SSMIF Quant Coding Challenge
Fall 2023 Application

Introduction

Hello and welcome to the SSMIF Quant Coding Challenge! The following document will outline the questions which make up our coding challenge this semester. Completion of this coding challenge is required for acceptance to the interview round. You are required to answer at least 2 out of the 5 questions detailed in this document. We greatly emphasize that quality is better than quantity, so please concentrate on the quality of your solutions. This challenge will be used to assess what skills and abilities you offer to SSMIF, but our primary focus is on clean, well-documented, efficient code that well accomplishes the tasks given to you. Also, please remember to read the problem descriptions **thoroughly**.

Your submission must be written in **Python** and your files for the questions must be named and formatted as shown in the following instructions to be considered. **All files that we provide to you will be in this google drive folder: https://drive.google.com/drive/.** Please use the following Google Form to submit your files: https://forms.gle/J3pCw469ezjyhcDu6. All applications should be submitted before **8:00am** on **3/09/2023**. We will not consider any late submissions.

Question 1: Puzzle

For this problem, you will be provided a .py file that you will implement to represent a Rubik's Cube. Within the file are functions you are required to complete to ensure the file works as intended for testing purposes. These are denoted by (#write your implementation below) in each of the required functions. You are also able to write helper functions if you wish to.

The Problem: You will first solve the stated functions to manipulate the matrices representing each side. Then complete the function that solves the cube. **Note: This is not solving the cube in the typical way where all sides are the same color**. The cube is solved when there is a cross on each side in the same color. We encourage you to be creative in how you solve this problem. While there are only a finite number of ways to manipulate the matrices that represent each side, there are many ways to solve a Rubik's Cube.

A sample matrix is as follows: [['R', 'W', 'Y'], ['B', 'G', 'O'], ['W', 'Y', 'R']]

Functions to complete:

- **rotate_row_right(row_num):**
  - This takes a row number and rotates that row to the right, respective rows on other faces should also be rotated
- **rotate_row_left(row_num):**
  - This takes a row number and rotates that row to the left, respective rows on other faces should also be rotated
- **rotate_col_up(col_num):**
  - This takes a column number and rotates that column upwards, respective columns on other faces should also be rotated
- **rotate_col_down(col_num):**
  - This takes a column number and rotates that column downwards, respective columns on other faces should also be rotated
- **rotate_cube_right():**
  - Rotates all faces one face to the right **eg.** front face becomes right face, right face becomes back face. **Hint:** What happens to the top and bottom faces when the cube is rotated right?
- **rotate_cube_left():**
  - Same as rotate_cube_right, but left(wow who could've known)
- **rotate_cube_up():**
  - Same as other rotate cube functions but upwards
- **rotate_cube_down():**
  - Same as other rotate cube functions but downwards
- **solve_cube():**
  - Solve the cube with whatever algorithm you would like to implement. This function will also print out the output of all faces Reminder: A solved cube has a cross of the same color on every face

Helper functions are encouraged. However, you should not alter the function headers of the defined functions above or the _init_ function

Submission Details: Edit and complete the specified functions in the given .py file, aptly named puzzle.py, then rename the file as firstname_lastname_puzzle.py

Question 2: Risk Assessment

Risk 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 two risk metrics on closed adjusted prices of a 5-year historical portfolio. You will be working with the 'risk.py' and 'historical_data.csv' files in the risk stub provided.

Part 1: Pitch a Metric
Research or create one (annualized) quantitative ratio that measures the risk of a portfolio (excluding the following metrics: Standard Deviation/Variance (volatility), Sharpe, Sortino, Calmar, Treynor, Downside/Semi Deviation, Max Drawdown, alpha/beta, VaR, CVaR, and Ulcer Index).

First, implement the method 'my_metric()' in the 'risk.py' file that computes the (annualized) quantitative ratio you chose of the sample portfolio provided in 'historical_data.csv'. Do not use any packages or API that directly computes the metric for you.

Second, write a short pitch in a separate document (pdf) and explain your metric's:

- Purpose in evaluating risk
- Statistical reasoning and/or methodology
- Pros and Cons

Part 2: Sharpe Ratio
Sometimes different metrics may have too similar a gauge on portfolio risk. To prevent computational and analytical redundancies, it may be a good idea to compare the correlation coefficient between two metrics.

In this part, implement the method 'sharpe_ratio()' in the 'risk.py' file that computes the (annualized) Sharpe Ratio of the sample portfolio provided in 'historical_data.csv'.

Part 3: Evaluate Metric Correlation
Implement the method 'compare_metrics()' in the 'risk.py' file by returning a DataFrame that tracks the metric you implemented in part (1) against the Sharpe Ratio. Explain (in your writeup pdf) how you tracked the two metrics against each other and why you chose to do it that way. (Hint: you want to track how your ratio moves when the sharpe ratio *changes*).

Finally, implement the method 'get_correlation_coefficient()' in the 'risk.py' file by returning the correlation coefficient between the two metrics. Explain (in your writeup pdf) the coefficient value's implications on the relationship between the two metrics. If similar, should you prefer one metric over the other? If not, what is causing the difference(s) between your metric and the Sharpe Ratio?

