SSMIF Quant Coding Challenge
Spring 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 questions. The question in Part 1 is required. Although you are only *required* to answer one question from Part 2, we expect *competitive applicants* to answer two. However, we do 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. Please use the following Google Form to submit your files: https://forms.gle/bFyZQUPbjQcesJjH9. All applications should be submitted before **11:59 pm** on **10/20/2022**. We will not consider any late submissions.

Part 1: Required Problem

General Statistics and Finance Questions (main.py)

The general problem will require you to create a python file, **main.py**, to complete 3 questions. You must first generate a pandas DataFrame containing the adjusted close stock history for the full year of 2021 for the following stocks: **Coca-Cola (KO)**, **Tesla (TSLA)**, and the **SPDR S&P 500 Trust ETF (SPY)**. Data should be collected using the Python library **pandas-datareader**. Then, limited to only using the python libraries **NumPy**, **Pandas**, and **SciPy**, you must implement the following functions in the main.py file. You will need to print your answers to the console in these functions, making sure that your code is clean and commented. The following functions need to be implemented:

1. **statistics():** Compute the following statistical tests on the two equities, using the significance level of $\alpha = 0.05$. Explain your reasoning for choosing any specific statistical test, including any assumptions you are making about the respective distributions. You should explain your result as well.
    a. Is there a statistically significant difference in the mean daily returns?
    b. Is there a statistically significant difference in their volatilities?

2. **metrics():** Calculate the following metrics on **Coca-Cola (KO)**, explaining what each value means thoroughly:
    a. Volatility
    b. 95% Value at Risk (VaR)
    c. Sharpe Ratio
    d. Downside Deviation
    e. Maximum Drawdown
3. **capm():** Calculate the Beta and Alpha for **Tesla (TSLA)**, explaining what both values mean in context.

An example output for **part (a)** of the first function, **statistics()**, is shown below:

```
a) To test for a significant difference in mean daily returns, I used a
   Student's t-test. To perform this test, I am assuming that the
   observations in each sample are: independent and identically distributed,
   normally distributed, and have the same variance.

   p-value: .0772

   result: because the p-value is greater than the significance level of
   0.05, we fail to reject the null hypothesis, meaning that there is no
   significant difference between the mean daily returns of the two equities
```

Submission: Please submit your solution for this problem as a single main.py file.

Note: 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.

Part 2: Select Questions (Choose AT LEAST 1 question to complete)

Logic Problem 1: Alien IT Support (aliens.py)

Aliens have come to our planet (Hoboken, specifically) and want to compare their financial service technology with the SSMIF's. They tried running their investment strategy program on the NYSE but it crashed immediately and only outputted a list of seemingly random binary numbers to a text file. The aliens have asked us to fix it and have given instructions on what to do.

Let's say our list of binary numbers is the following:

```
01010
00010
```

```
11001
10000
11100
00111
01111
10101
10111
10110
11110
00100
```

Part 1:
We can extract *two* important numbers from this list that will help us fix their program: the *alpha* and the *sigma* rate, which relate to the power consumption of their program. Both of these rates can be found by determining the most common bit in each place of the numbers in the list. For example, the most common bit in the leftmost place is 1 (there are seven 1s and five 0s).

The *alpha* rate is the binary number formed by the most common bits in each place in the list. In the example above, the alpha rate is `10110`. In decimal, this value is 22.

The *sigma* rate is the binary number formed by the least common bits in each place; this is essentially the opposite of the alpha rate. In the example above, the sigma rate is `01001`. In decimal, this value is 9.

When multiplying the decimal version of the alpha and sigma rates together we can get the total power consumption of the program, which in our example is 198. Please write a program that can print this number out. Please ensure the number is printed in **DECIMAL** (BASE 10) form.

Part 2:
After showing the aliens the power consumption, they tell us there was nothing wrong with the power being used. We have two more values to find to help the aliens: the *delta* rate and the *omega* rate. Both of them can be found by using the same input file of binary strings as before.

The *delta* rate can be found by determining the most common value (0 or 1) in the current bit position and keeping only numbers with that bit in that position. If 0 and 1 are equally common, keep values with a 1 in the position being considered.

The *omega* rate can be found by determining the *least common* value (0 or 1) in the current bit position and keeping only numbers with that bit in that position. If 0 and 1 are equally common, keep values with a *0* in the position being considered.

Using the same example input above, here is an example of finding the delta rate finding the omega rate is a nearly identical process):

Delta:

- Start with all 12 numbers and consider only the first bit (the left-most bit) of each number. As we know from before, there are more 1 bits than 0 bits in this place, so we only keep the numbers that have a 1 in this place: 11110, 10110, 10111, 10101, 11100, 10000, and 11001
- Using the remaining numbers above, we inspect the bits in the second place and find that there are more 0 bits than 1 bits. Therefore, we only keep the following numbers: 10110, 10111, 10101, 10000
- In the third place of the remaining numbers, a 1 is the most common bit. So we keep the numbers with a 1 in the third place: 10110, 10111, 10101.
- In the fourth place of the numbers above, two of the three numbers have a 1. So we only keep those numbers: 10110, 10111
- Now that we are in the last place, one number has a 0 and one number has a 1. According to the rules, we keep the number with 1 in this position, we keep: 10111.
- We have now found the delta rate: 10111. In decimal, this is 23.

