Group: Colin Beveridge, Ben Grudt, Grace Seiler, Hang Zhang Cao

Due Date: 2/18/2022

Heart Disease Prediction Project

ETL Report

Introduction: The group was assigned to this project by a client who wants to improve heart health in America. The client wishes to do this in two ways. The first way is by creating a dashboard for the public. This dashboard will include data visualizations indicating the most prevalent risk factors for heart disease and how these vary by demographics and other categories. It will also include a heat map indicating how different cardiovascular diseases are distributed throughout the United States. Additionally, there will be links to more articles and information on heart disease in the United States to educate the public, and links to insurance information to help insure as many Americans as possible. The second way the client wants to improve heart health in America is by creating several clinics throughout the United States. The goal of these clinics is to be a less expensive alternative to the hospital for people experiencing heart issues or chest pain. The clinics are specifically targeted at groups with no insurance who are low-income and at high risk for developing heart disease. The clinics will be equipped with several diagnostic tools, and a machine learning algorithm created by the group, that can diagnose whether the patient is likely to need more serious and immediate medical attention. The group seeks to write a report for the client to identify the areas that would most benefit from these clinics, maximizing the number of people assisted by its services.

Several data sources were needed to complete the above tasks. The first data source is the Heart Failure Prediction Dataset. It comes from Kaggle and includes 11 features used to train the Machine Learning Model to diagnose the patients. The transformations done to this data are mainly to get it clean enough to use in a sklearn machine learning algorithm. Several missing values or extreme outliers were imputed, and the data was split for training and testing, and then standardized. The second dataset used comes from the CDC and is part of the Behavioral Risk Factor Surveillance System. It provides information on heart disease and its risk factors throughout the United States broken down by several demographic groups. This data had several unnecessary columns that were removed, and several null values were dealt with.

The third dataset also comes from the CDC and focuses on behavioral risk factors in select metropolitan areas in the United States. As with the other CDC dataset, unnecessary columns were removed along with invalid data. The final data source is the United States Census Bureau American Community Survey and the table used includes information on the insurance coverage of citizens of the US broken down by several demographic groups over different states. This data required lots of transformation from its raw API format to make it into usable data to be examined. This is due to the complicated name structure of the US Census Data. Once this was done, additional standard data cleaning steps were taken as well.

Data Sources: All the data sources below were first accessed the week of 1/31/2022. The first dataset was found on Kaggle and cites several other data sources that were combined to form the final set. It is known as the Heart Failure Prediction Dataset (HFPD) and was used for the machine learning model training. The second dataset used comes from the Centers for Disease Control (CDC) and is part of the Behavioral Risk Factor Surveillance System (BRFSS). The third dataset also comes from the CDC and contains select data from the BRFSS but at a metropolitan level. The fourth and final dataset comes from the United States Census Bureau American Community Survey (US Census Bureau ACS). This dataset examines health insurance coverage by various selected characteristics in the US.

Kaggle. (2021, September 10). Heart Failure Prediction Dataset [Data provides information on 11 different risk factors and indicators of heart disease, allowing for an ML Prediction Model]. https://www.kaggle.com/fedesoriano/heart-failure-prediction

Behavioral Risk Factor Surveillance System (BRFSS) - National Cardiovascular Disease Surveillance Data | Chronic Disease and Health Promotion Data & Indicators. (2021, June 24). [Behavioral risk factors survey by state.]. Centers for Disease Control. https://chronicdata.cdc.gov/Heart-Disease-Stroke-Prevention/Behavioral-Risk-Factor-Surveillance-System-BRFSS-N/ikwk-8git

Behavioral Risk Factors: Selected Metropolitan Area Risk Trends (SMART) MMSA Prevalence Data (2011 to Present) | Chronic Disease and Health Promotion Data & Indicators. (2021, December 16). [Behavioral risk factors for select metropolitan areas.]. Centers for

Disease Control and Prevention. https://chronicdata.cdc.gov/Behavioral-Risk-Factors/Behavioral-Risk-Factors-Selected-Metropolitan-Area/j32a-sa6u

ACS. (2019). SELECTED CHARACTERISTICS OF HEALTH INSURANCE COVERAGE IN THE UNITED STATES (Version 2018) [Table of health insurance coverage broken down by many groups over all US states and counties.]. United States Census Bureau. https://data.census.gov/cedsci/table?t=Health%20Insurance&g=0100000US%2404000000&tid=ACSST1Y2018.S2701

ETL Steps: To make the process clearer and more repeatable, the extract, transform and load steps will be described together for each dataset.

