Accurately Value Risk in Real Time

Shift Energy Paradigms

Internship Screening Task

May 2022

Tasks overview

- You have 3 tasks to solve, solve as much as you can from these tasks.
- You also have 2 additional tasks to load and save the data\code:
  - Load the input data files from Azure Blob Storage
  - Save the final output & your code to Azure Blob Storage
- Note that all the tasks will be automatically graded so make sure to follow each task guidelines carefully
- Additional scores will be given for the following:
  - Code structure and comments
  - The execution time of your code (its time complexity)
- You will not be able to save files to blob storage after the submission date, so manage your time wisely
- You should upload a single .CSV file containing the output of all tasks and a single .py file containing all of your code for the tasks.

Tasks overview

Tasks outline:

- Task 0: Load the data from Azure Blob Storage
- Task 1: Create wells dataframe
- Task 2: Validate WellGroupCode column
- Task 3: Maximize the return on investment
- Final Task: Upload output and code to Azure Blob Storage

Task 0: Load the data from Azure Blob Storage

Task 0

- The data files you will be using for the tasks are stored on Azure Blob Storage
- The container name is **raisa-task**
- The files structure is
  - raisa-task/input/well_data.csv
  - raisa-task/input/well_ranks.json
  - raisa-task/input/well_expenses.txt
  - raisa-task/input/return_of_investment.csv
- You will need to write a code to load this data as prerequisite step for the next tasks
- Use the following Azure SAS URL to load the data
  - https://raisademo2.blob.core.windows.net/raisa-task/input?sp=r&st=2022-05-31T17:49:47Z&se=2022-06-05T21:59:59Z&sv=2020-08-04&sr=c&sig=9M8ql9nYOhEYdmAOKUyetWbCU8hoWS72UFczkShdbeY%3D
  - Note that this SAS URL has read access only to the input folder in the raisa-task container

Task 1: Create wells dataframe

Task 1

- Using the input files that you have loaded from step 0, you will need to generate a dataframe that contains the following 11 columns:

Column Name	Column Data Type
WellId	Int
WellValueA	Int
WellValueB	Int
WellValueC	Int
WellExpensesA	Int
WellExpensesB	Int
WellExpensesC	Int
WellRankA	Int
WellRankB	Int
WellRankC	Int
WellGroupCode	String

Task 1

Important notes:

- The unique identifier column is the WellId
- If some columns have missing values for some wells, then set these missing values to zero
- Take deep look into the data and do any required cleaning or preprocessing to implement the task

Task 2: Validate WellGroupCode column

Task 2

- The WellGroupCode is considered valid if it has unique characters.
- Also it is considered valid if you removed 1 or 2 characters and the remaining characters are unique.
- Examples:
  - abcdefg → valid because it has unique characters
  - aacdefg → valid because if you removed one a, it would have unique characters
  - aaccddf → invalid because you will need to remove more than 2 characters to have unique characters
  - aacddef → valid because if you removed 2 characters, it would have unique characters
- Write code to add new column to the dataframe called **IsValidCode** if the WellGroupCode is valid set it to 1, otherwise set it to 0
- After finishing task 1 and task 2 you should have 12 columns in your dataframe.

Task 2 Guidelines

- Your Python file should contain this function defined as follows:
  - *def check_code_validation(code: string)* → *int 0/1*
  - Function name must be '*check_code_validation*'
  - Function must take a single parameter of type string.
  - Function must return int value either 0 or 1.

- Your solution will be graded for time complexity using relatively larger string, so take time to design your solution wisely.

Please, follow these mentioned guidelines to make sure that your solution will be graded correctly on our automated grading system.

Task 3: Maximize the return on investment

Task 3: Business Problem Definition

- Raisa invests in Oil & Gas by periodically buying and selling well assets and as any investor, Raisa is very concerned with the return on investment.
- During the past 12 months, Raisa had bought and sold well assets which earned a good overall return on investment. However, the decision-making team is concerned whether the buying and selling decisions were made in the optimal time for each.
- Now, after the 12 months period is over, you have data that the decision-making team didn't have. You have actual monthly worth for each Well in the dataset for the past 12 months.
- Your task is to determine the optimal buying and selling time for each well. The Decision team will later use your output to assess their past 12 months decisions.
- *Hint*: Refer to ***this article*** to know more about Return On Investment (ROI).

