****

**Prevalence of Type-1 Diabetes Diagnoses Based on Demographic Disparities**

**Drexel University LeBow College of Business​**

**Fall Term 2023**

**Team 4**

**User Manual**

Table of Contents

[Data Flow Overview 2](#_Toc151386608)

[Data Cleanup and Transformation Process in Power Query M Language 2](#_Toc151386609)

[Merging Data in Python 4](#_Toc151386610)

[Exploratory Data Analysis in PowerBI 5](#_Toc151386611)

[Clustering with BIRCH in Python 6](#_Toc151386612)

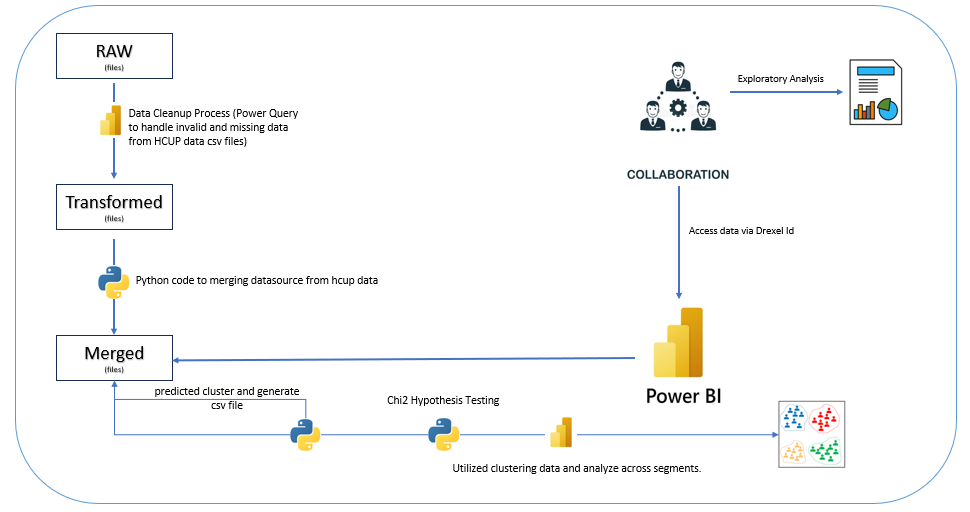
[Clustering with KModes in Python 7](#_Toc151386613)

[Cluster Profiles Analysis in PowerBI 7](#_Toc151386614)

[Chi-Square Hypothesis Testing in Python 9](#_Toc151386615)

[Appendix: How to get data import into PowerBI and Python 10](#_Toc151386616)

# Data Flow Overview



* The data processing begins with raw files, which undergo cleanup in Power Query to rectify missing or invalid entries.
* Post-cleansing, Python scripts to merge the Health Care & Utilization Project (HCUP) datasources, creating a consolidated "Transformed" dataset.
* This data is then analyzed to predict clusters and is saved as a CSV file for further cluster analysis.
* Concurrently, utilized the merged data output file in exploratory analysis offers insights into data patterns.
* Collaboration through Drexel ID access, and the final insights are visualized in Power BI, utilizing the clustered data for segment analysis patient demographic and type 1 diabetes diagnosis.

# Data Cleanup and Transformation Process in Power Query M Language



Data Source Import:

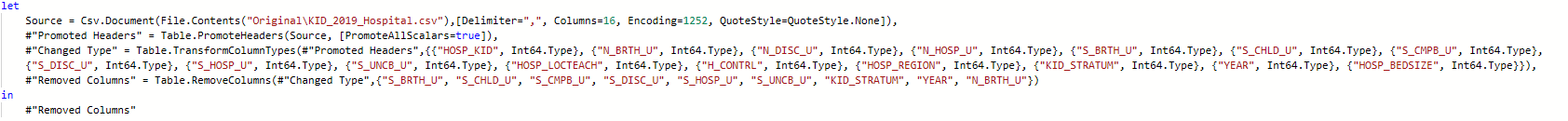
* Data imported from a ***KID\_2019\_Core.csv*** file located on your system and saved it as ***HCUP\_Data\_Cleanup.pibx*** format. Specified delimiters and encoding types ensure accurate data reading.

Header Promotion:

* The first row of the dataset is promoted to serve as headers, which helps in column identification.

