SSMIF Coding Challenge Fall 2023

If you are receiving this document, it means that you have successfully qualified for the second round of applications for the Quant side of the Stevens Student Managed Investment Fund (SSMIF). Below, you will find information regarding the questions that will help you understand the format of the challenge, as well as transparent guidelines to ensure your submissions meet our expectations. We wish you all the best and hope to review your submissions soon!

Guidelines

The structure of the coding challenge will involve 1 mandatory question, which all applicants are required to answer. In addition to this question, applicants will be required to answer 1 other question of their choice. You will notice that in most cases, your submission for the mandatory question will relate to the other challenges on this document. The deadline for this challenge is **Thursday, October 19th, 2023 at 11:59 PM** and must be submitted through the following google form: https://forms.gle/rRyh5ewJG9Q1L5Zg6.

There are 4 optional challenges to choose from; each of these challenges relate to a specific team on the Quant side of the SSMIF. While the optional challenges are of slightly varying difficulty, the challenge you choose to complete will increase your likelihood of being considered for that specific team, among other applicants that have submitted that specific challenge. You may also complete and submit multiple of the optional challenges, which will reflect accordingly when considering your application for the appropriate teams. **Any available materials for these challenges (data, submission templates, etc.) will be provided in the zip file attached to the email.**

Where feasible, ChatGPT may be used in your code - however, **any lines of code written by ChatGPT must be cited through a 1-line comment above it**. The same applies to all other sources, online or not, including forums like Stack Overflow, and sites similar to it (the only exception is documentation relevant to a function/module used in your code). **Collaborating with other applicants or non-applicants is forbidden and will only decrease the chances of your application moving to the next round.**

Please read each section carefully to ensure that you understand all instructions and available tools pertaining to the challenge. All challenges have notes clarifying possible discrepancies. Any generally vague statements are intentionally worded to encourage creativity. However, you may forward any additional questions regarding nuances in the challenge to Head of Quant, Tejas Appana, at tappana@stevens.edu. All submissions must strictly follow the requirements listed on this google form for your submission to be considered. **Any additional notes and opportunities for bonus points are also highlighted on the submission google form.**

Mandatory Question: Portfolio Analysis

Quantitative analysts of all specializations understand that the portfolio they are working with is the core of all their work. These analysts are comfortable handling large amounts of data relevant to their portfolio, transforming that data for useful analysis, and accurately computing several metrics that demonstrate trends and results of the portfolio over time.

In this assessment, you will be given data that resembles the assets of a fake portfolio, in the form of a .csv file. There are 4 sheets within the .csv file, resembling the details of the portfolio over 4 consecutive months. Each sheet contains a dataframe of 4 columns, that looks like this:

Stock	Quantity	UnitCost	MarketPrice
AAPL	100	177.49	179.83
AMZN	80	127.96	132.21

- Column 1 contains the ticker symbol of an equity position in the fake portfolio.
- Column 2 contains the number of shares the fake portfolio owns of that equity
- Column 3 contains the average purchase price of an equity in the fake portfolio
- Column 4 contains the adjusted close per share of the equity at the end of the month

The last row of data contains information about the fake portfolio's cash balance. The price of cash is $1.00 per unit, at any given moment. A quantity of 60719 can be interpreted as the fake portfolio holding $60,719 cash in the given month.

Cash	60719	1.00	1.00

Notes for this challenge:

- You may assume that the fake portfolio does not earn dividends from any equity holdings
- Additions or removals from the fake portfolio indicate purchases and sales respectively
- Cash is not removed or added to the portfolio from external factors like deposits
- This fake portfolio was constructed with $200,000 initial capital at the end of June 2023
- Sales and purchases to the fake portfolio can happen at anytime during the month
- All UnitCost and MarketPrice data are historically accurate for each ticker
- As always code should be as efficient, organized, and documented as possible

Now that you understand the data you are dealing with, further details regarding the task will be outlined on the *first_last_portfolio_analysis.py* file which you will need to download, edit, and submit on the google form once complete.

Optional 1: Development

Development analysts bridge the gap between raw data and decision-making by leveraging full-stack web development skills. They create user-friendly applications, interactive dashboards, and efficient backend processes that streamline data interpretation and communication of investment insights. This challenge aims at testing your adeptness in full-stack web development with front-end data rendering.

Part 1: Portfolio Dashboard

Your task here is to develop an interactive dashboard to present cleaned data from the portfolio given in the mandatory question in a structured and visually appealing format. The design for the dashboard is very open-ended and is a chance for you to show creativity and web development expertise.

Required libraries and frameworks to be used:

- Front-end development: ReactJS & TailwindCSS
- Back-end development: NextJS

You are **only** allowed to use TailwindCSS for styling and Font Awesome libraries for icons. React-Charts can be used for any graphs/charts desired. You should not import any components from other libraries. Points will be deducted if any other libraries are used.

Please provide a README.md file in your repository detailing the dashboard, primary components, functionality, and instructions on how to run it.

See the NextJS documentation link provided at the end of this section for guides on creating a NextJS application. It is recommended that you use App Router rather than Pages Router.

Part 2: API (Optional)

In the first part of this problem, you created a dashboard with hard-coded data that was calculated/created in the mandatory question. For this part, although not mandatory, you are expected to create an API using Python and Flask. This API should communicate with your solution from the mandatory question, prompting it to perform calculations and data cleaning. Upon completion, the results should be sent back as a response to be rendered onto your dashboard.

