e.g., no short selling, max allocation per stock), the program finds the optimal stock weights.

Risk and Return Calculation: Once optimal weights are determined, it calculates expected returns, volatility, and the Sharpe Ratio for the optimized portfolio.

Risk-Free Asset Inclusion: The program incorporates the user's risk aversion to adjust how much of their portfolio should be allocated to the risky asset versus a risk-free asset.

Final Portfolio Recommendation: Based on the calculated optimal weights and user's risk preference, it provides a final recommendation on asset allocation and prints expected portfolio return and volatility.

2 Code

```
# This program is written in Python programming language
# Disclaimer: This is no investment advice
# ChatGPT was used to correct and improve the program


# Importing necessary libraries for the program

# Standard Libraries
from datetime import datetime  # For handling date and time
from dateutil.relativedelta import relativedelta  # For manipulating dates with
respect to different time intervals

# Data Handling and Manipulation
import numpy as np  # For numerical operations
import pandas as pd  # For data manipulation and analysis

# Financial Data and Optimization
import yfinance as yf  # For downloading financial data from Yahoo Finance
from scipy.optimize import minimize  # For optimization tasks

# Visualization
import matplotlib.pyplot as plt  # For plotting graphs
import seaborn as sns  # For enhanced visualization style



##############################################
# INTRO QUESTIONS WHICH DEFINE INVESTOR TYPE #
##############################################



# Source: Bodie et al., 2014, pp. 174-175; Schmeiser, 2024, p.23

# You start with 0 points which must be defined, so the points of the answers can
be added
points = 0

# Introduction
print("Hey there, ")
print("This program will guide you through identifying your investor profile,
selecting stocks, analyzing their financial metrics, and allocating your investment
across a portfolio of stocks (risky asset) and a risk-free asset to mitigate risk
according to your risk preference.")
print("Answer the following questions to see what type of investor you are. Do so
by choosing between A, B or C. ")
print("Let's begin!")
```

```
# Question 1, giving the user three different options to choose from, depending on
the chosen answer, the points will be added
# \n for better layout
print("\n1. Just 60 days after you put money into an investment, its price falls
20%. Assuming none of the fundamentals have changed, what would you do? ")
print("\nA: Sell to avoid further worry and try something else, ")
print("\nB: Do nothing and wait for the investment to come back, ")
print("\nC: Buy more. It was a good investment before; now it's a cheap investment,
too ")
while True:
    choice = input("\nenter your choice (A/B/C): ")
    if choice.upper () == "A":
        print("you have chosen the first answer")
        points += 0
        break
    elif choice.upper () == "B":
        print("you have chosen the second answer")
        points += 1
        break
    elif choice.upper () == "C":
        print("you have chosen the third answer")
        points += 2
        break # This will break the loop
    else:
        print("Invalid choice. Please choose either A, B or C.")

# Question 2a
print("\n2. Now look at the previous question another way. Your investment fell
20%, but it's part of a portfolio being used to meet investment goals with three
different time horizons. ")
print("\n2a. What would you do if the goal were five years away? ")
print("\nA: Sell ")
print("\nB: Do nothing")
print("\nC: Buy more")
while True:
    choice = input("\nenter your choice (A/B/C): ")
    if choice.upper () == "A":
        print("you have chosen the first answer")
        points += 0
        break
    elif choice.upper () == "B":
        print("you have chosen the second answer")
        points += 1
        break
    elif choice.upper () == "C":
        print("you have chosen the third answer")
        points += 2
        break # This will break the loop
    else:
        print("Invalid choice. Please choose either A, B or C.")
```

```
# Question 2b
print("\n2b. What would you do if the goal were 15 years away? ")
print("\nA: Sell ")
print("\nB: Do nothing ")
print("\nC: Buy more")
while True:
    choice = input("\nenter your choice (A/B/C): ")
    if choice.upper () == "A":
        print("you have chosen the first answer")
        points += 0
        break
    elif choice.upper () == "B":
        print("you have chosen the second answer")
        points += 1
        break
    elif choice.upper () == "C":
        print("you have chosen the third answer")
        points += 2
        break # This will break the loop
    else:
        print("Invalid choice. Please choose either A, B or C.")

# Question 2c
print("\n2c. What would you do if the goal were 30 years away? ")
print("\nA: Sell ")
print("\nB: Do nothing ")
print("\nC: Buy more")
while True:
    choice = input("\nenter your choice (A/B/C): ")
    if choice.upper () == "A":
        print("you have chosen the first answer")
        points += 0
        break
    elif choice.upper () == "B":
        print("you have chosen the second answer")
        points += 1
        break
    elif choice.upper () == "C":
        print("you have chosen the third answer")
        points += 2
        break # This will break the loop
    else:
        print("Invalid choice. Please choose either A, B or C.")

# Question 3
print("\n3. The price of your retirement investment jumps 25% a month after you buy
it. Again, the fundamentals haven't changed. After you finish gloating, what do you
do? ")
print("\nA: Sell it and lock in your gains ")
print("\nB: Stay put and hope for more gain ")
print("\nC: Buy more; it could go higher")
```

```
while True:
    choice = input("\nenter your choice (A/B/C): ")
    if choice.upper () == "A":
        print("you have chosen the first answer")
        points += 0
        break
    elif choice.upper () == "B":
        print("you have chosen the second answer")
        points += 1
        break
    elif choice.upper () == "C":
        print("you have chosen the third answer")
        points += 2
        break # This will break the loop
    else:
        print("Invalid choice. Please choose either A, B or C.")

# Question 4
print("\n4. You're investing for retirement, which is 15 years away. Which would
you rather do? ")
print("\nA: Invest in a money-market fund or guaranteed investment contract, giving
up the possibility of major gains, but virtually assuring the safety of your
principal ")