Part 4: Submission
Please submit the completed 'risk.py' and pdf write up in a zip file named 'firstname_lastname_risk_assessment'.

Other Notes & Bonus Points

- Feel free to make structural changes to the class and methods as long as the output format remains as is in the stub.
- Organized and well-commented code is essential to the group. Please ensure your code is written with clarity and that all your processes have explanations
- We encourage your code to be as expandable as possible (it should be valid for several use-cases). E.g., your methods could support daily, weekly, monthly, etc. standardization in addition to just annualized metrics.
- We also encourage fast and efficient code. The quicker and less-memory intensive your data manipulation and operations, the better.

Question 3: Trading Strategy

The Algorithmic Trading coding challenge is broken down into two portions. Part 1 will demand an understanding of technical indicators through a series of tasks which require working with dataframes and object-oriented programming. The goal of this Technical Indicator super class is to return a dataframe which consists of the price data of a stock as well as 3 technical indicators associated with it. In Part 2, you must continue to demonstrate an enhanced comprehension of object-oriented programming while creating a Backtesting subclass. In this class, you will be encouraged to produce your own trading strategy for which you will also create visuals and metrics that display the performance of your strategy.

Part 1: algo_part1_firstname_lastname.py
In this file, create a class object called `IndicatorsTA`.

- **Constructor:**
  - **Inputs**: ticker symbol, start date, and end date, all as string inputs (dates should be of the format "YYYY-MM-DD")
  - **self.stockdf** = a dataframe which stores the daily stock data for the given ticker within the date ranges (you may use any library of your choosing to download this stock data). The following is an example of how this dataframe should look:

	Open	High	Low	Close	Volume	Adj Close
2022-02-19	189.50	194.52	186.45	190.27	2780967	190.29
2022-02-20	…	…	…	…	…	…
…	…	…	…	…	…	…
2023-02-17	…	…	…	…	…	…

  - **self.algodf = self.getIndicators()**
    Add this line in your constructor once you write the self.getIndicators() method to store the output of self.GetIndicators() in self.algodf

- **Methods:** You will create 3 methods, each of which should generate a dataframe with the same index as self.stockdf and should contain the values associated with the technical indicator in the columns. Example outputs of these methods should be:

	RSI(14)
2022-02-19	56.57
2022-02-20	57.34
…..	…
2023-02-17	41.95

	LBand(20, 2)	MBand(20, 2)	HBand(20, 2)
2022-02-19	56.57	59.87	64.50
2022-02-20	57.34	61.62	65.32
…..	…	…	…
2023-02-17	41.95	45.71	47.82

Paste the following function definitions into your class and write your code inside

def self.__getRSI(window : int = 14) -> pd.DataFrame:
This private method should return the RSI of the input stock in the input date range. The window parameter should be used in the generation of the RSI data, with 14 being the default value. The output dataframe should have 1 column : “RSI(window)”, making sure that the value of “window” is in the column name, not the actual word “window”.

def self.__getBBands(period : int = 20, stdev : int = 2) -> pd.DataFrame:
This private method should return the Bollinger Bands based on self.stockdf. The period and stdev (standard deviation) parameters should be used in the generation of the Bollinger Bands. The output dataframe should have 3 columns : “LBand(period, stdev)”, “MBand(period, stdev)", “HBand(period, stdev)”, which refer to the Lower Band, Moving Average, and Higher Band respectively.

def self.__getVWAP() -> pd.DataFrame:
This private method should return the Volume Weighted Average Price of the stock in the input date range. There are no external parameters for this technical indicator. The output dataframe should have 1 column : “VWAP”.

def self.__getIndicators() -> pd.DataFrame:
This private method should call the previous 3 methods within it and return a dataframe with the Adjusted Close of the stock and all the columns associated with the 3 indicators

def self.plotIndicators() -> None:
This method should plot self.algodf similarly to how a stock quote would look when you add on the 3 technical indicators to a daily chart of the stock. You may use matplotlib or pyplot libraries for this. Neatness and presentation of the visual will be taken into account.

Part 2: algo_part2_firstname_lastname.py
In this file, create a class object called `backtest`, which should be a subclass of IndicatorTA (meaning that IndicatorsTA is a superclass). You will implement YOUR OWN trading strategy using a combination of at least 3 technical indicators, of your choice, to maximize profitability (be creative).

- **Constructor:**

- **Inputs**: quantity (number of shares: float) and position (long or short: boolean)
- **self.cumulative_df** = a dataframe to store the cumulative percent returns of both a strategy in which you buy and hold from start to end and your own strategy (using the input daterange). The dataframe should have two columns, for each strategy, and store the respective cumulative percent returns for each day in the daterange. You can use any buying power you like, but ensure your percent returns are based off of the original buying power used.