Data Processing and Imputing:

* Each column is assigned a specific data type, such as text or integer. This ensures consistent data handling and reduces errors during analysis.
* Rows are filtered based on specific criteria. In this instance, it filters where PL\_NCHS is greater than or equal to 1 and HCUP\_ED is equal to 1 and also same process for AWEEKEND. This narrows down the dataset to more relevant entries.
* Specific columns are removed to declutter the dataset and focus on essential data. This can be especially helpful when dealing with extensive datasets with many variables.
* 'No Data Recorded' values are replaced in multiple columns to ensure data consistency. This step is vital for maintaining the integrity of the dataset and preparing it for further analysis.
* The cleaned data, after all the specified transformations, is presented as 'Replaced Value9'. This data is now ready for further analysis or visualization.



Data Source Import:

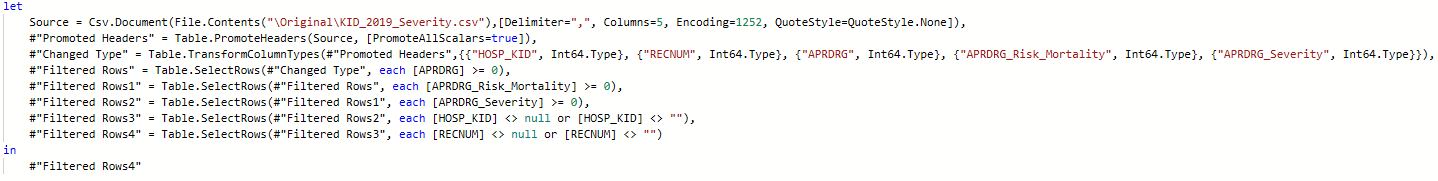
* Data imported from a ***KID\_2019\_Hospital.csv*** file located on your system.

Header Promotion:

* The first row of the dataset is promoted to headers, as identification throughout the process.

Data Processing and Imputing:

* Individual columns are given distinct data types, such as integer, to ensure uniform data handling.
* Certain columns are discarded to focus on the more relevant data points. This step helps simplify the dataset and make subsequent analysis more straightforward.
* After all transformations, the data is outputted as 'Removed Columns'. This cleaned data is now prepared for any subsequent operations or analyses.

****

Data Source Import:

* Data imported from a ***KID\_2019\_Severity.csv*** file located on your system.

Header Promotion:

* The initial row of the dataset is set as headers, enabling clearer column identification throughout the data processing phase.

Data Processing and Imputing:

* Columns, such as 'HOSP\_KID', 'RECNUM', 'APRDRG', 'APRDRG\_Risk\_Mortality', and 'APRDRG\_Severity', are assigned specific data types, predominantly integer, to ensure consistent data handling.
* Step 1: Rows where the 'APRDRG' column is not equal to 0 are selected.
* Step 2: From the filtered data in Step 1, rows with a value greater than 0 in the 'APRDRG\_Risk\_Mortality' column are further filtered.
* Step 3: Further refining the data, rows where the 'APRDRG\_Severity' column has a value equal to or above 0 are selected.
* Step 4: From the previous selection, rows without a blank 'HOSP\_KID' column are isolated.
* Step 5: Lastly, only rows where the 'RECNUM' column is not null or blank are fitlered.
* After all the transformation stages, the dataset is presented as 'Filtered Rows4'. This refined dataset is now set for data analysis.
* Lastly, ***HCUP\_Data\_Cleanup.pibx*** to extract the new cleanup and transform .csv files under the (*Data Cleaning/Transformed*) folder.

# Merging Data in Python

Dataset Importation:

* Import ‘dask.dataframe’ library for parallel computing with larger datasets.
* Using Dask, three datasets — ***Hospital***, ***Core***, and ***Severity*** — are read into memory.

Merging Hospital with Severity Dataset:

* The Hospital and Severity datasets are merged on the 'HOSP\_KID' column. It is a 'left join', which means all entries from the Hospital dataset and corresponding entries from the Severity dataset will be included.
* The resultant merged dataset is saved as a ***Hospital\_Severity.csv*** file in the specified output directory (*Dataset/Data Cleaning/Merged*).

Merging Hospital with Core Dataset:

* Similarly, the Hospital and Core datasets are merged on the 'HOSP\_KID' column using a 'left join'.
* This merged dataset is also saved as a ***Hospital\_Core.csv*** file in the designated output location (Dataset/Data Cleaning/Merged).