print("\nB: Invest in a 50-50 mix of bond funds and stock funds, in hopes of
getting some growth, but also giving yourself some protection in the form of steady
income ")
print("\nC: Invest in aggressive growth mutual funds whose value will probably
fluctuate significantly during the year, but have the potential for impressive
gains over five or 10 years ")
while True:
    choice = input("\nenter your choice (A/B/C): ")
    if choice.upper () == "A":
        print("you have chosen the first answer")
        points += 0
        break
    elif choice.upper () == "B":
        print("you have chosen the second answer")
        points += 1
        break
    elif choice.upper () == "C":
        print("you have chosen the third answer")
        points += 2
        break # This will break the loop
    else:
        print("Invalid choice. Please choose either A, B or C.")

# Question 5
print("\n5. You just won a big prize! But which one? It's up to you. ")
print("\nA: $2,000 in cash ")
print("\nB: A 50% chance to win $5,000")
print("\nC: A 20% chance to win $15,000")
```

```
while True:
    choice = input("\nenter your choice (A/B/C): ")
    if choice.upper () == "A":
        print("you have chosen the first answer")
        points += 0
        break
    elif choice.upper () == "B":
        print("you have chosen the second answer")
        points += 1
        break
    elif choice.upper () == "C":
        print("you have chosen the third answer")
        points += 2
        break #This will break the loop
    else:
        print("Invalid choice. Please choose either A, B or C.")

# Question 6
print("\n6. A good investment opportunity just came along. But you have to borrow
money to get in. Would you take out a loan? ")
print("\nA: Definitely not ")
print("\nB: Perhaps ")
print("\nC: yes")
while True:
    choice = input("\nenter your choice (A/B/C): ")
    if choice.upper () == "A":
        print("you have chosen the first answer")
        points += 0
        break
    elif choice.upper () == "B":
        print("you have chosen the second answer")
        points += 1
        break
    elif choice.upper () == "C":
        print("you have chosen the third answer")
        points += 2
        break #This will break the loop
    else:
        print("Invalid choice. Please choose either A, B or C.")

# Question 7
print("\n7. Your company is selling stock to its employees. In three years,
management plans to take the company public. Until then, you won't be able to sell
your shares and you will get no dividends. But your investment could multiply as
much as 10 times when the company goes public. How much money would you invest? ")
print("\nA: None ")
print("\nB: Two months' salary ")
print("\nC: Four months' salary")
while True:
    choice = input("\nenter your choice (A/B/C): ")
    if choice.upper () == "A":
```

```
        print("you have chosen the first answer")
        points += 0
        break
    elif choice.upper () == "B":
        print("you have chosen the second answer")
        points += 1
        break
    elif choice.upper () == "C":
        print("you have chosen the third answer")
        points += 2
        break # This will break the loop
    else:
        print("Invalid choice. Please choose either A, B or C.")

# Question 8
print("\n8. Make a choice: ")
print("\nA: Profit of CHF 30'000 with a probability of 25%")
print("\nB: Profit of CHF 45'000 with a probability of 20%")
while True:
    choice = input("\nenter your choice (A/B): ")
    if choice.upper () == "A":
        print("you have chosen the first answer")
        points += 1
        break
    elif choice.upper () == "B":
        print("you have chosen the second answer")
        points += 2
        break # This will break the loop
    else:
        print("Invalid choice. Please choose either A or B.")

# Question 9
print("\n9. Make a choice: ")
print("\nA: Profit of CHF 30'000 with a probability of 100%")
print("\nB: Profit of CHF 50'000 with a probability of 80%")
while True:
    choice = input("\nenter your choice (A/B): ")
    if choice.upper () == "A":
        print("you have chosen the first answer")
        points += 1
        break
    elif choice.upper () == "B":
        print("you have chosen the second answer")
        points += 2
        break # This will break the loop
    else:
        print("Invalid choice. Please choose either A or B.")

# Introduction of the results
print("\nThank you for your answers.")
```

```
# Telling the user the amount of collected points
print("You have collected: ", points, "points")

# Depending on the amount of points, the user will be assigned a coefficient of
risk aversion A
print("Your coefficient of risk aversion is: ")
if 0 <= points <= 3:
    A = 8
    print( "A = 8, extremely conservative")
if 4 <= points <= 7:
    print("A = 7, conservative investor")
    A = 7
if 8 <= points <= 11 :
    print("A = 6, conservative to moderate investor ")
    A = 6
if 12 <= points <= 15 :
    print("A = 5, moderate to aggressive investor")
    A = 5
if 16  <= points <= 19 :
    print("A = 4, aggressive investor")
    A = 4
if 20  <= points <= 22 :
    print("A = 3, extremely aggressive investor")
    A = 3
```

```
######################################################################
# STOCK PICKER: GENERAL INFORMATION, RATIO ANALYSIS & VISUALIZATION #
######################################################################



# Initial information print for the user
print("\n\nYou are now in the module GENERAL INFORMATION, RATIO ANALYSIS &
VISUALIZATION.")
print("This module provides an overview and financial analysis of publicly traded
companies.")
print("\nThe module progresses as follows:\n1. Enter a ticker symbol to fetch
data.\n2. Review the data presented.\n3. Choose to inspect another company or
exit.")

# Function to fetch general company information
def fetch_general_info(ticker):
    """
    Fetches general information about a company using its ticker symbol.

    Args:
    ticker (str): The stock ticker symbol of the company.

    Returns:
    dict: A dictionary containing key company information.
    """
    stock = yf.Ticker(ticker)  # Create a Ticker object
    info = stock.info          # Retrieve stock information
    general_info = {
        'Name': info.get('shortName', 'N/A'),          # Company name
        'Country': info.get('country', 'N/A'),         # Country of operation
        'City': info.get('city', 'N/A'),               # City of operation
        'Industry': info.get('industry', 'N/A'),       # Industry category
        'Sector': info.get('sector', 'N/A'),           # Sector category