Heart Failure Prediction Dataset:

Extract:

1. This data file was downloaded to a cloud Databrick machine as a CSV file from Kaggle at this link, https://www.kaggle.com/fedesoriano/heart-failure-prediction.
2. The method used with other datasets of pulling the file directly from the internet is not used here because the Kaggle file requires web interaction to access properly. It will be a stretch goal for the group to get this working, but for now, downloading to the cloud and then calling from there works fine.
3. Then, the csv was opened in Databricks using the spark read function and then converting directly to pandas. Pandas was used rather than spark because the dataset is not very large, precluding the need for spark parallelization. The following transformation and ML steps were easier to complete directly with pandas.

Transform:

1. Before anything else, drop the 'HeartDisease' column from the dataframe and make it into its own pandas series. Then split the data into a training set and a testing set using sklearn.model_selection.train_test_split. Split into 75% training data, 25% testing data.
2. The 75% will be used to create the machine learning model. This is not an automated process, so not many details are given. First, impute all 0 values in the 'Cholesterol'

column with the mean from patients of same gender and similar age. Then do the same imputation on the 'Resting BP' column. Next, apply the pandas.get_dummies() function to the training data to make all categorical variables into one hot encoding form. Finally, standardize the numeric columns ('Age', 'RestingBP, 'Cholesterol', 'MaxHR', 'Oldpeak') using sklearn.preprocessing.StandardScaler. These are the main transformations required to test different models.

Load:

1. The remaining 25% of the patient data will be used to simulate a live stream from patients actively entering clinics and receiving diagnoses.
2. The 25% is first saved to a storage blob in the group container called 'patient_data_raw.csv'
3. From here the Databrick 'Heart_PatientWalkInSimulate' calls the raw file, selects ten random patients from this file and saves them individually at random time intervals to the directory 'PatientLiveData' in a file called 'patient_data_lobby.csv'
4. Saving to a file of this name triggers the Data Pipeline to produce the message in that file, then the consumer consumes that message, and writes it to a table in the SQL database. For each run of 'Heart_PatientWalkInSimulate' this currently happens 10 times, giving 10 data pipeline triggers. But that number can be changed depending on the patient load the user would like to simulate.
5. From the SQL database, the table can be linked at any time to Power BI. The finished ML model can also be loaded into Power BI. This gives the user the ability to simulate live diagnoses for patients in Power BI.

BRFSS – National Cardiovascular Disease Surveillance Data:

Extract:

1. This data file was found at this link https://chronicdata.cdc.gov/Heart-Disease-Stroke-Prevention/Behavioral-Risk-Factor-Surveillance-System-BRFSS-N/ikwk-8git, on the CDC Data website. However, the URL for the direct download of the file is https://chronicdata.cdc.gov/api/views/ikwk-8git/rows.csv?accessType=DOWNLOAD

2. This direct download URL is used in a Databrick to access the file, removing the need for saving the file to a blob. This way, if the file is updated on the website, one simply needs to run the Databrick again to get updated data.
3. Use the python library requests and apply requests.get() to the download url. Then get the content from this request with the .content method, decode it with the .decode method, and turn it into a file object to be read with the StringIO function in the python IO library.
4. Make a csv reader object and apply it to the content file given by StringIO.
5. Loop through rows in the reader object. Make the first row into the column names. Make the rest into a list of lists that hold the data. Finally, create a spark dataframe from the data and the column names.