Task 3: Guidelines (1)

- Task input data: *'return_of_investment.csv' file with columns ['WellId', 'monthly_worth']*
  - *monthly_worth column values are in the form of a comma separated string.*
  - *You will have to parse this column so that each value is a list of integers. So that you can feed these lists to your solution function to produce the output.*
- Output will be 2 columns added to the output dataframe produced from task (1) & (2):
  - 'OptimalBuyingMonth', 'OptimalSellingMonth'
  - Use the array indices to indicate which months are optimal for buying and selling.
    - Index 0 means the first month of the 12 months period, 1 means second month, etc..
- A well **can** be bought once and sold once during the whole 12 months period.
- A well **can not** be both bought and sold in the same month.
- If there is more than one (buying, selling) time pair that maximizes return on investment, you **must** return the first pair.
  - EX: [120, 160, 120, 200, 240, 250, 260, 210, 215, 110, 90, 130]
  - Two possible solutions (buying_month, selling_month) → (0, 6), (2, 6)
  - Accepted solution (optimal_buying_month, optimal_selling_month) → (0, 6)

Task 3: Guidelines (2)

- If there is no (buying_month, selling_month) time pair that will make a **positive** return on investment, you **must** set both 'optimal_buying_month' and 'optimal_selling_month' to -1 as a flag for the decision team that they shouldn't have bought this well at all.
  - EX: [298, 291, 242, 209, 120, 106, 84, 78, 65, 60, 34, 33]
  - No (buying_month, selling_month) pair can return a positive return of investment.
  - Accepted solution (optimal_buying_month, optimal_selling_month) → (-1, -1)

- Your solution will be graded for accuracy using the provided dataset.
- Your solution will be graded for time complexity using relatively larger inputs than the 12 months that you have, so take time to design your solution wisely.

Task 3: Guidelines (3)

- Your Python file should contain this function defined as follows:
  - *def return_of_investiment(monthly_worth: list) → tuple(int, int)*
  - Function name must be '*return_of_investiment*'
  - Function must take a single parameter of type list.
  - Function must return a tuple (optimal_buying_month: int, optimal_selling_month: int)
- After *finishing this task beside task1 and task2, you should have 14 columns on your dataframe*

Please, follow these mentioned guidelines to make sure that your solution will be graded correctly on our automated grading system.

Task 3: Additional Testcases – With Solution

- [154, 270, 183, 206, 99, 209, 109, 91, 53, 77, 37, 254] →(10, 11)
- [61, 190, 92, 261, 121, 291, 185, 192, 85, 236, 40, 176] → (0, 5)
- [133, 229, 151, 184, 194, 299, 197, 43, 261, 179, 77, 141] → (7, 8)
- [124, 245, 119, 39, 213, 230, 220, 264, 200, 198, 100, 132] → (3, 7)
- [113, 32, 212, 98, 252, 217, 34, 291, 137, 214, 147, 214] → (1, 7)
- [298, 291, 242, 209, 120, 106, 84, 78, 65, 60, 34, 33] → (-1, -1)

Final Task: Upload output & code to Azure Blob Storage

Final Task

- You should upload your output data and code to the blob storage, even if you just implemented subset of the 3 tasks
- Since the tasks are automatically graded, if you didn't upload the files correctly, you will take zero score.
- The blob container name is **raisa-task-solution**
- You will upload your files directly to the container root (don't create any folders)
- You will upload the files using this SAS URL
  - https://raisademo2.blob.core.windows.net?sp=acw&st=2022-05-31T17:54:22Z&se=2022-06-05T10:59:59Z&sv=2020-08-04&sr=c&sig=H0g%2BY%2FOnNIaYNVTsX%2FP42buWaowxIlxQDJ0xqH0gvqQ%3D
  - Note that this URL has write access only to raisa-task-solution container
- The output csv file name should be youremail.csv and the python file should be youremail.py
  - Remove any dots from your email
- Make sure that you use the same email that you have received the task into it.

Final Task

- Example for the name of the files we are expecting
  - If your email is Adel.Maher_2@gmail.com
  - CSV filename will be AdelMaher_2@gmailcom.csv
  - The python filename will be AdelMaher_2@gmailcom.py
  - This how we will see them on the blob storage
    - raisa-task-solution/AdelMaher_2@gmailcom.csv
    - raisa-task-solution/AdelMaher_2@gmailcom.py
- You should upload 2 files only one csv and one .py
  - This means that all the code should be in one file
  - And all the data should be in one file
- We will not accept any file formats other than .csv and .py
- Both the SAS URLs for the read and write will expire after the due date, so manage your time wisely

Final Task

- If you completed all the tasks, then these are the expected columns in the CSV file
  - It is okay if you didn't finish all tasks, but you must upload the .csv file and the .py file to the blob before the deadline.
  - The .csv file is expected to have 50,000 rows
- Good Luck!

Column Name	Column Data Type
WellId	Int
WellValueA	Int
WellValueB	Int
WellValueC	Int
WellExpensesA	Int
WellExpensesB	Int
WellExpensesC	Int
WellRankA	Int
WellRankB	Int
WellRankC	Int
WellGroupCode	String
IsValidCode	Int
OptimalBuyingMonth	Int
OptimalSellingMonth	Int