Factors Impacting Rates of Medical Readmission

Gerald Burke

Western Governors University

D206: Data Cleaning

Dr. Keiona Middleton

Revised: October 17, 2024

Contents

Factors Impacting Rates of Medical Readmission

A. Research Question

A1. The Question:

"What are the factors that impact rates of medical readmission?"

Readmissions can represent failures in first-line care and can burden the resources of both hospitals and insurance companies. Having a clearer image of what variables have an impact on readmission rates can help hospitals and insurance companies identify areas of opportunity for development in the interest of decreasing the readmission rates.

A2. The Variables:

1. CaseOrder

- **Data Category**: Qualitative (Nominal)
- **Example**: 1
- **Description**: A placeholder variable to preserve the original order of the raw data file.

2. Customer_id

- **Data Category**: Qualitative (Nominal)
- **Example**: 11223344
- **Description**: Unique patient ID assigned to each individual.

3. Interaction

- **Data Category**: Qualitative (Nominal)
- **Example**: 987654
- **Description**: Unique IDs related to patient transactions, procedures, and admissions.

4. City

- **Data Category**: Qualitative (Nominal)
- **Example**: New York
- **Description**: City of residence listed on the billing statement.

5. State

- **Data Category**: Qualitative (Nominal)
- **Example**: NY
- **Description**: State of residence listed on the billing statement.

6. County

- **Data Category**: Qualitative (Nominal)
- **Example**: Kings
- **Description**: County of residence listed on the billing statement.

7. Zip

- **Data Category**: Qualitative (Nominal)
- **Example**: 11201
- **Description**: ZIP code of the patient's residence.

8. Lat

- **Data Category**: Quantitative
- **Example**: 40.7128
- **Description**: Latitude of the patient's residence.

9. Lng

- **Data Category**: Quantitative
- **Example**: -74.0060
- **Description**: Longitude of the patient's residence.

10. Population

- **Data Category**: Quantitative
- **Example**: 8,398,748
- **Description**: Population within a one-mile radius of the patient's residence based on census data.

11. Area

- **Data Category**: Qualitative (Nominal)
- **Example**: Urban
- **Description**: Type of area where the patient resides (urban, suburban, rural).

12. TimeZone

- **Data Category**: Qualitative (Nominal)
- **Example**: EST
- **Description**: Time zone of the patient's residence.

13. Job

- **Data Category**: Qualitative (Nominal)
- **Example**: Teacher
- **Description**: Job of the patient or primary insurance holder as reported in the admissions information.

14. Children

- **Data Category**: Quantitative
- **Example**: 2
- **Description**: Number of children in the patient's household.

15. Age

- **Data Category**: Quantitative
- **Example**: 45
- **Description**: Age of the patient as reported in the admissions information.

16. Education

- **Data Category**: Qualitative (Ordinal)
- **Example**: Bachelor's Degree
- **Description**: Highest level of education achieved by the patient.

17. Employment

- **Data Category**: Qualitative (Nominal)
- **Example**: Employed
- **Description**: Employment status of the patient as reported in the admissions information.

18. Income

- **Data Category**: Quantitative
- **Example**: 55,000
- **Description**: Annual income of the patient or primary insurance holder.

19. Marital

- **Data Category**: Qualitative (Nominal)
- **Example**: Married
- **Description**: Marital status of the patient or primary insurance holder.

20. Gender

- **Data Category**: Qualitative (Nominal)
- **Example**: Female
- **Description**: Gender identity of the patient (male, female, nonbinary).

21. ReAdmis

- **Data Category**: Qualitative (Nominal)
- **Example**: Yes
- **Description**: Indicates whether the patient was readmitted to the hospital within a month of release.

22. VitD_levels

- **Data Category**: Quantitative
- **Example**: 24.7
- **Description**: Patient's vitamin D levels in ng/mL.

23. Doc_visits

- **Data Category**: Quantitative
- **Example**: 4
- **Description**: Number of times the primary physician visited the patient during hospitalization.

24. Full_meals_eaten

- **Data Category**: Quantitative
- **Example**: 2
- **Description**: Number of full meals the patient consumed while hospitalized.

25. VitD_supp

- **Data Category**: Quantitative
- **Example**: 3
- **Description**: Number of vitamin D supplements administered to the patient.

26. Soft_drink

- **Data Category**: Qualitative (Nominal)
- **Example**: No
- **Description**: Indicates whether the patient habitually drinks three or more sodas daily.

27. Initial_admin

- **Data Category**: Qualitative (Nominal)
- **Example**: Emergency admission
- **Description**: The means by which the patient was admitted to the hospital initially (emergency, elective, or observation).