        'Full Time Employees': info.get('fullTimeEmployees', 'N/A')  # Number of
employees
    }
    return general_info

# Function to fetch financial ratios
def fetch_financial_ratios(ticker):
    """
    Fetches financial ratios for a company.

    Args:
    ticker (str): The stock ticker symbol of the company.

    Returns:
    dict: A dictionary containing key financial ratios.
    """
    stock = yf.Ticker(ticker)
```

```
    ratios = {
        'PE Ratio': round(stock.info.get('trailingPE', 'N/A'), 2),
        'PEG Ratio': round(stock.info.get('pegRatio', 'N/A'), 2),
        'Price to Book': round(stock.info.get('priceToBook', 'N/A'), 2)
    }
    return ratios

# Function to plot stock price and moving averages
def plot_price_and_moving_averages(ticker):
    """
    Plots the closing prices and moving averages for a given stock ticker.

    Args:
    ticker (str): The stock ticker symbol for which to download and plot data.
    """
    # Fetch historical stock price data for the specified ticker over the last 300
days
    data = yf.download(ticker, period='300d')

    # Initialize a new plot with specified dimensions
    plt.figure(figsize=(10, 5))

    # Plot the closing prices in black
    data['Close'].plot(label='Closing Price', color='black')

    # Calculate and plot the 50-day moving average in blue
    data['50_MA'] = data['Close'].rolling(window=50).mean()
    data['50_MA'].plot(label='50-Day MA', color='blue')

    # Calculate and plot the 200-day moving average in red
    data['200_MA'] = data['Close'].rolling(window=200).mean()
    data['200_MA'].plot(label='200-Day MA', color='red')

    # Set the title, legend, and axis labels
    plt.title(f"{ticker} - Price and Moving Averages")
    plt.legend()
    plt.xlabel('Date')
    plt.ylabel('Close Price in $')

    # Adjust layout to prevent overlap and then display the plot
    plt.tight_layout()
    plt.show()

# Function to plot financial ratios
def plot_financial_ratios(ratios):
    """
    Plots a bar chart of financial ratios.

    Args:
    ratios (dict): A dictionary containing financial ratio names and their values.
    """
```

```
    # Create a figure for plotting
    plt.figure(figsize=(8, 6))

    # Convert the dictionary of ratios into a DataFrame and plot as a bar chart
    pd.DataFrame(list(ratios.items()), columns=['Ratio',
'Value']).set_index('Ratio').plot.bar(color='skyblue')

    # Set the plot title, and labels
    plt.title('Financial Ratios')
    plt.ylabel('Ratio Value')

    # Rotate the x-axis labels for better readability
    plt.xticks(rotation=45)

    # Adjust the layout and display the plot
    plt.tight_layout()
    plt.show()

# Function to plot closing prices, monthly returns and rolling standard deviation
def plot_stock_data(ticker):
    """
    Fetches historical stock data for a given ticker over the past 5 years and
plots closing prices,
    monthly returns, and a rolling standard deviation of these returns.

    Args:
    ticker (str): Stock ticker symbol for which data is to be plotted.
    """
    # Fetch historical data from the past 5 years
    stock_data = yf.download(ticker, period='5y')

    # Calculate monthly returns
    monthly_prices = stock_data['Adj Close'].resample('M').ffill()
    monthly_returns = monthly_prices.pct_change().dropna()

    # Calculate rolling standard deviation of monthly returns over 12 months
    rolling_std = monthly_returns.rolling(window=12).std()

    # Create a figure with three subplots
    fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(14, 12))

    # Plot closing prices
    ax1.plot(stock_data['Adj Close'], color='blue')
    ax1.set_title(f'{ticker} Closing Prices')
    ax1.set_ylabel('Price (in $)')
    ax3.set_xlabel('Date')

    # Plot monthly returns
    ax2.plot(monthly_returns, color='green')
    ax2.set_title(f'{ticker} Monthly Returns')
    ax2.set_ylabel('Monthly Return (in %)')
```

```
    ax3.set_xlabel('Date')

    # Plot rolling standard deviation
    ax3.plot(rolling_std, color='red')
    ax3.set_title(f'{ticker} Rolling Standard Deviation of Returns (12 Months)')
    ax3.set_ylabel('Standard Deviation (in %)')
    ax3.set_xlabel('Date')

    # Improve layout and display the plot
    plt.tight_layout()
    plt.show()

# Loop indefinitely to process multiple company inquiries
while True:
    # Prompt the user to enter a ticker symbol and format it to uppercase
    ticker = input("\nWhat company do you wish to inspect? \n\nTICKER:
").strip().upper()

    # Check if the input consists only of alphabetic characters (valid ticker)
    if not ticker.isalpha():
        # Inform the user of invalid input and restart the loop
        print("Invalid input. Please enter a valid ticker symbol consisting of
letters only.")
        continue

    try:
        # Attempt to fetch general company information from an external function
        general_info = fetch_general_info(ticker)

        # Check if the company name is valid and data is available
        if 'Name' in general_info and general_info['Name'] != 'N/A':
            # Display the general company information in a tabular format
            print(f"\nGENERAL INFORMATION:\n{pd.DataFrame.from_dict(general_info,
orient='index')}")
        else:
            # If data is invalid or unavailable, raise an error
            raise ValueError("Invalid ticker symbol or no data available.")

        # Fetch financial ratios from an external function
        ratios = fetch_financial_ratios(ticker)
        # Display financial ratios in a tabular format
        print(f"\nFINANCIAL RATIOS:\n{pd.DataFrame.from_dict(ratios,
orient='index')}")

        # Plotting
        plot_price_and_moving_averages(ticker)
        plot_financial_ratios(ratios)
        plot_stock_data(ticker)

    except Exception as e:
```

```
        # Handle any exceptions that occur during fetching and display an error
message
        print(f"\nFailed to retrieve data for {ticker}. Error: {e}")

    # Prompt the user to decide whether to continue analyzing other companies or to
proceed with their selected stock portfolio
    response = input("\nWould you like to analyze another company? If so, please
write 'yes'. Have you decided on the stock portfolio which will make your Risky
Asset? If so, enter anything else to continue the program.").lower()
    if response.lower() != 'yes':
        break
```

```
###################################
# PORTFOLIO OPTIMIZATION MODULE #
###################################



# Step 1: Risk/Return optimization of the Risky Asset (portfolio of stocks)
# Source: O'Connell, R. (2023). Portfolio Optimization in Python: Boost Your
Financial Performance. YouTube

