ASTHMA ANALYTICS:

BRIDGING THE GAP IN ASTHMA CARE

Team 6:
Otabek Juraev
Jozil Jacob
Saloni Tiwari
Yashveer Shrimal
Roshan Dubey

Table of Content:

Data Set Up

We began by outlining the data flow for our project, we dumped the data into the data warehouse and made 4 main tables (Severity, Hospital, GRPS and Core). This provided a clear roadmap for our subsequent analysis, ensuring data integrity and facilitating targeted insights.

Data Cleaning

Next, we cleaned the raw data to ensure accuracy and consistency. We addressed missing values, outliers, and inconsistencies, creating a refined dataset that formed the foundation for our analysis. For instance, the length of stay column had invalid and inconsistent values like “A” and “B”, We cleaned these values to have data consistency across our entire dataset

As we moved further in our analysis, we created separate tables for each specific disease based on the diagnosis code, having specific table for each disease helped us analyse each disease in detail.

Exploratory Data Analysis using SQL and Tableau

In this phase, we leveraged SQL queries to extract relevant insights from the dataset. Subsequently, we utilized Tableau for visual exploration, creating interactive visuals to uncover patterns, trends, and relationships within the data.

Statistical Testing Using Chi-Squared Test

Chi-Squared Test for statistical analysis, examining the relationships between categorical variables. The test generated a p-value and a Chi-Square value. A low p-value (<0.05) suggested significant associations, enabling us to make informed decisions about the validity of patterns within our dataset and test our initial hypothesis.

K-Means Clustering

Applying K-Means clustering, we segmented the data into distinct groups based on similarities. This unsupervised machine learning technique allowed us to identify inherent patterns and groupings within the dataset.

For better results we created bins for variables like Age and Total Charges.

The clustering algorithm divided the entire data for asthmatic patients into 5 clusters.

K-Modes Clustering

We also employed K-Modes clustering, specifically designed for categorical data. This approach further enriched our understanding by revealing patterns in non-numeric variables, providing a holistic view of the dataset.

The clustering algorithm divided the entire data for asthmatic patients into 3 clusters.

Analysis of Cluster Results in Tableau

To interpret and communicate the cluster results effectively, we integrated the findings into Tableau. By visualizing the clusters and their characteristics, we gained valuable insights into the underlying structure of the data, facilitating informed decision-making.

