Unit 5 - Financial Planning

Background

You decided to start a FinTech consultancy firm, and you want to make a difference by working on projects with high social impact in local communities. You just won your first contract to help one of the biggest credit unions in your area. They want to create a tool that helps their members enhance their financial health. The Chief Technology Officer (CTO) of the credit union asked you to develop a prototype application to demo in the next credit union assembly.

The credit union board wants to allow the union's members to assess their monthly personal finances, and also be able to forecast a reasonably good retirement plan based on cryptocurrencies, stocks, and bonds.

In this homework activity, you will use all the skills you have learned until now - focusing on using APIs as part of the technical solution - to create two financial analysis tools.

The first will be a personal finance planner that will allow users to visualize their savings composed by investments in shares and cryptocurrencies to assess if they have enough money as an emergency fund.

The second tool will be a retirement planning tool that will use the Alpaca API to fetch historical closing prices for a retirement portfolio composed of stocks and bonds, then run Monte Carlo simulations to project the portfolio performance at 30 years. You will then use the Monte Carlo data to calculate the expected portfolio returns given a specific initial investment amount.

Files

- Personal Finance Planner starter code
- MCForecastTools toolkit

Resources

This homework will utilize two APIs:

- The **Alpaca Markets API** will be used to pull historical stocks and bonds information.
- The **Alternative Free Crypto API** will be used to retrieve Bitcoin and Ethereum prices.

The documentation for these APIs can be found via the following links:

- Free Crypto API Documentation
- AlpacaDOCS

Instructions

Part 1 - Personal Finance Planner

In this section of the challenge, you will create a personal finance planner application. To develop the personal finance planner prototype, you should take into account the following assumptions:

- The average household income for each member of the credit union is $12,000.
- Every union member has a savings portfolio composed of cryptocurrencies, stocks and bonds:
  - Assume the following amount of crypto assets: `1.2` BTC and `5.3` ETH.
  - Assume the following amount of shares in stocks and bonds: `50` SPY (stocks) and `200` AGG (bonds).

Use the starter Jupyter notebook to complete the following steps.

Collect Crypto Prices Using the `requests` Library

1. Create two variables called `my_btc` and `my_eth`. Set them equal to `1.2` and `5.3`, respectively.
2. Use the `requests` library to fetch the current price in US dollars of bitcoin (`BTC`) and ethereum (`ETH`) using the **Alternative Free Crypto API** endpoints provided in the starter notebook.
3. Parse the API JSON response to select only the crypto prices and store each price in a variable.

   Hint: Be aware of the particular identifier for each cryptocurrency in the API JSON response - the bitcoin identifier is `1` whereas ethereum is `1027`.
4. Compute the portfolio value of cryptocurrencies and print the results.

Collect Investments Data Using Alpaca: `SPY` (stocks) and `AGG` (bonds)

Important: Remember to create a `.env` file in your working directory to store the values of your Alpaca API key and Alpaca secret key.

1. Create two variables named `my_agg` and `my_spy` and set them equal to `200` and `50`, respectively.
2. Set the Alpaca API key and secret key variables, then create the Alpaca API object using the `tradeapi.REST` function from the Alpaca SDK.
3. Format the current date as ISO format. You may change the date set in the starter code to the current date.

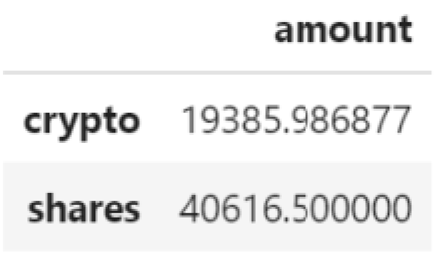
4. Get the current closing prices for `SPY` and `AGG` using Alpaca's `get_barset()` function. Transform the function's response to a Pandas DataFrame and preview the data.
5. Pick the `SPY` and `AGG` close prices from the Alpaca's `get_barset()` DataFrame response and store them as Python variables. Print the closing values for validation. Make sure to add the parameter `limit=1000` to the function call in order to receive back as much data from Alpaca as possible.
6. Compute the value in dollars of the current amount of shares and print the results.

Savings Health Analysis

In this section, you will assess the financial health of the credit union's members.