The omega value can be found in a similar process following its instructions above. The omega value for this example would be 01010, or 10 in decimal.

Please add a function to the original program that can print out the product of the delta and omega values in decimal form. In our example above, the product is 230.

Example output for your program to follow:
```
Part One: 198
Part Two: 230
```

Submission: Please submit your solution for this problem as a single aliens.py file.

Note: 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. A short stub and input file is can be found in the logic1.zip file.

Logic Problem 2: Redistribution (redistribute.py)

Given a list of ten decimals that sum to 1, redistribute the list so that all values in the list fall within a min and max value. For example, a list = [0.45, 0.2, 0.1, 0.1, 0.05, 0.05, 0.05] with min

= 0.05 and max = 0.25, should redistribute so that 0.45 is reduced to the max of 0.25, and the extra 0.2 (because 0.45 - 0.25 = 0.2) will be redistributed **proportionately** to the rest of the list. The remaining values should receive a portion of the excess amount relative to their individual size (for example, 0.1 would receive more than 0.05).

Return the new list with newly redistributed values, and print the sum to ensure it remains negligibly close to 1.

Header:

```
def redistribute(weights : List[float], min : float = 0.02, max : float = 0.25):
        '''
                Redistribute weights to fall within [min, max]
                Returns: weights (and prints sum)
        '''
        # code here
```

Some sample test cases:

1. [0.056, 0.219, 0.322, 0.025, 0.012, 0.048, .105, .097, .065, .051]
2. [0.01195615, 0.04479124, 0.01239616, 0.07624428, 0.15352109, 0.28317437, 0.23725617, 0.02611223, 0.13620537, 0.01834294]
3. [0.36515358, 0.2742581, 0.00618948, 0.03771716, 0.01165502, 0.0048555, 0.06809406, 0.20242359, 0.01526042, 0.01439309]
4. [0.4, 0.248, 0.03, 0.1, 0.12, 0.001, 0.0, 0.02, 0.04, .161]
5. [0.5, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
6. [0.5, 0.5, 0.0, 0.0, 0.0, 0.0]
7. [0.15, 0.5, 0.1501, 0.1999, 0]
8. [0.16, 0.5, 0.14, 0.19, 0.01]
9. [0.4, 0.3, 0.2, 0.01, 0.01, 0.02, 0.01, 0.03, 0.02]

Submission: Please submit your solution for this problem as a single redistribute.py file.

Note: 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.

Machine Learning Problem (ml.ipynb)

For this problem, you will be provided a dataset (ml_data.csv) and you will create **two models** to predict the target value. Before creating either model, you must **preprocess the data**. 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. Commenting and explanations alongside your steps are highly emphasized. You may use **Google Colab** or **Jupyter Notebook** for all work related to this problem. Below is a description of the models you will have to create:

First Model: You will create a Linear Regression model. However, you are **not allowed to use any models from existing libraries** (such as scikit-learn, etc.). You may only use **NumPy**. Evaluate the performance of your model.

Second Model: You can create any advanced ML model you choose (feel free to use packages such as TensorFlow and Keras). Explain why you chose the specific model and how you selected the hyperparameters. Then, evaluate the performance of your model.

Submission: Please submit your solution for this problem as a single ml.ipynb file.

Note: 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.

Development Problem (development.zip)

For this problem, you will be provided a dataset for a Pandas dataframe and you will create a dashboard to display this data using python. You will be asked to create two specific views of this data with interactive functionality. You may use **python** and any library for this problem (flask, django, dash, etc.), however, you must list all dependencies in a document named requirements.txt.

To begin start a python file and import the data:

```
import pandas as pd
netflix =
        pd.read_csv("https://raw.githubusercontent.com/practiceprobs/datasets/main/netflix-titles/netflix-ti
        tles.csv")
```

Your goal will be to use this Netflix data to create a dashboard with the following features:

Table: Display the data in a table with the Title, Director, Country of Origin, and Year of Release on a live page. Allow users to filter this table view based on the year of release and what the movie is *listed_in*.

Pie Chart: Create a pie chart that groups together movies based on *listed_in*. Allow the user to filter based on 3 flags - country of origin, type (movie or tv show), and rating (PG 13, etc.). Additionally, when the user hovers over a ‘slice of the pie’ the program should display the first 3 titles in that slice (in order of show_id).

Submission: Please submit your solution for this problem as a zipped folder containing your .py file(s) and the requirements.txt. Please name the file development.zip.

Note: 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.