Code

SQL Queries:

```
--Query to create separate table for respiratory diseases based on
diagnosis code:
CREATE TABLE capstone2.RESPIRATORY_DISEASES AS

SELECT HOSP_KID
        ,RECNUM
        ,DXCCSR_RSP001
        ,DXCCSR_RSP002
        ,DXCCSR_RSP003
        ,DXCCSR_RSP004
        ,DXCCSR_RSP005
        ,DXCCSR_RSP006
        ,DXCCSR_RSP007
        ,DXCCSR_RSP008
        ,DXCCSR_RSP009
        ,DXCCSR_RSP010
        ,DXCCSR_RSP011
        ,DXCCSR_RSP012
        ,DXCCSR_RSP013
        ,DXCCSR_RSP014
        ,DXCCSR_RSP015
        ,DXCCSR_RSP016
        ,DXCCSR_RSP017
FROM `capstone - 400517. capstone2.kid_GPRS`
```

```
--QUERY FOR asthma cases by Gender
SELECT FEMALE
        ,count(*)
FROM `capstone - 400517. capstone2.table_core`
WHERE Asthma <> '0'
        AND Asthma IS NOT NULL
GROUP BY 1
```

```
--QUERY FOR asthma cases by INCOME
SELECT ZIPINC_QRTL
        ,count(*)
FROM `capstone - 400517. capstone2.table_core`
```

```
WHERE Asthma <> '0'
GROUP BY 1
```

```
--QUERY FOR asthma cases by race
SELECT RACE
        ,count(*)
FROM `capstone - 400517. capstone2.table_core`
--WHERE Asthma <> '0'
GROUP BY 1
```

```
--query for asthma and infectious diseases
SELECT Asthma
        ,INF_All
        ,COUNT(*) AS INFANDASTHMA
FROM `capstone - 400517. capstone2.table_core`
WHERE Asthma <> '0'
        AND INF_All <> 0
GROUP BY 1
        ,2
```

```
--COMORBIDITY ANALYSIS FOR ASTHMA AND OTHER DISEASES
SELECT RECNUM
FROM `capstone - 400517. capstone2.table_core`
WHERE (
                Blood_All <> 0
                OR MAL_All <> 0
                OR MBD_ALL <> 0
                OR END_ALL <> 0
                OR INF_All <> 0
                OR DIG_ALL <> 0
                OR NVS_ALL <> 0
                OR PRG_All <> 0
                OR CIR_All <> 0
                OR SKN_All <> 0
                OR MUS_All <> 0
                OR Tumor_All <> 0
                OR EYE_All <> 0
                OR EAR_All <> 0
                )
        AND Asthma <> '0'
```

```
--Data cleaning in gender
DELETE
FROM `capstone - 400517. capstone2.table_core`
WHERE FEMALE IN (
                'A'
                ,'C'
                ,'nan'
                );
```

```
--query to get total charges by payer
SELECT
        --RECNUM
        Pay1
        ,sum(TotalChargesInteger)
FROM `capstone - 400517. capstone2.table_core`
WHERE Asthma <> '0'
        AND (
                Blood_All = 0
```

```
                AND MAL_All = 0
                AND MBD_ALL = 0
                AND END_ALL = 0
                AND INF_All = 0
                AND DIG_ALL = 0
                AND NVS_ALL = 0
                AND PRG_All = 0
                AND CIR_All = 0
                AND SKN_All = 0
                AND MUS_All = 0
                AND Tumor_All = 0
                AND EYE_All = 0
                AND EAR_All = 0
                )
        AND PAY1 IN (
                '1'
                ,'2'
                )
--and race <> 'nan'
--and Race not like '%nan'
GROUP BY 1
```

```
--Query to check count of each endocrine disease
SELECT SUM(CASE
                        WHEN CAST(DXCCSR_END001 AS INT64) = 1
                            THEN 1
                        ELSE 0
                        END) AS Count_1
        ,SUM(CASE
                        WHEN CAST(DXCCSR_END002 AS INT64) = 1
                            THEN 1
                        ELSE 0
                        END) AS Count_2
        ,SUM(CASE
                        WHEN CAST(DXCCSR_END003 AS INT64) = 1
                            THEN 1
                        ELSE 0
                        END) AS Count_3
        ,SUM(CASE
                        WHEN CAST(DXCCSR_END004 AS INT64) = 1
                            THEN 1
                        ELSE 0
                        END) AS Count_4
        ,SUM(CASE
                        WHEN CAST(DXCCSR_END006 AS INT64) = 1
                            THEN 1
                        ELSE 0
                        END) AS Count_6
        ,SUM(CASE
                        WHEN CAST(DXCCSR_END007 AS INT64) = 1
                            THEN 1
                        ELSE 0
                        END) AS Count_7
        ,SUM(CASE
                        WHEN CAST(DXCCSR_END008 AS INT64) = 1