1. Create a variable called `monthly_income` and set its value to `12000`.
2. To analyze savings health, create a DataFrame called `df_savings` with two rows. Store the total value in dollars of the crypto assets in the first row and the total value of the shares in the second row.

   Hint: The `df_savings` DataFrame should have one column named `amount` and two rows where `crypto` and `shares` are the index values:



	amount
crypto	19385.986877
shares	40616.500000

3. Use the `df_savings` DataFrame to plot a pie chart to visualize the composition of personal savings.
4. Use `if` conditional statements to validate if the current savings are enough for an emergency fund. An ideal emergency fund should be equal to three times your monthly income.
   - If total savings are greater than the emergency fund, display a message congratulating the person for having enough money in this fund.
   - If total savings are equal to the emergency fund, display a message congratulating the person on reaching this financial goal.
   - If total savings are less than the emergency fund, display a message showing how many dollars away the person is from reaching the goal.

Part 2 - Retirement Planning

In this section, you will use the Alpaca API to fetch historical closing prices for a retirement portfolio and then Use the MCForecastTools toolkit to create Monte Carlo simulations to project the portfolio performance at `30` years. You will then use the Monte Carlo data to answer questions about the portfolio.

Follow the steps outlined in the starter notebook to complete the following:

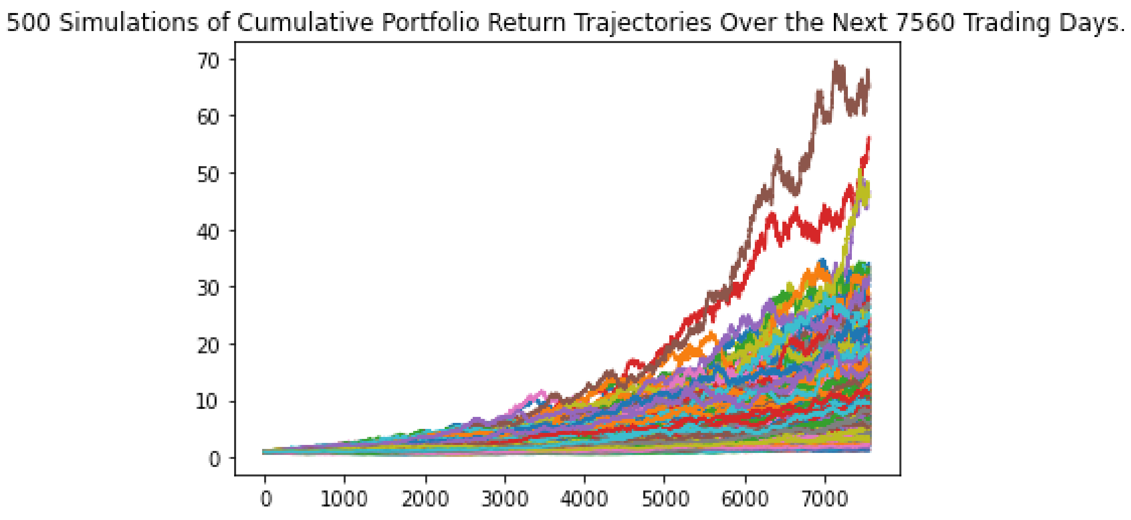
Monte Carlo Simulation

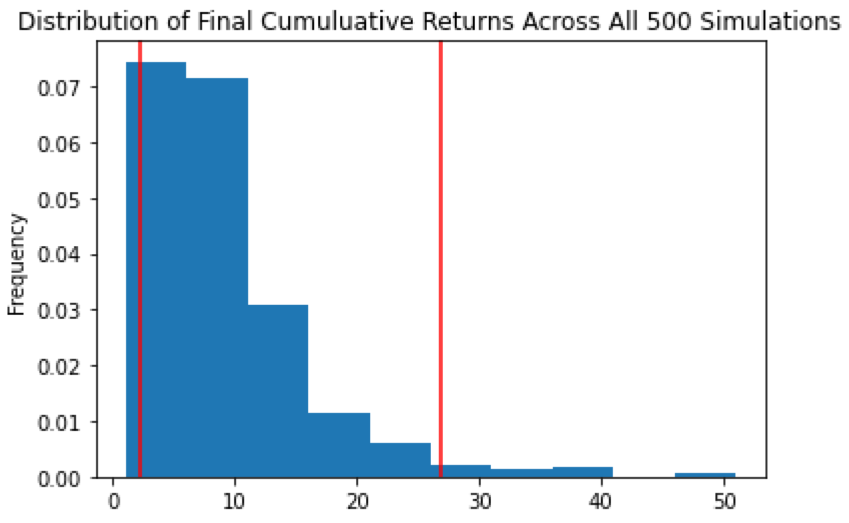
1. Use the Alpaca API to fetch five years historical closing prices for a traditional `40/60` portfolio using the `SPY` and `AGG` tickers to represent the `60%` stocks (`SPY`) and `40%` bonds (`AGG`) composition of the portfolio. Make sure to convert the API output to a DataFrame and preview the output.