# Initial information print for the user
print("\n\nIn this module, the risk and return of the Risky Asset will be optimized
using the Sharpe Ratio.")
print("You will have to enter the tickers of the stocks that will constitute your
Risky Asset first. Each ticker will be assigned the optimal weight.")
print("In a second step, your degree of risk aversion from the first module is
incorporated by allocating a percentage of your wealth to a Risk-Free Asset.")

def is_valid_ticker(ticker):
    """
    Checks if a given ticker symbol is valid by attempting to retrieve its
information from Yahoo Finance.

    Args:
    ticker (str): The stock ticker symbol to validate.

    Returns:
    bool: True if the ticker is valid and exists in Yahoo Finance's database, False
otherwise.
    """
    try:
        # Create a Ticker object from the yfinance library
        stock = yf.Ticker(ticker)
        # Attempt to retrieve information about the ticker
        info = stock.info
        # Check if 'symbol' is in the info dictionary and matches the input ticker
        # This verifies that the fetched data corresponds to the requested ticker
        return "symbol" in info and info["symbol"] == ticker
    except Exception:
        # If an error occurs during fetching or processing, assume the ticker is
invalid
        return False

# Define function to get the input from user and control for user input
def get_valid_tickers():
    """
    Continuously prompts the user to enter a list of stock tickers until a valid
list of at least three tickers is provided.

    Returns:
```

```
    list of str: A list of validated ticker symbols that are recognized by Yahoo
Finance.
    """
    while True:
        # Prompt the user to input multiple stock tickers separated by commas
        tickers_input = input("\nPlease determine the composition of the Risky
Asset by entering your final multiple tickers (stocks) separated by commas. (e.g.,
AFX.DE, NESN.SW, LIN, MDLZ): ").strip()

        # Check if input is not empty and contains commas, indicating multiple
tickers
        if tickers_input and "," in tickers_input:
            # Convert the string of tickers into a list, removing spaces and
converting to upper case
            tickers = [ticker.strip().upper() for ticker in
tickers_input.split(",")]

            # Ensure that the list contains at least three tickers
            if len(tickers) < 3:
                print("Please enter at least three tickers.")
                continue

            # Validate each ticker using the is_valid_ticker function; proceed only
if all are valid
            if all(is_valid_ticker(ticker) for ticker in tickers):
                return tickers  # Return the list of validated tickers
            else:
                # Inform the user if one or more tickers are invalid
                print("One or more tickers are invalid. Please provide valid
tickers separated by commas.")
        else:
            # Prompt the user to correctly format their input if it does not meet
the expected criteria
            print("Invalid input. Please enter at least three tickers separated by
commas.")

# Get the user input
tickers = get_valid_tickers()

# Store stocks in new variable
stocks = tickers
end_date = datetime.today()  # Set the end date for the data to today
start_date = end_date - relativedelta(years=10)  # Set the start date for the data
to 10 years ago from today

# Initialize an empty DataFrame to store the adjusted close prices of the stocks
adj_close_df = pd.DataFrame()
# Loop through each stock symbol to download its data
for stock in stocks:
    try:
        # Download stock data from Yahoo Finance from start_date to end_date
```

```
        data = yf.download(stock, start=start_date, end=end_date)
        # Extract the 'Adjusted Close' prices and add them to the DataFrame
        adj_close_df[stock] = data["Adj Close"]
    except Exception as e:
        # If there's an error during download, print the error
        print(f"Error downloading {stock}: {e}")

# Input loop to capture and validate the user-provided risk-free rate
while True:
    try:
        # Request input from the user for the risk-free rate in decimal form
        risk_free_rate = float(input("\n\nPlease insert the return (risk-free rate)
of the Risk-Free Asset you would like to invest in to adjust the risk given your
degree of risk aversion (e.g., 0.04 for 4%): "))
        # Validate if the input rate is within the logical range (-1, 1) -> risk-
free rate might be negative
        if not -1 < risk_free_rate < 1:
            print("Please enter a rate between -1 and 1.")
            continue  # Restart the loop if input is not valid
        break  # Break the loop if the input is valid
    except ValueError:
        # Handle the case where input cannot be converted to float
        print("Invalid input. Please enter a valid risk-free rate in decimal form
(e.g., 0.04).")

# Calculate log returns of the stocks
log_returns = np.log(adj_close_df / adj_close_df.shift(1)).dropna()  # Divide one
day's stock price by the previous day's stock price, drop NA values

# Calculate the covariance matrix of the log returns to understand how stocks move
together
cov_matrix = log_returns.cov() * 252  # Annualize the covariance by multiplying by
the number of trading days

# Define a function to calculate the standard deviation of portfolio returns, which
is a measure of risk
def standard_deviation(weights, cov_matrix):
    """
    Calculates the portfolio standard deviation based on asset weights and the
covariance matrix of asset returns.

    Args:
    weights (np.ndarray): An array of asset weights in the portfolio.
    cov_matrix (np.ndarray): The covariance matrix of asset returns.

    Returns:
    float: The standard deviation of the portfolio, which quantifies its risk.
    """
    # Calculate the portfolio variance using matrix multiplication
    # weights.T is the transpose of the weights array
    # The operation @ denotes matrix multiplication in numpy
```

```
    return np.sqrt(weights.T @ cov_matrix @ weights)

# Define a function to calculate the expected return of the portfolio
def expected_return(weights, log_returns):
    """
    Calculates the expected annual return of a portfolio based on asset weights and
their historical log returns.

    Args:
    weights (np.ndarray): An array representing the percentage of the total
portfolio allocated to each asset.
    log_returns (pd.DataFrame): A DataFrame containing log returns of the assets.

    Returns:
    float: The expected annual return of the portfolio.
    """
    # Calculate the mean log return for each asset, multiply by the corresponding
