Assessment for Cloud Operations Team (Fall 2024)

Bombardier Aerospace

Prepared by: Sushant Sinha (sinhasushant04@gmail.com)

Submitted to: Mamaleshwar Kovi Gowri Kumar (Mamaleshwar.kg@aero.bombardier.com)

Below is quick rundown of my thought process and the steps which I considered while completing this:

1) Used python for writing the code as it has the most powerful tools needed for data handling and was the preferred language for the submission
2) Started by analysing the provided data and found the data types used for blood sugar levels was in numeric form, cancerPresent was in Boolean form and the atrophy_present was in 1/0 form (this was Boolean too).
3) With thorough analysis of the csv file, I found the possible places where we can have hiccups while handling the numeric data. These points were:
    a. Missing values: glucose level was missing for patient_id 176
    b. Not available values: glucose level was not available for patient_id 137
    c. Negative values: negative glucose level found for patient_id 252
    d. Infinite values: infinite blood sugar reading was found for patient_id 738
    e. Outliers: blood sugar for patient_id 705 was very high(10000, more than 3 times than the highest blood sugar level survived by a human)
4) To handle the missing, negative and infinite values, I chose to average the values for that entire column.
5) HOWEVER, this solution is flawed. Because of the above mentioned possibility of the outliers in our data, the average is drastically impacted and thus resulting in wrong readings. The solution was to choose IQR (Interquartile range - Wikipedia) to spot the outliers and then calculate the mean WITHOUT the outliers.

Below are the steps followed for the implementation of the solution:

1) Remove the PHI columns. I removed the first name, last name, email and address. Anyways, this data wont have any impact on our diagnosis. We can still track down the patient using his patient_id.
2) Cleaned the data stripped data by handling the infinities, negatives and missing values. A small normalization was also performed on the cancerPresent and the atrophy_present columns which technically wont have any effect on our diagnosis.
3) This was followed by the implementation of the IQR to tackle the outliers.
4) Once we have all 3 values needed for the calculation of the average glucose, we can simply sum the values and divide it by 3.
5) Now, we can begin with the diagnosis phase. Using the ranges given for the 3 stages of diabetes, I used a simple ladder if-else condition to generate the result and assigned it the new column: diabetes_diagnosis.
6) At this point, we have all the data cleaned, normalised, processed and ready for utilization. I chose to store the result in CSV format as this was the format in which we received the initial data so it will be better to store the result in the same file format.

We're done with the code creation part, but we still have to create the test file which will test the implementation, working of function and the results. I used the unittest framework and referred the documentation for its implementation(unittest documentation).

For scalability, below are the ways in which we can make our script handle TBs of data. The obvious route will be to use distributed methods of processing. These are:

1) Using Spark for distributed processing by braking the data into multiple small chunks and operating on them individually.
2) Execute these subtasks parallelly on independent core/machines
3) Use a more efficient storage technique instead of CSV. Another observation, the provided data was read intensive, and we rarely write any data back(except for the end). So, we can optimise the implementation to deal with read efficient file formats.
4) ETL!!! We can use Apache tools(Kafka, Spark etc) in conjunction with Airflow for better orchestration.
5) Since we're dealing with TBs of data, we need to use a better database system, probably distributed. Followed by many optimising techniques like indexing, sorting and hashing(using Hashmaps).

Finally, here is the list of all the assumptions that I've made while working on the problem:

1) Data Imputation: for the missing values in the glucose readings, we're replacing the values with the average of the entire column. This route was chosen because we cannot delete that patient's entire row. We need to give a result to the patient.
   Else, we could have given a message `incomplete data` in the diabetes_diagnostic column for that patient.
2) Handling outliers(for patient_id:705): considering the glucose_mg/dl_t2 value for calculating the mean will affect all the data of all the patients which have a missing glucose_mg/dl_t2(because we are using the data imputation). To tackle this, we will ignore these values(outliers) while calculating the mean.
3) cancerPresent and atrophy_Present have no effect on our diagnosis. It is just for user's reference.

Below is the link for the GitHub repository:

sushant-sinha/Bombardier-Assessment

Here you can see the step by step creation of files. In addition to this, a Readme.md file is also created for better understanding of the execution of the python script and its corresponding output.

Here is the list of resources that I used for this assessment:

1) Ways to handle missing data: Top Techniques to Handle Missing Values | DataCamp
2) Ways to handle outlier: How to Find Outliers (scribbr.com)
3) IQR: How to Find Interquartile Range (IQR) (scribbr.com)
4) Lambda expressions in python: Python Lambda Functions - GeeksforGeeks
5) Python testing frameworks: unittest documentation

Thanks Mamaleshwar, for giving me the opportunity to work on this project!