28. HighBlood

- **Data Category**: Qualitative (Nominal)
- **Example**: Yes
- **Description**: Indicates whether the patient has high blood pressure.

29. Stroke

- **Data Category**: Qualitative (Nominal)
- **Example**: No
- **Description**: Indicates whether the patient has had a stroke.

30. Complication_risk

- **Data Category**: Qualitative (Ordinal)
- **Example**: Medium
- **Description**: Level of complication risk for the patient, assessed as high, medium, or low.

31. Overweight

- **Data Category**: Qualitative (Nominal)
- **Example**: Yes
- **Description**: Indicates whether the patient is considered overweight based on age, gender, and height.

32. Arthritis

- **Data Category**: Qualitative (Nominal)
- **Example**: No
- **Description**: Indicates whether the patient has arthritis.

33. Diabetes

- **Data Category**: Qualitative (Nominal)
- **Example**: Yes

- **Description**: Indicates whether the patient has diabetes.

34. Hyperlipidemia

- **Data Category**: Qualitative (Nominal)
- **Example**: Yes
- **Description**: Indicates whether the patient has hyperlipidemia (high cholesterol).

35. BackPain

- **Data Category**: Qualitative (Nominal)
- **Example**: No
- **Description**: Indicates whether the patient has chronic back pain.

36. Anxiety

- **Data Category**: Qualitative (Nominal)
- **Example**: Yes
- **Description**: Indicates whether the patient has been diagnosed with an anxiety disorder.

37. Allergic_rhinitis

- **Data Category**: Qualitative (Nominal)
- **Example**: No
- **Description**: Indicates whether the patient has allergic rhinitis.

38. Reflux_esophagitis

- **Data Category**: Qualitative (Nominal)
- **Example**: No
- **Description**: Indicates whether the patient has reflux esophagitis.

39. Asthma

- **Data Category**: Qualitative (Nominal)
- **Example**: Yes
- **Description**: Indicates whether the patient has asthma.

40. Services

- **Data Category**: Qualitative (Nominal)
- **Example**: Blood work

- **Description**: Primary service the patient received while hospitalized (blood work, intravenous, CT scan, MRI, etc.).

41. Initial_days

- **Data Category**: Quantitative
- **Example**: 5
- **Description**: The number of days the patient stayed in the hospital during the initial visit.

42. TotalCharge

- **Data Category**: Quantitative
- **Example**: 15,000
- **Description**: Total amount charged to the patient for daily services, averaged per day.

43. Additional_charges

- **Data Category**: Quantitative
- **Example**: 2,500
- **Description**: Additional charges for miscellaneous procedures, treatments, or medicines.

44. Item1

- **Data Category**: Quantitative (Ordinal)
- **Example**: 3
- **Description**: Rating of the importance of timely admission for the patient.

45. Item2

- **Data Category**: Quantitative (Ordinal)
- **Example**: 2
- **Description**: Rating of the importance of timely treatment for the patient.

46. Item3

- **Data Category**: Quantitative (Ordinal)
- **Example**: 4
- **Description**: Rating of the importance of timely visits for the patient.

47. Item4

- **Data Category**: Quantitative (Ordinal)
- **Example**: 1

- **Description**: Rating of the importance of reliability for the patient.

48. Item5

- **Data Category**: Quantitative (Ordinal)
- **Example**: 6
- **Description**: Rating of the importance of options provided for care.

49. Item6

- **Data Category**: Quantitative (Ordinal)
- **Example**: 5
- **Description**: Rating of the importance of available hours of treatment.

50. Item7

- **Data Category**: Quantitative (Ordinal)
- **Example**: 7
- **Description**: Rating of the importance of courteous staff interactions.

51. Item8

- **Data Category**: Quantitative (Ordinal)
- **Example**: 8
- **Description**: Rating of the importance of active listening from the doctor.

B. Data Cleaning Plan

B1. Proposed Plan for Assessing Data Quality

Duplicates

To check for duplicates I used the Pandas Data Frame duplicated() function with the value_counts() function.