weights,
    # and sum the total to get the weighted average log return of the portfolio.
    # Multiply by 252 to annualize the return, assuming 252 trading days in a year.
    return np.sum(log_returns.mean() * weights) * 252

# Define a function to calculate the Sharpe ratio, which measures the performance
of an investment compared to a risk-free asset
def sharpe_ratio(weights, log_returns, cov_matrix, risk_free_rate):
    """
    Calculates the Sharpe Ratio for a portfolio, which is a measure of risk-
adjusted return.

    Args:
    weights (np.ndarray): An array representing the percentage of the total
portfolio allocated to each asset.
    log_returns (pd.DataFrame): A DataFrame containing log returns of the assets.
    cov_matrix (np.ndarray): The covariance matrix of asset returns.
    risk_free_rate (float): The annual risk-free rate used to calculate excess
returns.

    Returns:
    float: The Sharpe Ratio of the portfolio.
    """
    # Compute the Sharpe Ratio using the formula:
    # (Portfolio Return - Risk-Free Rate) / Portfolio Standard Deviation
    # This measures how much excess return is received for the extra volatility of
holding a riskier asset.
    return (expected_return(weights, log_returns) - risk_free_rate) /
standard_deviation(weights, cov_matrix)

# Define the objective function to be minimized (negative Sharpe Ratio)
def neg_sharpe_ratio(weights, log_returns, cov_matrix, risk_free_rate):
    """
```

```
    Calculates the negative Sharpe Ratio of a portfolio. The goal is to maximize
the function, by minimizing the negative Sharpe Ratio.

    Args:
    weights (np.ndarray): An array representing the percentage of the total
portfolio allocated to each asset.
    log_returns (pd.DataFrame): A DataFrame containing log returns of the assets.
    cov_matrix (np.ndarray): The covariance matrix of asset returns.
    risk_free_rate (float): The annual risk-free rate used to calculate excess
returns.

    Returns:
    float: The negative value of the Sharpe Ratio of the portfolio.
    """
    # Calculate the Sharpe Ratio using the predefined sharpe_ratio function
    return -sharpe_ratio(weights, log_returns, cov_matrix, risk_free_rate)

# Constraints to ensure that the sum of portfolio weights is 1
constraints = {"type": "eq", "fun": lambda weights: np.sum(weights) - 1}
# Set bounds for the weights of each stock in the portfolio (no short selling, max
40% allocation to any stock)
bounds = [(0, 0.4) for _ in range(len(stocks))]
# Initial guess for the weights
initial_weights = np.array([1/len(stocks)] * len(stocks))

# Perform the optimization to maximize the Sharpe Ratio (minimize the negative
Sharpe Ratio)
optimized_results = minimize(neg_sharpe_ratio, initial_weights, args=(log_returns,
cov_matrix, risk_free_rate),
                             method="SLSQP", constraints=constraints,
bounds=bounds)

# Retrieve the optimal weights from the optimization results
optimal_weights = optimized_results.x

# Porfolio Performance Metrics Calculation
optimal_portfolio_return = expected_return(optimal_weights, log_returns)  #
Calculate the expected return of the portfolio
optimal_portfolio_volatility = standard_deviation(optimal_weights, cov_matrix)  #
Compute the portfolio's volatility (standard deviation)
optimal_sharpe_ratio = sharpe_ratio(optimal_weights, log_returns, cov_matrix,
risk_free_rate)  # Determine the Sharpe Ratio of the portfolio

# Check if the optimization was successful, show performance metrics and weigths
allocated to each stock
if optimized_results.success:
    print("\n\nThe following weights were allocated to each stock (in decimals):")
    for stock, weight in zip(stocks, optimal_weights):
        print(f"{stock}: {weight: .4f}")

    # Visualize the optimal portfolio weights for each stock
```

```
    plt.figure(figsize=(10, 6))
    plt.bar(stocks, optimal_weights)  # Create a bar chart
    plt.xlabel("Stocks")  # Label for the x-axis
    plt.ylabel("Optimal Weights (in %)")  # Label for the y-axis
    plt.title("Optimal Portfolio Weights")  # Title of the chart
    plt.show()  # Display the chart

    # Display key performance metrics of the Risky Asset
    print("\n\nRisky Asset:")
    print(f"The Expected Annual Return of the Risky Asset is:
{optimal_portfolio_return:.4f}")
    print(f"The Expected Volatility of the Risky Asset is:
{optimal_portfolio_volatility:.4f}")
    print(f"The Optimal Sharpe Ratio of the Risky Asset is:
{optimal_sharpe_ratio:.4f}")
else:
    print("\n\nOptimization did not converge.")  # Error message


# Step 2: Implement the coefficient or risk aversion to determine the optimal
allocation in the risky and risk-free asset
# Source: Bodie et al., 2014, p. 182

# Initialize variables for the expected return and standard deviation (volatility)
of the risky asset
risky_asset_return = optimal_portfolio_return
risky_asset_sd = optimal_portfolio_volatility

# Here we use the coefficient 'A' of risk aversion from the first module
# Calculate the optimal weight for investment in the risky asset based on utility
maximization
# where utility U = risk_free_rate + weight_risky_asset * (E(r) - rf) - 0.5 * A *
sd_risky_asset**2 * weight_risky_asset**2
# Derive utility function, set to zero and solve for weight_risky_asset
weight_risky_asset = ((risky_asset_return - risk_free_rate) / (A *
risky_asset_sd**2)) * 100

# Ensure that the calculated weight for the risky asset does not exceed 100%
if weight_risky_asset > 100:
    weight_risky_asset = 100  # Cap the investment at 100% of total wealth

# Calculate the remaining weight for the risk-free asset
weight_risk_free_asset = 100 - weight_risky_asset

# Calculate the return and standard deviation of the two-asset portfolio
portfolio_return = weight_risky_asset * risky_asset_return + weight_risk_free_asset
* risk_free_rate
portfolio_sd = weight_risky_asset * risky_asset_sd

# Print the optimal allocations for both the risky and risk-free assets
```

```
print(f"\n\nGiven your degree of risk aversion, the optimal percentage of your
wealth that should be allocated to the Risky Asset is: {weight_risky_asset:.2f}
%.")
print(f"Consequently, the optimal percentage of your wealth that should be
allocated to the Risk-Free Asset is: {weight_risk_free_asset:.2f} %.")

# Print portfolio return and standard deviation
print(f"\n\nYour final portfolio consisting of a Risky Asset (portfolio of stocks)
and a Risk-Free Asset will yield {portfolio_return:.2f} % in return annually and
has a volatility of {portfolio_sd:.2f} %.")

# Print Farewell message
print("\n\nThank you very much for using our program. Goodbye.")
```