   > *Note*: As before, use the parameter `limit=1000` to ensure you get the most data possible back from the API. In Monte-Carlo Simulation, getting data as far back as possible matters, because if we simulate using only small amounts of data during a recent time when markets are booming, or instead falling precipitously, a Monte-Carlo Analysis will inadvertently extrapolate this temporary market movement too far into the future. Getting data over a longer time period mitigates this effect.

2. Configure and execute a Monte Carlo Simulation of `500` runs and `30` years for the `40/60` portfolio.

3. Plot the simulation results and the probability distribution/confidence intervals.





Retirement Analysis

1. Fetch the summary statistics from the Monte Carlo simulation results.
2. Given an initial investment of `$20,000`, calculate the expected portfolio return in dollars at the `95%` lower and upper confidence intervals.
3. Calculate the expected portfolio return at the `95%` lower and upper confidence intervals based on a `50%` increase in the initial investment.

Optional Challenge - Early Retirement

The CTO of the Credit Union was really impressed with your work on this planner, but commented that `30` years seems like such a long time to wait to retire! The CTO starts wondering if the retirement plan could be adjusted to account for an earlier than normal retirement.

Try adjusting the portfolio to either include more risk (a higher stock than bond ratio) or to have a larger initial investment and rerun the retirement analysis to see what it would take to retire in `5` or `10` years instead of `30`!

Hints and Considerations

- To allow for quicker work during the Monte Carlo simulation, start out by running `100` simulations for one year of returns, and when you have the code worked out, run the full `500` simulations for `30` years.
- Remember to add the `.env` files to the `.gitignore` configuration to avoid exposing your API keys in your GitHub repository.
- A `.gitignore` file contains file names and extensions of files that you don't want pushed to your repository. For more information on how a `gitignore` works, you can read the documentation here.

Submission

1. Use the starter Jupyter Notebook for your Personal Finance Planner.
2. Submit your notebook to a new GitHub repository and create a `README.md` file.
3. Submit the link to your GitHub project to Bootcampspot for grading.

Requirements

Personal Finance Planner (35 points)

To receive all points, your code must:

- Collect crypto prices using the requests Library. (10 points)
- Collect investments data using Alpaca: SPY (stocks) and AGG (bonds). (10 points)
- Perform a savings health analysis. (15 points)

Retirement Planning (35 points)

To receive all points, your code must:

- Complete a Monte Carlo Simulation with 500 runs. (15 points)
- Plot the Monte Carlo simulation results. (5 points)
- Plot the probability distribution and confidence intervals. (5 points)
- Complete the retirement analysis. (10 points)

Optional Bonus (10 points)

To receive all bonus points, your code must:

- Optional: Adjust the portfolio to reflect an early retirement and rerun the analysis to show either a higher stock than bond ratio, or to have a higher initial investment. (10 points)

Coding Conventions and Formatting (10 points)

To receive all points, your code must:

- Place imports at the beginning of the file, just after any module comments and docstrings and before module globals and constants. (3 points)
- Name functions and variables with lowercase characters and with words separated by underscores. (2 points)
- Follow Don't Repeat Yourself (DRY) principles by creating maintainable and reusable code. (3 points)
- Use concise logic and creative engineering where possible. (2 points)

Deployment and Submission (10 points)

To receive all points, you must:

- Submit a link to a GitHub repository that's cloned to your local machine and contains your files. (5 points)
- Include appropriate commit messages in your files. (5 points)

Code Comments (10 points)

To receive all points, your code must:

- Be well commented with concise, relevant notes that other developers can understand. (10 points)