Missing Values

For missing values, I wanted to get not only the count of the missing values, but the associated percentage of missing values in regards to the total number of records. I also aimed to create a visualization of the missing values. For those, I used the following:

I used the df.isnull().sum() method to get the count of missing values

I used the len() function to get the percentage with the formula:

df.isnull().sum() * 100 / len(df)

I created a new data frame for the display of the information using:

```
pd.DataFrame({‘Missing Count’ : missing_count, ‘Missing Percent’ : missing_percent})
```

I used a combination of Missingno and PyPlot to generate a matrix of missing data using the following code:

```
msno.matrix(df, fontsize=12, labels=True)

plt.title('Missing data matrix')

plt.xlabel(df.columns)

plt.show()
```

Outliers

To run the following commands on the data set, I made an array of the column names of each of the quantitative variables in the set:

```
quantitative_columns = ['Population', 'Children', 'Age', 'Income', 'VitD_levels', 'Doc_visits',
'Full_meals_eaten', 'VitD_supp', 'Initial_days', 'TotalCharge', 'Additional_charges']
```

I then iterated over the array of columns:

```
for column in quantitative_columns:
```

First I printed the minimum, maximum, and range of values present in each column:

```
  #Print min, max, and range of values present

  print(f'Minimum value: {df[column].min()}')

  print(f'Maximum value: {df[column].max()}')

  print(f'Value range: {df[column].max() - df[column].min()}')
```

I also took the sums of all values with a z-score above 3 or below -3.

```
  print(f'Values with a z-score over 3: {sum(stats.zscore(df[column]) > 3)}')

  print(f'Values with a z-score under -3: {sum(stats.zscore(df[column]) < -3)}')
```

From there I calculated Q1 and Q3 for each and determined the values that fell outside of the range:

(Q1 – 1.5 * IQR) – (Q3 + 1.5 * IQR)

```
q1 = np.quantile(df[column], 0.25)

  q3 = np.quantile(df[column], 0.75)

  lower_threshold = q1 - 1.5 * stats.iqr(df[column])

  upper_threshold = q3 + 1.5 * stats.iqr(df[column])

  print(f'Values below IQR lower threshold: {(df[column] < lower_threshold).sum()}')

  print(f'Values above IQR upper threshold: {(df[column] > upper_threshold).sum()}')
```

I then calculated the z-scores of each column and added them to the DataFrame:

```
  name = 'Z_Score_' + column
  df[name] = stats.zscore(df[column])
```

I then plotted all columns without NaN values with histograms:

```
  if np.isnan(stats.zscore(df[column])).all() == False:
    plt.hist(df[name])
    plt.title(column)
    plt.show()
```

I then generated a set of boxplots for each quantitative column.

```
  boxplot=sns.boxplot(x=column, data=df)
  plt.show()
```

Re-Expression of Categorical Variables

To determine which categorial variables would need to be re-expressed, I created a report of the values present in each of the columns that could be ordinally encoded:

```
#Import the libraries I'll be using
import pandas as pd


#Read in the data file
df = pd.read_csv('medical_raw_data.csv')


#Compose array of columns that will be coded as ordinal
ordinal_columns = ['Education', 'Complication_risk', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5',
'Item6', 'Item7', 'Item8', 'ReAdmis', 'Soft_drink',
          'HighBlood', 'Stroke', 'Overweight', 'Arthritis', 'Diabetes', 'Hyperlipidemia', 'BackPain',
'Anxiety', 'Allergic_rhinitis',
          'Reflux_esophagitis', 'Asthma']


#Loop through columns in ordinal_columns and print the range of values
for column in df[ordinal_columns]:
```

I used the df.column.unique() function to print an array of values present within the column.

print(df[column].unique())

B2. Justification for Assessing Data Quality

Duplicates

The duplicated() function provides a series of Boolean values. True, if a row is duplicated. False if a row is not duplicated. I used the value_counts() function on the returned series to return a count of each value category, in this case "True" or "False".

Missing Values

Using the df.isnull().sum() function gave me the count of every missing value in each column within the set. By calculating the percentage using df.isnull().sum() * 100 / len(df) I was able to relate the count of missing values to the overall percentage they represent for the total records in their column.

I wanted to get a more visual representation of the breakdown of the set. This is where I brough in the Missingno matrix() function and the PyPlot plt function.

Outliers

I created an array of each quantitative column. This allowed me to iterate over each in a loop without having to write each instruction separately.

quantitative_columns = ['Population', 'Children', 'Age', 'Income', 'VitD_levels', 'Doc_visits', 'Full_meals_eaten', 'VitD_supp', 'Initial_days', 'TotalCharge', 'Additional_charges']

for column in quantitative_columns:

In order to get a sense of the scale of the set described in the column, I calculated and printed the minimum, maximum, and range of values in each column:

#Print min, max, and range of values present

print(f'Minimum value: {df[column].min()}')

print(f'Maximum value: {df[column].max()}')

print(f'Value range: {df[column].max() - df[column].min()}')

I wanted to get a count of all values within each column that featured z-scores above 3 and below

-3, which is a potential indicator for outliers.

print(f'Values with a z-score over 3: {sum(stats.zscore(df[column]) > 3)}')

print(f'Values with a z-score under -3: {sum(stats.zscore(df[column]) < -3)}')

For the same reason I generated the boxplots, I wanted to get a picture of values or ranges whose significance might be missed with the z-score and box-plot analysis. To this end, I created a count of those values that fell below or above the range described below.