# Exploratory Data Analysis in PowerBI

Dataset Importation:

* Data imported from the location *\Dataset\Data Cleaning\Merged* folder **Hospital\_Core.csv, Hospital\_Severity.csv** and **icd10cm\_Codes\_Diagnosis.csv** file located on your system and saved it as ***HCUP\_EDA.pibx*** format.

Visualization:

* *Distribution by Age Group and Gender used by Stacked column chart*: Y-Axis shows the percentage of hospital discharges (% grand total count of ***RECNUM*** columns). X-Axis is presented by ***GENDER*** column. In this chart, ***AGE\_GROUPS*** custom column was created, and apply as in stack column chart Legend.
* *Distribution by Age Group and Race used by Stacked column chart:* Y-Axis shows the percentage of hospital discharges (% grand total count of ***RECNUM*** columns). X-Axis is presented by ***RACE\_DESC*** column. In this chart, ***AGE\_GROUPS*** custom column was created, and apply as in stack column chart Legend.
* *Distribution by Race used by Stacked bar chart:* X-Axis shows the percentage of hospital discharges (% grand total count of ***RECNUM*** columns). Y-Axis is presented by ***RACE\_DESC*** column.
* *Number of Hospitals in US Regions used by Donut chart:* ***HOSPITAL\_REGIONS*** column used as Legend and Value used the (count of ***HOSP\_KID)*** column as total number of hospitals across regions.
* *Average age of LOS by Gender and Age Group used by Clustered column chart*: X-Axis shows ***Gender*** column and Y-Axis presented (***Average of LOS***) average length of stay column. ***Age Group*** column apply in the Legend in the clustered column chart.
* *Distribution of Diabetes in KID’s Inpatient used by Pie chart*: ***Diabetes\_Type*** column used as in Legend and Value used the (count of ***RECNUM****)* column as total number of diabetes.
* *Top 5 Primary Diagnoses in 2019 used by Stacked column chart*: X-Axis shows the diagnosis (***Short Name\_DX1*** columns from icd10 table). Y-Axis is presented by total discharge which is from custom created column name called (% of ***Total Discharges***). Apply page level filters for ***AGE*** column is less than or equal 18 and custom column ***Diabetes\_Flag*** is selected as Diabetes.

# Clustering with BIRCH in Python

Dataset Importation:

* Import libraries such as Pandas, Dask, and Numpy are imported to handle data manipulation and computations.
* Datasets are read into memory using Dask for efficient parallel computing.

Data Preprocessing:

* The **hospital\_core\_df** and **hospital\_severity\_df** dataframes are created.
* The **hospital\_core\_df** is filtered based on age, including only individuals aged 0 to 18.
* A vectorized approach is used to identify records with a pre-existing diagnosis of type 1 diabetes, searching across multiple diagnosis columns.

Feature Engineering:

* A custom function *extract\_diagnosis\_codes* is applied to extract all relevant diagnosis codes related to type 1 diabetes.
* The diagnosis codes are then one-hot encoded to prepare them for machine learning models.

Data Transformation and Cleaning:

* Columns are renamed for clarity, and null values are handled appropriately to ensure data integrity.
* The dataframes **hospital\_core\_df** and **hospital\_severity\_df** are merged on the **'RECNUM**' column using an inner join.

Data Scaling and Dimensionality Reduction:

* The StandardScaler function from Scikit-learn is used to scale the features.
* Principal Component Analysis (PCA) is performed to reduce dimensionality, and the explained variance is visualized to determine the number of components to retain.

Elbow Method:

* The Elbow Method is used to determine the optimal number of clusters, which is visualized through a plot.
* A summary of the clusters is obtained by grouping the data by the cluster labels and computing the mean for each feature.

Cluster Profiling with BIRCH Algorithm:

* After compare and evaluate with other method like **KMeans**, **DBSCAN**. **BIRCH** method is significantly better in *Silhouette Score*. So, we decided BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) algorithm is applied for further analysis.
* The data is then grouped by the 'BIRCH\_Cluster' labels to profile the clusters. The mean of each feature within each cluster is calculated to provide insights into the characteristics of the clusters.

# Clustering with KModes in Python

Dataset Importation:

* Import libraries such as Pandas, Dask, and Numpy and read the hospital datasets using Dask for efficient memory.
* Apply age filters to isolate patients between 0 to 18 years and filter diagnosis codes for type 1 diabetes.