	Buy & Hold	Your Strategy
2022-02-19	0.000	0.000
2022-02-20	0.001	0.003
2022-02-21	0.002	0.007
…	…	…
2023-02-17	0.011	0.026

- **Methods:** Create a method, calculateReturns(), which both contains your strategy's logic and produces/stores the cumulative percent returns dataframe that compares your strategy against the standard buy and hold strategy.
    - **def self.calculateReturns() -> pd.Dataframe:**
      This private method should contain your *strategy's logic* and produces/store the cumulative percent returns dataframe that compares your strategy against the standard buy and hold strategy.
    - **def self.PlotReturns():** This private method should produce a plot of the resulting dataframe using either matplotlib.pyplot or plotly.express. You can also produce any other visualizations that highlight other benefits of your strategy.

Submission Details:
You will be required to submit a zip file named `algo_zip_firstname_lastname` in which you have 2 files named `algo_part1_firstname_lastname.py` and `algo_part2_firstname_lastname.py`. Example submission should look like this:
Zip File: `algo_zip_tejas_appana`
    Part1: `algo_part1_tejas_appana`
    Part2: `algo_part2_tejas_appana`
A reminder that on top of being able to compile for most input stocks, your code should be readable, formatted properly, well documented, and commented thoroughly.

Question 4: Machine Learning

For this problem, you will be provided a dataset (ml_data.csv) and you will create a model to predict the target value with the goal of minimizing the error term: Root Mean Squared Error (RMSE) in your test set and our external test set. You will use the outline provided in the Jupyter Notebook file (ml_question.ipynb). Below is a description the model you will have to create:

Part 1: Exploratory Data Analysis & Preprocessing
The dataset needs to be read into your program and processed into a form that optimizes the performance of the model. You may do this in any way you wish. However, you will be expected to defend your reasoning as to why you preprocessed your data the way you did with comments and visuals if you so choose. This section should explain your preprocessing methods.

Part 2: Exploratory Model Training
Show your process of creating models and how you worked to minimize the RMSE. This process should involve creating multiple models and outputting their error terms. It is recommended that you start with a base model, output the error terms, and then make improved models and output their error terms. Feel free to create any type of model of your choice (Linear regression, logistic regression, advanced ML models, etc., or a combination if you so choose). Packages such as TensorFlow and Keras are allowed. Explain why you chose the specific model, how the model trains, and the steps you took to optimize the model. Include necessary comments and visuals for each model.

Part 3: Final Model
The last part of the problem entails writing a class that can automatically take in csv files and generate the RMSE based on that train and test data. This is where you will consolidate your exploratory sections into a final model that generates a final error term. You can assume the train_set.csv and test_set.csv are identical in format to ml_data.csv.

Submission Details: Edit and complete the specified functions in the given .ipynb file, aptly named ml_question.py, then rename the file as firstname_lastname_ml.py

Question 5: Development

For this challenge your task is to build a simple but elegant dashboard using Dash. This challenge will test your abilities in web development and pandas through a series of tasks involving callbacks functions, data fetching and overall understanding of design/user experience.

Part One: app.py
For part one of the assignment you must include every component listed below

- Create a navbar that displays your creative name for your dashboard!
    - Navbar does not need to route anywhere just add mock routes that you think would fit in a technical dashboard
- Using Tejas's web scraper (an analyst in SSMIF) or yahoo finance create a dynamic input/output table
    - Input: build an input field table that accepts
        - Ticker name ex."MSFT"
        - Number of shares ex. 20
    - Output: Create a data table that outputs with headers
        - Ticker name
        - Number of shares
        - Total value
        - Today's change
        - One year estimate
    - It is advised to use Tejas's web scraper for this portion of the challenge as it contains all of the information needed to be outputted
    - If you have any questions about his web scraper, contact him at tappana@stevens.edu
- Using plotly create a candlestick chart with a rangeslider, in the chart be sure to include
    - Ticker information
        - Date
        - Open price
        - High price
        - Low price
        - Close price
    - Relevant chart title and x/y axis titles

For part one we are also looking at how you designed the layout of your dashboard.

- Does your application look good when resizing the window?
- Overall color scheme and user experience

Part Two (optional):
For part two of the challenge we emphasize that this is **optional** and to set yourself apart from other applicants! To really elevate your project we want to see things like

- Adding a loader that appears when the API is fetching data
- Styling of data table
- Error handling

        - What happens when non-existent tickers are inputted
        - Can letters be inputted into the number of stocks field
        - What happens when no data is inputted
- Deploying the website using platform of choice
    - AWS
    - Netlify
    - Hostinger

Submission details

For the submission attach a zip file of your project title dev_app_firstname_lastname (ex. dev_app_sarang_hadagali.zip). You should also push your code to a Github repo you create called dev_app_first_last and make sure to give us viewing permissions following this guide (and invite this github username: SSMIF-Quant). With your submission make sure to include a requirement.txt file so we can use all of your dependencies. Also make sure to add a readme or text file that explains how to run your code, if we are unable to compile your code we will not be able to grade your work. We are looking for great coding practices, including code readability, commenting and structure. Finally, if you were able to deploy your site make sure to include the link within the readMe. We look forward to seeing your submission, good luck!