(Q1 – 1.5 * IQR) – (Q3 + 1.5 * IQR)

```
q1 = np.quantile(df[column], 0.25)
  q3 = np.quantile(df[column], 0.75)
  lower_threshold = q1 - 1.5 * stats.iqr(df[column])
  upper_threshold = q3 + 1.5 * stats.iqr(df[column])
  print(f'Values below IQR lower threshold: {(df[column] < lower_threshold).sum()}')
  print(f'Values above IQR upper threshold: {(df[column] > upper_threshold).sum()}')
```

In order to plot the z-scores, I created a new column for each within the existing Data Frame and assigned the respective z-score for each value in the column to the new column's fields.

```
  name = 'Z_Score_' + column
  df[name] = stats.zscore(df[column])
```

As I had not yet removed missing or null values, I was getting an error when these values were plotted with a histogram. To circumvent this, I printed histograms only for the columns that did not feature these values. This is code that will be corrected when the appropriate procedures for cleaning missing and null values are met.

```
  if np.isnan(stats.zscore(df[column])).all() == False:
    plt.hist(df[name])
    plt.title(column)
    plt.show()
```

In order to assess columns that were missed by the histograms, and to get a better visual understanding of outliers in the set, I also generated a box plot for each column.

```
  boxplot=sns.boxplot(x=column, data=df)
  plt.show()
```

Re-Expression of Categorical Variables

Variables included in the set that feature dichotomous values, while not ordinal by definition, can be treated with ordinal encoding, so I included them in my set.

```
#Import the libraries I'll be using
import pandas as pd


#Read in the data file
```

```
df = pd.read_csv('medical_raw_data.csv')


#Compose array of columns that will be coded as ordinal

ordinal_columns = ['Education', 'Complication_risk', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5',
'Item6', 'Item7', 'Item8', 'ReAdmis', 'Soft_drink',

            'HighBlood', 'Stroke', 'Overweight', 'Arthritis', 'Diabetes', 'Hyperlipidemia', 'BackPain',
'Anxiety', 'Allergic_rhinitis',

            'Reflux_esophagitis', 'Asthma']


#Loop through columns in ordinal_columns and print the range of values

for column in df[ordinal_columns]:
```

In order to visualize the range of values present in the ordinally coded columns, I used the unique function to print all unique values present in the columns.

```
  print(f'{column} : {df[column].unique()})
```

B3. Justification of Python and Libraries

Python

Coming from a software engineering background, I have extensive experience in Python. Python works intuitively with the way that I think and solve programming problems. It is a fast and efficient language, and the libraries and packages available make statistical analysis in Python relatively smooth and easy.

Pandas

Data Frames in Pandas are indispensable to the data analysis process. The built in functions that allow you to read, manipulate, and write data are standouts. The code for detecting and mitigating both duplicates and missing values is entirely dependent on the package, and the other processes performed in this analysis were either enabled or assisted by its inclusion.

NumPy

NumPy's numeric functions were critical in evaluating NaN values and performing more complex numeric operations.

PyPlot

PyPlot was essential in generating basic plots such as the histograms included in the analysis. For the packages designed to generate more niche or dedicated plots, PyPlot was still essential in displaying the resulting plots.

SciPy

SciPy was primarily used for its zscore() function, which can be given a column and return a column populated with all of the z-scores from the original column. With this, I was able to determine ranges of z-scores and plot them in histograms.

Seaborn

Seaborn was used for the generation of boxplots. These boxplots were necessary to the analysis as they were able to visually represent the distribution and significance of outliers within the set.

MissingNo

MissingNo was used to generate a missing data matrix. This matrix created a visual representation of missing values within the columns of the entire set. This gave a broad overview of the impact of these missing values beyond what a simple count or percentage would provide.

C. Data Cleaning Summary

C1. Findings for Data Quality Issues

Duplicates

The results of my detection of duplicates was as follows, with 'True' meaning that duplicate records were found and 'False' indicating that they were not:

```
False    10000
Name: count, dtype: int64
```

As seen above, there were no duplicate records discovered within the set.