Data Transformation:

* Categorize patients into age groups (*'Infants'*, *'Children'*, and *'Adolescents'*). Extract relevant diagnosis codes and convert Dask dataframes to Pandas dataframes for further processing.

Elbow Method:

* Use the Elbow method to determine the optimal number of clusters by plotting the cost function against the number of clusters.

KModes Algorithm:

* Apply the KModes clustering algorithm using the optimal number of clusters. Ensure that the algorithm is initialized with a consistent random seed for reproducibility.

Export Predicted Cluster Dataset:

* Export the predicted clustering dataframe with assigned clusters to a **HCUP\_Clustering.csv** file for further cluster data analysis.

# Cluster Profiles Analysis in PowerBI

Dataset Importation:

* Data imported from the location \Dataset\Data Cleaning\Merged folder **HCUP\_Clustering.csv**, and **icd10cm\_Codes\_Diagnosis.csv** file located on your system and saved it as **HCUP\_Cluster\_Profiles.pibx** format.

Visualizations:

*Distribution of Age*

* **Cluster 1**: Distribution of Age used by Donut chart, ***AGE\_GROUP*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **1** on visual level.
* **Cluster 2**: Distribution of Age used by Donut chart, ***AGE\_GROUP*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **2** on visual level.
* **Cluster 3**: Distribution of Age used by Donut chart, ***AGE\_GROUP*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **3** on visual level.

*Distribution of Gender*

* **Cluster 1**: Distribution of Gender used by Donut chart, ***GENDER*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **1** on visual level.
* **Cluster 2**: Distribution of Gender used by Donut chart, ***GENDER*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **2** on visual level.
* **Cluster 3**: Distribution of Gender used by Donut chart, ***GENDER*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **3** on visual level.

*Distribution of Race*

* **Cluster 1**: Distribution of Race used by Donut chart, ***RACE\_DESC*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **1** on visual level.
* **Cluster 2**: Distribution of Race used by Donut chart, ***RACE\_DESC*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **2** on visual level.
* **Cluster 3**: Distribution of Race used by Donut chart, ***RACE\_DESC*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **3** on visual level.

*Distribution of Hospital Region and Age Groups*

* **Cluster 1**: Distribution of Hospital Region and Age Groups used by Clustered bar chart, Y-Axis shows the ***HOSPITAL\_REGION*** column. X-Axis is presented by (count of ***RECNUM***) column as No. of Discharges. ***AGE\_GROUP*** column as in the Legend in the clustered bar chart. Apply filter for ***CLUSTERS*** column equal **1** on visual level.
* **Cluster 2**: Distribution of Hospital Region and Age Groups used by Clustered bar chart, Y-Axis shows the ***HOSPITAL\_REGION*** column. X-Axis is presented by (count of ***RECNUM***) column as No. of Discharges. ***AGE\_GROUP*** column as in the Legend in the clustered bar chart. Apply filter for ***CLUSTERS*** column equal **2** on visual level.
* **Cluster 3**: Distribution of Hospital Region and Age Groups used by Clustered bar chart, Y-Axis shows the ***HOSPITAL\_REGION*** column. X-Axis is presented by (count of ***RECNUM***) column as No. of Discharges. ***AGE\_GROUP*** column as in the Legend in the clustered bar chart. Apply filter for ***CLUSTERS*** column equal **3** on visual level

*Distribution of Discharges by Hospital Region*

* **Cluster 1**: Distribution of Discharges by Hospital Region used by Donut chart, ***HOSPITAL\_REGION*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **1** on visual level.
* **Cluster 2**: Distribution of Discharges by Hospital Region used by Donut chart, ***HOSPITAL\_REGION*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **2** on visual level.
* **Cluster 3**: Distribution of Discharges by Hospital Region used by Donut chart, ***HOSPITAL\_REGION*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **3** on visual level.

*Average Length of Stay based on Severity*

* **Cluster 1**: Average Length of Stay based on Severity used by Clustered column chart, X-Axis shows ***ALL\_PATIENT\_SEVERITY*** column and Y-Axis presented (***Average of LOS***) average length of stay column. Apply filter for ***CLUSTERS*** column equal **1** on visual level.
* **Cluster 2**: Average Length of Stay based on Severity used by Clustered column chart, X-Axis shows ***ALL\_PATIENT\_SEVERITY*** column and Y-Axis presented (***Average of LOS***) average length of stay column. Apply filter for ***CLUSTERS*** column equal **2** on visual level.
* **Cluster 3**: Average Length of Stay based on Severity used by Clustered column chart, X-Axis shows ***ALL\_PATIENT\_SEVERITY*** column and Y-Axis presented (***Average of LOS***) average length of stay column. Apply filter for ***CLUSTERS*** column equal **3** on visual level.