                            THEN 1
                        ELSE 0
                        END) AS Count_8
        ,SUM(CASE
                        WHEN CAST(DXCCSR_END009 AS INT64) = 1
```

```
                        THEN 1
                    ELSE 0
                    END) AS Count_9
        ,SUM(CASE
                    WHEN CAST(DXCCSR_END010 AS INT64) = 1
                        THEN 1
                    ELSE 0
                    END) AS Count_10
        ,SUM(CASE
                    WHEN CAST(DXCCSR_END011 AS INT64) = 1
                        THEN 1
                    ELSE 0
                    END) AS Count_11
        ,SUM(CASE
                    WHEN CAST(DXCCSR_END012 AS INT64) = 1
                        THEN 1
                    ELSE 0
                    END) AS Count_12
        ,SUM(CASE
                    WHEN CAST(DXCCSR_END013 AS INT64) = 1
                        THEN 1
                    ELSE 0
                    END) AS Count_13
        ,SUM(CASE
                    WHEN CAST(DXCCSR_END014 AS INT64) = 1
                        THEN 1
                    ELSE 0
                    END) AS Count_14
        ,SUM(CASE
                    WHEN CAST(DXCCSR_END015 AS INT64) = 1
                        THEN 1
                    ELSE 0
                    END) AS Count_15
        ,SUM(CASE
                    WHEN CAST(DXCCSR_END016 AS INT64) = 1
                        THEN 1
                    ELSE 0
                    END) AS Count_16
        ,SUM(CASE
                    WHEN CAST(DXCCSR_END017 AS INT64) = 1
                        THEN 1
                    ELSE 0
                    END) AS Count_17
        ,
FROM `capstone - 400517. capstone2.ENDOCRINE_DISEASES`
```

```
--Query to analyze asthma and length of stay
SELECT RECNUM
        ,LOS
FROM `capstone - 400517. capstone2.table_core`
WHERE (
                Blood_All <> 0
                OR MAL_All <> 0
                OR MBD_ALL <> 0
                OR END_ALL <> 0
                OR INF_All <> 0
                OR DIG_ALL <> 0
                OR NVS_ALL <> 0
                OR PRG_All <> 0
                OR CIR_All <> 0
                OR SKN_All <> 0
                OR MUS_All <> 0
```

```
                OR Tumor_All <> 0
                OR EYE_All <> 0
                OR EAR_All <> 0
                )
        AND Asthma <> '0'
        AND Asthma IS NOT NULL
```

```
--Query to get count of individual infectious diseases
SELECT SUM(CASE
                        WHEN CAST(DXCCSR_INF001 AS INT64) <> 0
                                THEN 1
                        ELSE 0
                        END) AS Count_INF001
        ,SUM(CASE
                        WHEN CAST(DXCCSR_INF002 AS INT64) <> 0
                                THEN 1
                        ELSE 0
                        END) AS Count_INF002
        ,SUM(CASE
                        WHEN CAST(DXCCSR_INF003 AS INT64) <> 0
                                THEN 1
                        ELSE 0
                        END) AS Count_INF003
        ,SUM(CASE
                        WHEN CAST(DXCCSR_INF004 AS INT64) <> 0
                                THEN 1
                        ELSE 0
                        END) AS Count_INF004
        ,SUM(CASE
                        WHEN CAST(DXCCSR_INF005 AS INT64) <> 0
                                THEN 1
                        ELSE 0
                        END) AS Count_INF005
        ,SUM(CASE
                        WHEN CAST(DXCCSR_INF006 AS INT64) <> 0
                                THEN 1
                        ELSE 0
                        END) AS Count_INF006
        ,SUM(CASE
                        WHEN CAST(DXCCSR_INF007 AS INT64) <> 0
                                THEN 1
                        ELSE 0
                        END) AS Count_INF007
        ,SUM(CASE
                        WHEN CAST(DXCCSR_INF008 AS INT64) <> 0
                                THEN 1
                        ELSE 0
                        END) AS Count_INF008
        ,SUM(CASE
                        WHEN CAST(DXCCSR_INF009 AS INT64) <> 0
                                THEN 1
                        ELSE 0