More specifically, your frontend should be able to send a request for desired data to the program you made for the mandatory question. This request will run your program that cleans/calculates the original dummy data then sends back a response to the frontend that hydrates your tables and graphs with the data. *You are required to use Python and Flask as your frameworks.*

Part 3: Deployment (Optional)

Finally, deploy your application to be publicly accessible online. We recommend that you use Vercel as it works well with NextJS, but you are allowed to deploy anyway you choose. Provide the link to the website in your submission.

The following links are to helpful documentation on the libraries/frameworks:

- NextJS: https://nextjs.org/learn/foundations/about-nextjs
- ReactJS: https://react.dev/reference/react
- Tailwind CSS: https://tailwindcss.com/docs/installation
- Flask: https://flask.palletsprojects.com/en/3.0.x/#
- React Charts: https://react-charts.tanstack.com/docs/overview

Optional 2: Algorithmic Trading

Algorithmic trading analysts need a strong command of data manipulation across various time frames, the ability to construct models and signals, and the aptitude to design execution logic for automated trading strategies. In the upcoming assessment, we will evaluate your capability to research, develop, and test algorithmic trading strategies, emphasizing both innovative design and coding proficiency, to ensure you are well-equipped for this integral position within our investment team.

Part 1: Strategy Implementation

Using the data cleaned from the required question, create an algorithmic trading strategy, implemented in the 'Strategy Implementation' section where you source yfinance data for the time period defined in the uncleaned data frame for one or more of the equities defined, and create a strategy (not buy and hold) that maximizes profit and minimizes risk. Your buying power is $200,000. You will not necessarily be graded on a positive profit/loss (although having a profitable strategy will be taken into account), but rather graded on creativity and displayed coding ability. In addition to the libraries allowed for the required challenge, you may also use scikit-learn. Note that your strategy theory doesn't have to be original or novel as much of algorithmic trading is understanding prior research and working out the implementation. *Make sure your code blocks are executed prior to submitting so we are able to easily see your outputs.*

Part 2: Plotting

In the 'Plot P/L' section , plot the profit/loss of your strategy over the defined time period with markers for when trades are executed. In the 'Plot Liquidity' section, plot the liquidity ratio of the strategy itself over time. Make your graphs as detailed as possible and clean looking so that we can easily analyze the performance of your strategy overtime.

Further details regarding the task will be outlined on the *first_last_factor.ipynb* file which you will need to download, edit, and submit on the google form once complete.

Optional 3: Factor Model

Analysts who are a part of factor modeling teams have the ability to analyze the performance of the fund's different assets over time to assess how much money should be invested in each asset by determining a set of weights for how much money should go to each asset. Equally as important is backtesting these weights to assess their performance in past time periods. In this assessment, you will create a model to determine the weights for the assets in the portfolio you worked with in the mandatory challenge, and then backtest your weights against a benchmark.

Part 1: Model Creation

One of the most important aspects to model creation is getting the necessary data for the creation of your model. Using the yfinance package, you will have to collect the returns of the assets in the portfolio from the mandatory challenge. How far back in time you need to go in your data collection is up to you. You are welcome to collect as much data as you believe is necessary, but do carefully consider how far back you need to go, as economic conditions do change over time.

Once you have all of your data, you can then create your model. This model can be any model of your choice, from a machine learning model to a more standard financial model like the Capital Asset Pricing Model (CAPM). We recommend trying out multiple models to see which model performs best, but you are only *required* to create and test at least one model. For *each* model you try, explain in a comment why you decided on using that model, and if you tried multiple models you must pick one as your final model and explain why you selected it as your model of choice (ex: based off of backtesting in part 2).

Part 2: Backtesting

Once you have created your model that successfully generates the allocation weights, you will need to backtest your weights to assess their performance against a benchmark to see if the weights perform well or not. For this challenge, you will need to write code to backtest your model's output weights for the portfolio against the benchmark of the S&P 500. Your backtest is required to have the following:

- A graph of the portfolio's cumulative returns vs. the cumulative returns of the benchmark
- A data frame comparing the portfolio's metrics vs. the benchmark, with the following metrics being required:
    - Sharpe Ratio
    - Volatility
    - Cumulative Returns

Further details regarding the task will be outlined on the *first_last_factor.ipynb* file which you will need to download, edit, and submit on the google form once complete.

Optional 4: Risk Management

Risk management analysts are capable of understanding, using, and applying various statistical metrics to assess portfolio performance and strength. In this assessment, you will define, compute, and compare several risk metrics on the fake portfolio data from the mandatory question. Your task is to complete the risk.py file provided to you in the stub.

Part 1: Financial Ratio Calculation

One of the most important tasks that a risk management analyst has within a financial firm is to assess the level of risk that a given portfolio has. One effective way to do this is to use financial ratios that give a quantitative insight into the risk associated with a particular set of equities.

Implement a financial_ratio() function that calculates and returns the following metrics:

- Volatility
- Sharpe Ratio
- 95% VaR (Value at Risk)
- Maximum Drawdown
- Beta

Do not use any packages or API that directly compute the metrics/ratios for you.

Additionally, comment your code/functions explaining what these metrics are, differences between them, and how they are effective in evaluating risk.

Part 2:

Two key and unique ways that risk analysts compute returns are time weighted returns and money weighted returns.

Implement time_weighted() and money_weighted() functions that calculate these returns respectively. Create labeled plots showing the returns for both methods. Do not use any packages or API that directly compute the returns for you.

Add to your comments with a short explanation of what time weighted and money weighted returns are. Be sure to touch on:

- Methodology for each
- Pros and cons of each
- Your opinion on which is more useful.

Further details regarding the task will be outlined on the *first_last_risk.py* file which you will need to download, edit, and submit on the google form once complete.