3 Input Example

Hey there,
This program will guide you through identifying your investor profile, selecting stocks, analyzing their fin ancial metrics, and allocating your investment across a portfolio of stocks (risky asset) and a risk-free ass et to mitigate risk according to your risk preference.
Answer the following questions to see what type of investor you are. Do so by choosing between A, B or C.

Let's begin!

1. Just 60 days after you put money into an investment, its price falls 20%. Assuming none of the fundame ntals have changed, what would you do?

A: Sell to avoid further worry and try something else,

B: Do nothing and wait for the investment to come back,

C: Buy more. It was a good investment before; now it's a cheap investment, too

enter your choice (A/B/C): B
you have chosen the second answer

2. Now look at the previous question another way. Your investment fell 20%, but it's part of a portfolio be ing used to meet investment goals with three different time horizons.

2a. What would you do if the goal were five years away?

A: Sell

B: Do nothing

C: Buy more

enter your choice (A/B/C): B
you have chosen the second answer

2b. What would you do if the goal were 15 years away?

A: Sell

B: Do nothing

C: Buy more

enter your choice (A/B/C): C
you have chosen the third answer

2c. What would you do if the goal were 30 years away?

A: Sell

B: Do nothing

C: Buy more

enter your choice (A/B/C): C
you have chosen the third answer

3. The price of your retirement investment jumps 25% a month after you buy it. Again, the fundamentals haven't changed. After you finish gloating, what do you do?

A: Sell it and lock in your gains

B: Stay put and hope for more gain

C: Buy more; it could go higher

enter your choice (A/B/C): A
you have chosen the first answer

4. You're investing for retirement, which is 15 years away. Which would you rather do?

A: Invest in a money-market fund or guaranteed investment contract, giving up the possibility of major ga ins, but virtually assuring the safety of your principal

B: Invest in a 50-50 mix of bond funds and stock funds, in hopes of getting some growth, but also giving yo urself some protection in the form of steady income

C: Invest in aggressive growth mutual funds whose value will probably fluctuate significantly during the y ear, but have the potential for impressive gains over five or 10 years