                        END) AS Count_INF009
        ,SUM(CASE
                        WHEN CAST(DXCCSR_INF010 AS INT64) <> 0
                                THEN 1
                        ELSE 0
                        END) AS Count_INF010
        ,SUM(CASE
                        WHEN CAST(DXCCSR_INF011 AS INT64) <> 0
                                THEN 1
                        ELSE 0
```

```
                        END) AS Count_INF011
        ,SUM(CASE
                        WHEN CAST(DXCCSR_INF012 AS INT64) <> 0
                            THEN 1
                        ELSE 0
                        END) AS Count_INF012
FROM `capstone - 400517. capstone2.Infectious_DISEASES`
```

```
--Query to clean total charges col.
--ALTER TABLE `capstone-400517.capstone2.table_core`
--ADD COLUMN TotalChargesInteger INT64;
-- Copy data from the existing column to the new column
UPDATE `capstone - 400517. capstone2.table_core`
SET TotalChargesInteger = CASE
                WHEN TOTCHG = 'nan'
                        THEN NULL -- or set to a default value
                WHEN TOTCHG = 'A'
                        THEN NULL
                WHEN TOTCHG = 'C'
                        THEN NULL
                WHEN TOTCHG = 'TOTCHG'
                        THEN NULL
                ELSE CAST(TOTCHG AS INT64)
                END
WHERE 1 = 1
```

```
--Query to get LOS AND TOTAL CHARGE FOR PATIENTS WITH ASTHMA AND OTHER
DISEASES
SELECT t1.DXCCSR_RSP009
        ,COUNT(*)
FROM (
        SELECT RECNUM
                ,DXCCSR_Default_DX1
                ,DXCCSR_RSP009
        FROM `capstone - 400517. capstone2.kid_GPRS` A11
        WHERE A11.DXCCSR_Default_DX1 = 'RSP009'
        ) t1
GROUP BY 1
```

```
--query to create separate table for CongenitalMalfunction
CREATE TABLE capstone2.CongenitalMalfunction_DISEASES AS

SELECT HOSP_KID
        ,RECNUM
        ,DXCCSR_MAL001
        ,DXCCSR_MAL002
        ,DXCCSR_MAL003
        ,DXCCSR_MAL004
        ,DXCCSR_MAL005
        ,DXCCSR_MAL006
        ,DXCCSR_MAL007
        ,DXCCSR_MAL008
        ,DXCCSR_MAL009
        ,DXCCSR_MAL010
FROM `capstone - 400517. capstone2.kid_GPRS`
```

```
--query to make new table for infectious diseases
CREATE TABLE capstone2.Infectious _DISEASES AS

SELECT HOSP_KID
        ,RECNUM
```

```
        ,DXCCSR_INF001
        ,DXCCSR_INF002
        ,DXCCSR_INF003
        ,DXCCSR_INF004
        ,DXCCSR_INF005
        ,DXCCSR_INF006
        ,DXCCSR_INF007
        ,DXCCSR_INF008
        ,DXCCSR_INF009
        ,DXCCSR_INF010
        ,DXCCSR_INF011
        ,DXCCSR_INF012
FROM `capstone - 400517. capstone2.kid_GPRS`
```

```
--Query to make range for LOS
UPDATE capstone2.table_core
SET LOS_Range = CASE
                WHEN LOS BETWEEN '0'
                                 AND '1'
                        THEN '0-1'
                WHEN LOS BETWEEN '2'
                                 AND '3'
                        THEN '2-3'
                WHEN LOS BETWEEN '4'
                                 AND '9'
                        THEN '4-9'
                WHEN LOS > '10'
                        THEN '10+'
                END
WHERE 1 = 1
```

```
--query to make range for total charges
UPDATE capstone2.table_core
SET TotalChargeRange = CASE
                WHEN TotalChargesInteger BETWEEN 0
                                 AND 12000
                        THEN 1
                WHEN TotalChargesInteger BETWEEN 12001
                                 AND 20000
                        THEN 2
                WHEN TotalChargesInteger BETWEEN 20001
                                 AND 40000
                        THEN 3
                WHEN TotalChargesInteger BETWEEN 40001
                                 AND 80000
                        THEN 4
                WHEN TotalChargesInteger > 80001
                        THEN 5
                END
WHERE 1 = 1
```

```
# Python script for calculating Chi-Squared value and p value
from scipy.stats import chi2_contingency
import matplotlib.pyplot as plt
import numpy as np

observed_data = [[101427, 1333965], [103168, 1248246]]

# Perform chi-square test
chi2, p, _, _ = chi2_contingency(observed_data)
```

```
print(f"P-value: {p}")

print(f"Chi-squared value: {chi2}")
```