# Chi-Square Hypothesis Testing in Python

Dataset Importation:

* Import libraries such as Pandas, and read the cluster datasets **HCUP\_Clustering.csv** to dataframe. It includes relevant columns for the analysis, including 'GENDER', 'ALL\_PATIENT\_SEVERITY','TYPE1\_DIABETES\_CODES','HOSPITAL\_REGION', 'RACE\_DESC', 'LOS', and 'AGE\_GROUP'.
* To conducting hypothesis tests, first to import scipy.stats library from stats.

Gender and Severity of Diagnosis:

* Create a contingency table for 'GENDER' and 'ALL\_PATIENT\_SEVERITY' whether the test is between two different categorical variables are associate or not.
* Perform a chi-square test and interpret the results.

Gender and Type 1 Diabetes Codes:

* Create a contingency table for 'GENDER' and 'TYPE1\_DIABETES\_CODES'.
* Perform a chi-square test and interpret the results.

Hospital Region vs. Ethnicity:

* Create a contingency table for 'HOSPITAL\_REGION' and 'RACE\_DESC'.
* Perform a chi-square test and interpret the results.

Ethnicity vs. Type 1 Diabetes Codes:

* Create a contingency table for 'RACE\_DESC' and 'TYPE1\_DIABETES\_CODES'.
* Perform a chi-square test and interpret the results.

Length of Stay vs. Age Group:

* Create a contingency table for 'LOS' and 'AGE\_GROUP'.
* Perform a chi-square test and interpret the results.

Length of Stay vs. Severity:

* Create a contingency table for 'LOS' and 'ALL\_PATIENT\_SEVERITY'.
* Perform a chi-square test and interpret the results.

Age Group vs. Severity:

* Create a contingency table for 'AGE\_GROUP' and 'ALL\_PATIENT\_SEVERITY'.
* Perform a chi-square test and interpret the results.

Interpreting the Results:

* These hypothesis tests will output the Chi-square statistic, p-value, degrees of freedom, and a conclusion to accept or reject the null hypothesis based on 5% significance level

# Appendix: How to get data import into PowerBI and Python

Prerequisites:

* Power BI Desktop and Python (Anaconda) Jupyter Notebook installed on your computer.

List of File Name and Type:

* Power BI

1. HCUP\_Data\_Cleanup (pbix)
2. HCUP\_EDA (pbix)
3. HCUP\_Cluster\_Profiles (pbix)

* Python Jupyter Notebook
  + 1. hcup\_merging\_datasource\_v1(W1\_Final)
    2. hcup\_clustering\_v2.8.1(W4\_FInal)
    3. hcup\_clustering\_v2.6.1(W3\_Final)
    4. hcup\_chi\_square\_testing\_v2(W5\_Final)

Open Power BI Desktop

* Launch the Power BI Desktop application on your computer.

Get Data

* In the Home tab on the ribbon, click on the 'Get Data' option.
* Select 'Text/CSV' from the options presented.

Open CSV File

* Navigate through your directories and select the CSV file you want to import.
* Click 'Open' to proceed.

Preview Data

* Power BI will present a preview of the data.
* Review to ensure that the columns and rows are displayed correctly.

Load or Edit

* To make changes to the data before importing, click 'Transform Data'. This will open the Power Query Editor where you can perform data cleaning and transformation.
* If no changes are needed, click 'Load' to import the data directly into Power BI.

Data Model

* Once loaded, the dataset will appear in the 'Fields' pane.
* You can now use this data to create reports and visuals.

Save Your Work

* Save your Power BI project by clicking on 'File' and then 'Save', or by pressing Ctrl + S.
* Name your project and choose a location to save the file.

Launch Jupyter Notebook:

* Type jupyter notebook and hit Enter.
* Your default web browser will open with the Jupyter file explorer.
* Click on the "Open" in the menu of the Jupyter file explorer.

Importing CSV Data:

* Change your file path into pd.read\_csv('your\_data.csv')

---------------------------------------------------------------------End-------------------------------------------------------------------------