Transform:

1. The first transform step was to drop all unnecessary columns using the df.drop command in spark Databricks. Here is a list of all the column names that were dropped.
    a. [‘row_id’, ‘DataSource’, ‘PriorityArea1’, ‘PriorityArea2’, ‘PriorityArea3’, ‘PriorityArea4’, ‘Data_Value_Unit’, ‘Data_Value_Footnote_Symbol’, ‘Data_Value_Footnote’, ‘LowConfidenceLimit’, ‘HighConfidenceLimit’, ‘CategoryID’, ‘TopicID’, ‘IndicatorID’, ‘Data_Value_TypeID’, ‘BreakOutCategoryId’, ‘BreakOutId’, ‘LocationID’, ‘Geolocation’, ‘Data_Value_Alt’]
2. The second and last transform command was to simply filter the dataframe for only the year 2018. This was done with the df.where command:
    a. newdf = df.where(df.Year == 2018)

Load:

1. This data will then be loaded into the group’s SQL Database. This uses the standard command for writing from Databricks to SQL with the following parameters:
    a. database = ‘group2’
    b. tablename = ‘dbo.CDC_Data’

    c. user = ‘group2’
    d. password = ‘capstonedev10!!’
    e. server = ‘gen10-data-fundamentals-21-11-sql-server.database.windows.net’
2. Now the 2018 table is in an SQL Database, and the loading is complete.

Behavioral Risk Factors - Selected Metropolitan Area Risk Trends:

Extract:

1. This data file was found at this link https://chronicdata.cdc.gov/Heart-Disease-Stroke-Prevention/Behavioral-Risk-Factor-Surveillance-System-BRFSS-N/ikwk-8git, on the CDC Data website. However, the URL for the direct download of the file is https://chronicdata.cdc.gov/api/views/j32a-sa6u/rows.csv?accessType=DOWNLOAD.
2. This direct download URL is used in a Databrick to access the file, removing the need for saving the file to a blob. This way, if the file is updated on the website, one simply needs to run the Databrick again to get updated data.
3. Use the python library requests and apply requests.get() to the download url. Then get the content from this request with the .content method, decode it with the .decode method, and turn it into a file object to be read with the StringIO function in the python IO library.
4. Make a csv reader object and apply it to the content file given by StringIO.
5. Loop through rows in the reader object. Make the first row into the column names. Make the rest into a list of lists that hold the data. Finally, create a spark dataframe from the data and the column names.

Transform:

1. The first transformation step was to drop the unnecessary columns using the pyspark drop method in Databricks. Here is a list of all the columns that were dropped:
    a. ["Locationabbr", "Confidence_limit_Low", "Confidence_limit_High", "Display_order", "Data_value_unit", "Data_value_type", "Data_Value_Footnote_Symbol", "DataSource", "ClassId", "TopicId",

"LocationID", "BreakoutID", "BreakOutCategoryID", "QuestionID", "RESPONSEID"]

2. The second transformation step performed on this data was to select only the year the research is focusing on, 2018. This was performed with the where method:
    a. newdf = df.where(df.Year == 2018)
3. The third transformation step is to split the column “GeoLocation”, which contains latitude and longitude coordinates, into separate columns for its parts. This was accomplished with the pyspark split method.
    a. df2018 = df2018.withColumn("Latitude", split(df2018["GeoLocation"],",").getItem(0)).withColumn("Longitude", split(df2018["GeoLocation"],",").getItem(1))
4. The fourth transformation step is to remove the remaining parentheses from the newly created latitude and longitude columns using the pyspark translate method.
    a. df2018 = df2018.withColumn("Latitude", translate("Latitude", "(", ""))
    b. df2018 = df2018.withColumn("Longitude", translate("Longitude", ")", ""))
5. The last transformation step is to drop the “GeoLocation” column as it is no longer necessary. This was accomplished in the same way as step 1, using the pyspark drop method.

Load:

1. This data will then be loaded into the group’s SQL Database. This uses the standard command for writing from Databricks to SQL with the following parameters:
    a. database = ‘group2’
    b. tablename = ‘dbo.CDC_Metro_Data’
    c. user = ‘group2’
    d. password = ‘capstonedev10!!’
    e. server = ‘gen10-data-fundamentals-21-11-sql-server.database.windows.net’
2. Now the 2018 table is in an SQL Database, and the loading is complete.