Code for clustering:

```
# Filtering only for RSP009
results = job.to_dataframe()
results = results[results['DXCCSR_Default_DX1']== 'RSP009']
```

```
# Import numpy to handle NaN values
import numpy as np

# Replace all instances of 'nan' with NaN in the results DataFrame
results.replace('nan', np.nan, inplace=True)
```

```
# Drop NA values from table
results.dropna(inplace=True)


results.replace('A', np.nan, inplace=True)
```

```
# Drop NA values form table
results.dropna(inplace=True)
```

```
# check if we have NA values among unique values

cols = results.columns
for i in cols:

  print(i, results[i].unique())
```

```
#Converting all values into numeric so that we can run ML algorithms

import pandas as pd

# Convert all columns to numeric, coercing errors to NaN
```

```
results = results.apply(pd.to_numeric, errors='coerce')


# Convert to Int64
results = results.astype('Int64')
```

```
#Checking if the values converted into numeric

cols = results.columns
for i in cols:

  print(i, results[i].unique())
```

```
#Converting all values into numeric so that we can run ML algorithms

import pandas as pd

# Convert all columns to numeric, coercing errors to NaN
results = results.apply(pd.to_numeric, errors='coerce')


# Convert to Int64
results = results.astype('Int64')
```

```
#Checking if the values converted into numeric

cols = results.columns
for i in cols:

  print(i, results[i].unique())
```

```
# Separating the features that  we need for K-Mode algorthm

df_kmode = results[['AGE','FEMALE', 'RACE', 'HOSP_REGION',
'ZIPINC_QRTL', 'LOS', 'TOTCHG', 'PAY1']]
```

```
# Creating bins for Length of stay column

## labels = ['0-1', '2-3', '4-9', '10 and over']


bins = [-1, 1, 3, 9, float('inf')]  # Using -1 as the lower bound to
include 0
labels = [1, 2, 3, 4]

# Create a new column with the binned categories
```

```
df_kmode.loc[:, 'LOS'] = pd.cut(df_kmode['LOS'], bins=bins,
labels=labels)
```

```
# Creating bins for "Total charge" feature

## labels = ['0-12000', '12,000-20,000' '20,000-40,000', '40,000-
80,000', '80,000+']


bins = [-1, 12000, 20000, 40000, 80000, float('inf')]  # Using -1 as
the lower bound to include 0
labels = [1, 2, 3, 4, 5]

# Create a new column with the binned categories
df_kmode.loc[:, 'TOTCHG'] = pd.cut(df_kmode['TOTCHG'], bins=bins,
labels=labels)
```

```
# Creating bins for "AGE" feature of the model

## labels = ['0-1', '1-10' '11-20']


bins = [-1, 1, 10, 21]  # Using -1 as the lower bound to include 0
labels = [1, 2, 3]

# Create a new column with the binned categories
df_kmode.loc[:, 'AGE'] = pd.cut(df_kmode['AGE'], bins=bins,
labels=labels)
```

```
# Install K-Modes ML library

!pip install kmodes
```

```
# Import necessary libraries from KModes and create clusters


from kmodes.kmodes import KModes
import pandas as pd
from sklearn.preprocessing import OneHotEncoder

encoder = OneHotEncoder(sparse=False)
df_encoded = encoder.fit_transform(df_kmode)

km = KModes(n_clusters=3, init='Huang', n_init=5, verbose=1)
```

```
#Check of the clusters created in vector format

df_encoded
```

```
# Fit the cluster

clusters = km.fit_predict(df_encoded)
```

```
# Checking the optimal number of clusters using Elbow method

import matplotlib.pyplot as plt
from kmodes.kmodes import KModes

costs = []
for num_clusters in range(1, 10):
    km = KModes(n_clusters=num_clusters, init='Huang', n_init=5,
verbose=0)
    km.fit(df_encoded)
    costs.append(km.cost_)

plt.plot(range(1, 10), costs, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Cost')
plt.title('Elbow Method For Optimal Number of Clusters')
plt.show()
```

```
# Join the cluster results to the original table

df_kmode['Cluster'] = clusters
```

```
# Using dataframe df_kmode: save to csv file for future analysis and
visualization

df_kmode.to_csv('df_kmode_Asthma_primary_3.csv', index=False)
```

```
#Importing Library for kMeans
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

```
#Scaling the dataset
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df_asthma)
```

```
# Elbow Method
inertia = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=0)
    kmeans.fit(scaled_features)
    inertia.append(kmeans.inertia_)

plt.plot(range(1, 11), inertia)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.show()
```

```
# Initialize KMeans with k clusters
kmeans = KMeans(n_clusters=5, random_state=0)

# Fit the model
kmeans.fit(scaled_features)

# Get the cluster labels for each data point
labels = kmeans.labels_
```

```
# Add the cluster labels as a new column in the original DataFrame
df_asthma['Cluster'] = labels


#Export result to CSV
df_asthma.to_csv("/df_cluster_new.csv")
```