enter your choice (A/B/C): B
you have chosen the second answer

5. You just won a big prize! But which one? It's up to you.

A: $2,000 in cash

B: A 50% chance to win $5,000

C: A 20% chance to win $15,000

enter your choice (A/B/C): A
you have chosen the first answer

6. A good investment opportunity just came along. But you have to borrow money to get in. Would you ta ke out a loan?

A: Definitely not

B: Perhaps

C: yes

enter your choice (A/B/C): B
you have chosen the second answer

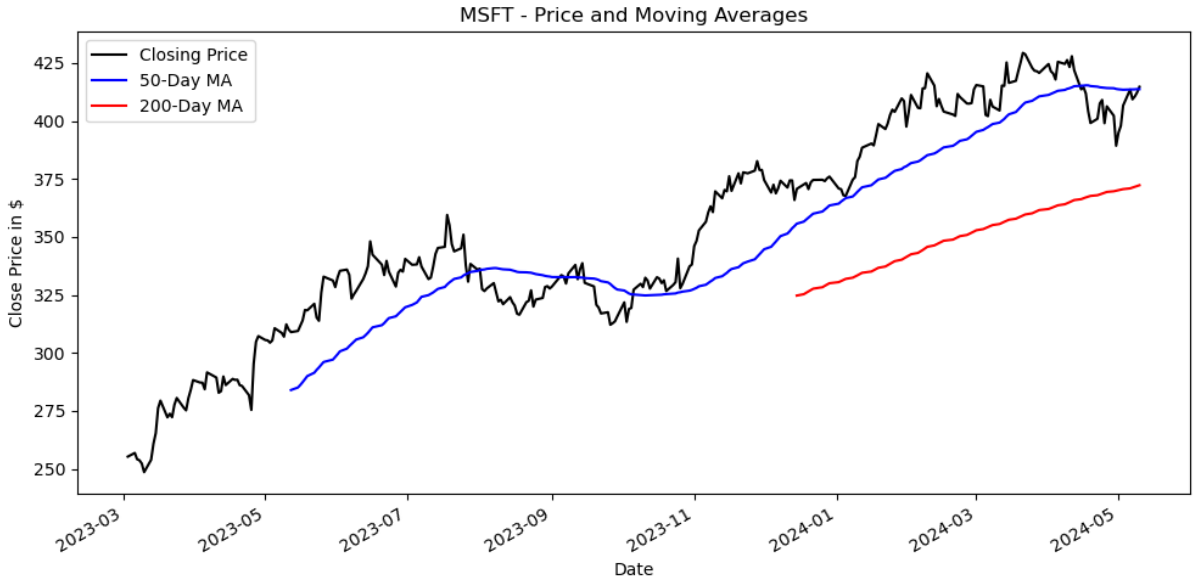
7. Your company is selling stock to its employees. In three years, management plans to take the company public. Until then, you won't be able to sell your shares and you will get no dividends. But your investmen t could multiply as much as 10 times when the company goes public. How much money would you invest?

A: None

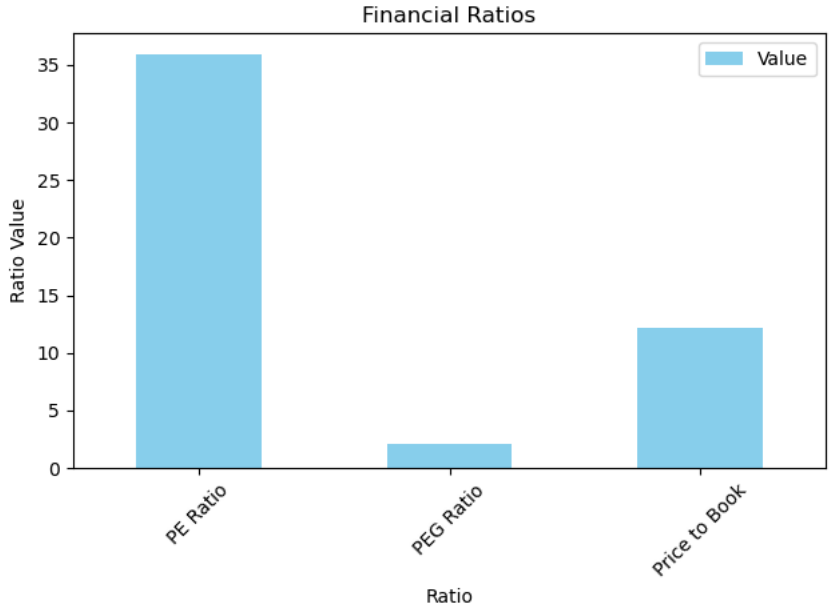
B: Two months' salary

C: Four months' salary

```
enter your choice (A/B/C):  B
you have chosen the second answer

8. Make a choice:

A: Profit of CHF 30'000 with a probability of 25%

B: Profit of CHF 45'000 with a probability of 20%

enter your choice (A/B):  B
you have chosen the second answer

9. Make a choice:

A: Profit of CHF 30'000 with a probability of 100%

B: Profit of CHF 50'000 with a probability of 80%

enter your choice (A/B):  B
you have chosen the second answer

Thank you for your answers.
You have collected:  13 points
Your coefficient of risk aversion is:
A = 5, moderate to aggressive investor


You are now in the module GENERAL INFORMATION, RATIO ANALYSIS & VISUALIZATION.
This module provides an overview and financial analysis of publicly traded companies.

The module progresses as follows:
1. Enter a ticker symbol to fetch data.
2. Review the data presented.
3. Choose to inspect another company or exit.

What company do you wish to inspect?

TICKER:  MSFT
[**********************100%%**********************]  1 of 1 completed

GENERAL INFORMATION:
                             0
Name                 Microsoft Corporation
Country                 United States
City                     Redmond
Industry          Software - Infrastructure
Sector                   Technology
Full Time Employees          221000

FINANCIAL RATIOS:
              0
PE Ratio      35.94
PEG Ratio      2.15
Price to Book  12.18
```



