**Capstone ML Models DataPull**

**Objective:**

The objective of this report is to provide a detailed explanation of the code chunk that reads data from the Google BigQuery dataset for use in the Capstone Machine Learning models. This report will explain the specific steps taken in the code, as well as the functions and libraries used.

**Methodology:**

The code chunk uses Python programming language and the following libraries:

**pandas**: for creating and manipulating DataFrames

**numpy:** for numerical computations

**google-auth:** for authenticating the user in Google Colab

**pandas\_gbq:** for connecting to BigQuery

The code follows the following steps:

**Step 1: Importing and Authenticating**

The code first authenticates the user in Google Colab using the google-auth library. The necessary libraries are then imported, including pandas, numpy and pandas\_gbq.

**Step 2: Reading Data from Google BigQuery**

In this step, the code reads data from the Google BigQuery dataset by executing SQL queries on specific tables.

The code uses the pandas\_gbq.read\_gbq() function to execute SQL queries on the tables and create Pandas dataframes for each of them. For instance, the Demographics table is queried with the SQL command "SELECT \* FROM surveyproject-378222.Capstone\_Project.Demographics", which selects all columns from the Demographics table in the Capstone\_Project dataset. Similarly, other tables such as Accommodation, Accommodation\_feedback, Academic are also queried and transformed into pandas dataframes.

Each data frame is named according to the name of the table being queried. The parameters passed to the read\_gbq function include the SQL query, the project ID where the dataset is stored, and the dialect which is set to "standard".

The result of this step is the creation of dataframes for each of the tables in the Capstone\_Project dataset.

**Step 3 : Compute the accommodation score and academic score of each student based on the feedback they provided.**

The first part of the code creates a copy of the **Accommodation\_feedback dataframe(acf)** and creates a label map dictionary to map the **textual feedback values** to **numerical values.** It then applies the label map to the columns that need to be mapped and computes the weighted average of the scores. The weights dictionary contains the weightage given to each feedback category. The code then computes the final accommodation score and scaled accommodation score by dividing the **accommodation score by 5**. It creates **two new dataframes,** one with the ID and **accommodation score** and the other with ID and **scaled accommodation score**.

The second part of the code does a similar computation for the **Academic\_feedback dataframe.**It creates a copy of the dataframe **(af)**. and computes the weighted average of the scores for each student. It then computes the final academic score and scaled academic score by dividing the **academic score by 5.** It creates **two new dataframes**, one with the ID and **academic score** and the other with ID and **scaled academic score**.

Overall, the code computes the weighted average of the feedback scores for each student and generates the **final scores** that will be used for further analysis in the project.

**Step 4: Using two methods for encoding categorical variables (Label Encoding and One-hot Encoding)**

Convert the Demographics data into a numerical form that can be used in the analysis. The first method used is Label Encoding, which assigns a **numerical label** to each category of a categorical variable. The second method used is One-hot Encoding, which creates a **binary column** for each category of a categorical variable.

In the code, the **Demographics data** is split into three **dataframes - d2, d3, and d4**, which contain different demographic information about the students. For each dataframe, the appropriate encoding method is applied to the categorical variables in order to convert them into numerical form.

**For Label Encoding,** a label map dictionary is created for each categorical variable with the category as the key and the assigned numerical label as the value. The **LabelEncoder()** function from the scikit-learn library is used to apply the label map to the appropriate column in the dataframe.

**For One-hot Encoding,** the pandas **get\_dummies()** function is used to create binary columns for each category of the categorical variable. The resulting dataframes are then merged back together to form a single dataframe containing all the encoded demographic information.

Finally, **any unnecessary columns are dropped**, and the resulting dataframe is cast to the integer data type. The resulting dataframes, **demo1 for Label Encoding** and **demo2 for One-hot Encoding**, can now be used for further analysis in the project.

**Step 5: Creating final df1 by merging Encoded demographics data with demo1 and academic scores.**

First, it merges **demo1** with the **acc\_score** dataframe on the common **ID** column using the **pd.merge()** function. The resulting merged dataframe is then merged with **acd\_score** dataframe on the common **ID** column using the same **pd.merge()** function.

The resulting merged dataframe has four columns - **ID, Enc\_Education\_lvl, Enc\_Majors, Enc\_Yrs\_exp, Enc\_Age, Enc\_Gender, Acc\_score, and Acd\_score**. The Acc\_score and Acd\_score columns are the academic scores for the accountancy and academic subjects, respectively. Finally, it changes the datatype of all columns except **Acc\_score** and **Acd\_score** to integer using the **astype ()** method.