US Census Bureau ACS:

Extract:

1. The table used by the group from the census comes from the 2018 American Community Survey. It is accessible at this link, Census - Table Results
2. However, the group decided to use the Census API to call this table instead. After importing the 'requests' python library, use the command requests.get() and then the census API URL for this table, which is:
    a. https://api.census.gov/data/2018/acs/acs1/subject?get=group(S2701)&for=state:*&key={key}
    b. The 'key' is set as a variable and corresponds to this groups API key. Each group must obtain their own key from the census for this search to work properly.
3. The response retrieved from the Census API is in a csv format, with metadata in the form of complex column codes rather than names. A variables table containing the dictionary mapping between code and name is retrieved from the Census API to assist in the conversion from code to name.
    a. Here is the API URL for the variables table, 'https://api.census.gov/data/2018/acs/acs1/subject/variables.json'
4. Use requests.get() to retrieve the variables table. It will be important eventually when renaming the columns in the original table.

Transform:

1. After inspecting the Census table and the variables table, the group found some of the columns are unnecessary for analysis. These columns end in either 'M', 'MA' or 'EA'. Get all the indices of these columns and store them in a list.
2. Loop through each row of the Census table and assign the string "None" to cells that are None. Then, create a nested loop for the list of indices from the previous step. Within the nested loop, remove the unwanted columns from each row using .pop() method. Each time pop is used, an offset variable should increment by 1, and the indices for .pop() should be adjusted based on this offset variable. After the nested loop is

complete, append the current row to a list variable. This new variable will be the new table with only columns ending in “E”.

3. Next, use the variables table to rename the columns of the new table using the column codes as a map. Create a for loop from the first row of the new table variable. Each iteration value represents a column code. The corresponding name for the column code is stored in the variables table, under ‘variables’, ‘column code’, ‘label’. Replace the column code in the new table with the name obtained from the variables table.
4. With the columns properly named and table properly cleaned, convert the new table variable into a pandas dataframe using pandas.read_csv().
5. The table is a collection of data from different categories. To isolate each category, use the .filter() method from the dataframe to filter column names containing the string “Total”. Use the .split() method on the column names with the delimiter “!!”. The second cell of the list returned by .split() is the name of the category. Append the category name into a list, then convert the list into a set to remove all the duplicates.
6. Construct a for loop from the list of categories. Use .filter() method on the table dataframe to obtain columns specific for each category. Append each .filter() result into a list of dataframes to be loaded into MSSQL.

Load:

1. This data will then be loaded into the group’s SQL Database. This uses the standard command for writing from Databricks to SQL with the following parameters:
    a. database = ‘group2’
    b. user = ‘group2’
    c. password = ‘capstonedev10!!’
    d. server = ‘gen10-data-fundamentals-21-11-sql-server.database.windows.net’
2. The table name specification will change for each table created in the for loop that writes the dataframes to SQL. Here is a list of the tables created.
    a. dbo.Age
    b. dbo.Disability_Status
    c. dbo.Educational_Attainment

d. dbo.Employment_Status
e. dbo.Household_Income_In_2018_Inflation_Adjusted_Dollars
f. dbo.Living_Arrangements
g. dbo.Nativity_And_US_Citizenship_Status
h. dbo.Race_And_Hispanic_Or_Latino_Origin
i. dbo.Ratio_of_Income_To_Poverty_Level_In_The_Past_12_Months
j. dbo.Sex
k. dbo.Total_Agg
l. dbo.Work_Experience

3. Be sure to create a datatype conversion list, that lists the SQL datatypes for each column. Occasionally without specifying the proper data type for SQL, an exception is raised.
4. Now the group's SQL Database is complete.

Conclusion: The group used four datasets to complete this project, each needing different types of and levels of transformation. The first comes from Kaggle and was used to create a Machine Learning Model to predict heart disease in a patient. The second and third come from the CDC, studying heart disease and risk factors across different states and many US metropolitan areas. Finally, the fourth dataset comes from the US Census Bureau ACS and involves data on nationwide health insurance coverage broken down by several break out groups.