<Figure size 800x600 with 0 Axes>



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

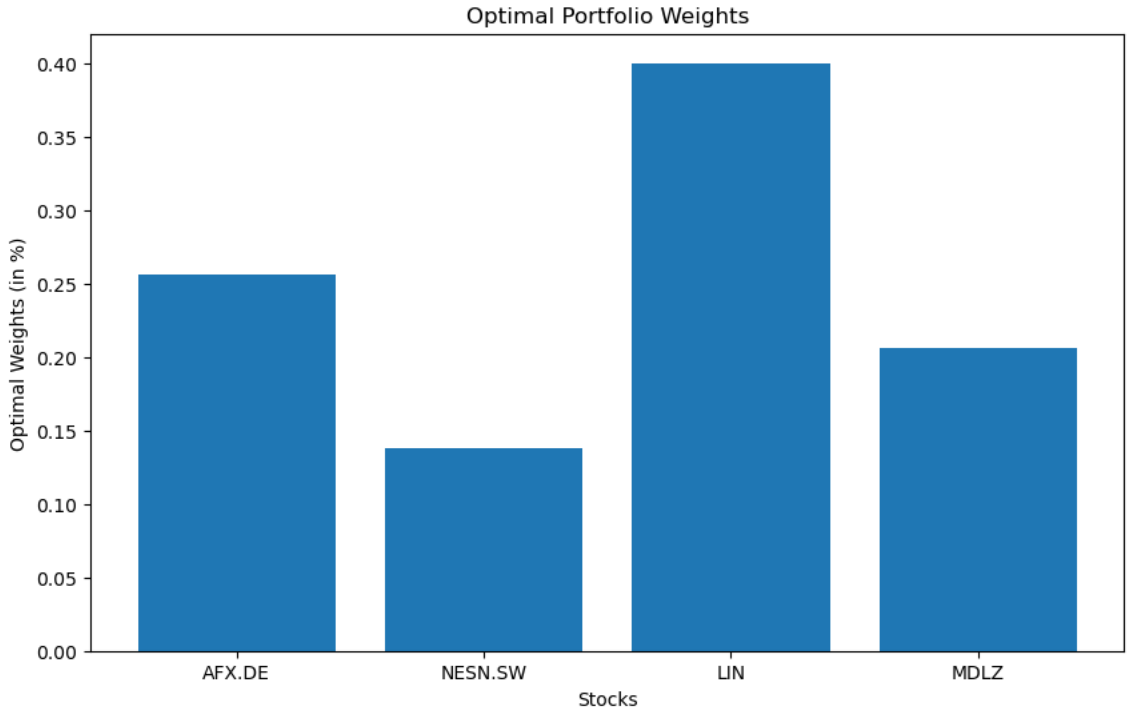
Would you like to analyze another company? If so, please write 'yes'. Have you decided on the stock portf olio which will make your Risky Asset? If so, enter anything else to continue the program. continue

In this module, the risk and return of the Risky Asset will be optimized using the Sharpe Ratio.
You will have to enter the tickers of the stocks that will constitute your Risky Asset first. Each ticker will b e assigned the optimal weight.
In a second step, your degree of risk aversion from the first module is incorporated by allocating a percen tage of your wealth to a Risk-Free Asset.

Please determine the composition of the Risky Asset by entering your final multiple tickers (stocks) separ ated by commas. (e.g., AFX.DE, NESN.SW, LIN, MDLZ): AFX.DE, NESN.SW, LIN, MDLZ
[**********100%%**********] 1 of 1 completed
[**********100%%**********] 1 of 1 completed
[**********100%%**********] 1 of 1 completed
[**********100%%**********] 1 of 1 completed

Please insert the return (risk-free rate) of the Risk-Free Asset you would like to invest in to adjust the risk given your degree of risk aversion (e.g., 0.04 for 4%): 0.04

The following weights were allocated to each stock (in decimals):
AFX.DE: 0.2565
NESN.SW: 0.1377
LIN: 0.4000
MDLZ: 0.2058



Risky Asset:
The Expected Annual Return of the Risky Asset is: 0.1312
The Expected Volatility of the Risky Asset is: 0.1714
The Optimal Sharpe Ratio of the Risky Asset is: 0.5321

Given your degree of risk aversion, the optimal percentage of your wealth that should be allocated to the Risky Asset is: 62.09 %.
Consequently, the optimal percentage of your wealth that should be allocated to the Risk-Free Asset is: 37 .91 %.

Your final portfolio consisting of a Risky Asset (portfolio of stocks) and a Risk-Free Asset will yield 9.66 % in return annually and has a volatility of 10.64 %.

Thank you very much for using our program. Goodbye.

References

Bodie, Z., Kane, A., & Marcus, A. J. (2014). Investments (Tenth global edition). McGraw-Hill Education.

O'Connell, R. (2023). Portfolio Optimization in Python: Boost Your Financial Performance. Youtube. Retrieved from https://www.youtube.com/watch?v=9GA2WlYFeBU&t=512s on April 26th, 2024

Schmeiser, H. (2024). Risk Management and Insurance, Part I [PowerPoint-Slides]. University of St. Gallen (HSG). Retrieved from https://learning.unisg.ch/courses/19117/files/2702467?module_item_id=552742 on May 1st , 2024

Declaration of Authorship

"We hereby declare

- that we have written this thesis without any help from others and without the use of documents and aids other than those stated above;
- that we have mentioned all the sources used and that we have cited them correctly according to established academic citation rules;
- that we have acquired any immaterial rights to materials we may have used such as images or graphs, or that we have produced such materials ourselves;
- that the topic or parts of it are not already the object of any work or examination of another course unless this has been explicitly agreed on with the faculty member in advance and is referred to in the thesis;
- that we will not pass on copies of this work to third parties or publish them without the University's written consent if a direct connection can be established with the University of St. Gallen or its faculty members;
- that we are aware that my work can be electronically checked for plagiarism and that we hereby grant the University of St. Gallen copyright in accordance with the Examination Regulations in so far as this is required for administrative action;
- that we are aware that the University will prosecute any infringement of this declaration of authorship and, in particular, the employment of a ghostwriter, and that any such infringement may result in disciplinary and criminal consequences which may result in the expulsion from the University or being stripped of the degree."

St. Gallen, May 12th, 2024

By submitting, we confirm through our conclusive action that we are submitting the Declaration of Authorship, that we have read and understood it, and that it is true.