Missing Values

Below is the result of my detection of missing values:

```
                    Missing Count  Missing Percent
Unnamed: 0                      0             0.00
CaseOrder                       0             0.00
Customer_id                     0             0.00
Interaction                     0             0.00
UID                             0             0.00
City                            0             0.00
State                           0             0.00
County                          0             0.00
Zip                             0             0.00
Lat                             0             0.00
Lng                             0             0.00
Population                      0             0.00
Area                            0             0.00
Timezone                        0             0.00
Job                             0             0.00
Children                     2588            25.88
Age                          2414            24.14
Education                       0             0.00
Employment                      0             0.00
Income                       2464            24.64
Marital                         0             0.00
```

```
Gender                     0           0.00
ReAdmis                    0           0.00
VitD_levels                0           0.00
Doc_visits                 0           0.00
Full_meals_eaten           0           0.00
VitD_supp                  0           0.00
Soft_drink              2467          24.67
Initial_admin              0           0.00
HighBlood                  0           0.00
Stroke                     0           0.00
Complication_risk          0           0.00
Overweight               982           9.82
Arthritis                  0           0.00
Diabetes                   0           0.00
Hyperlipidemia             0           0.00
BackPain                   0           0.00
Anxiety                  984           9.84
Allergic_rhinitis          0           0.00
Reflux_esophagitis         0           0.00
Asthma                     0           0.00
Services                   0           0.00
Initial_days            1056          10.56
TotalCharge                0           0.00
Additional_charges         0           0.00
Item1                      0           0.00
Item2                      0           0.00
Item3                      0           0.00
Item4                      0           0.00
Item5                      0           0.00
Item6                      0           0.00
Item7                      0           0.00
Item8                      0           0.00
```

The columns with missing values were:

Children – 2,588 missing values accounting for 25.88% of the set.

Age – 2,414 missing values accounting for 24.14% of the set.

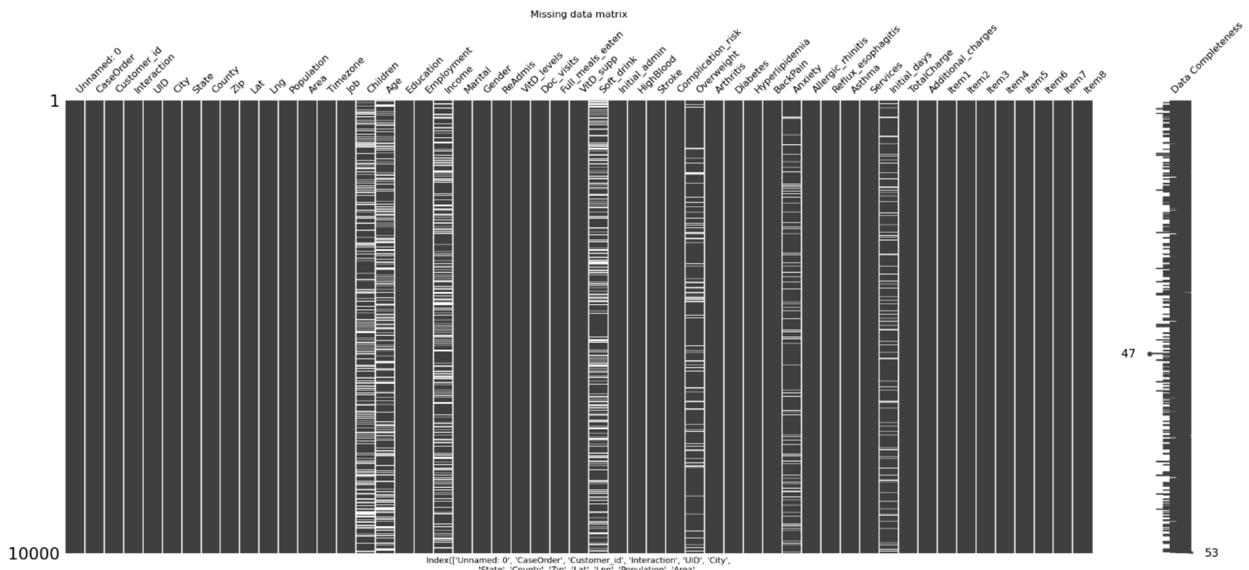
Income – 2,464 missing values accounting for 24.64% of the set.

Overweight – 982 missing values accounting for 9.82% of the set.

Anxiety – 984 missing values accounting for 9.84% of the set.

Initial_days – 1056 missing values accounting for 10.56% of the set.

In addition to the counts and percentages, I generated a missing number matrix to better visualize the distribution of the missing values.

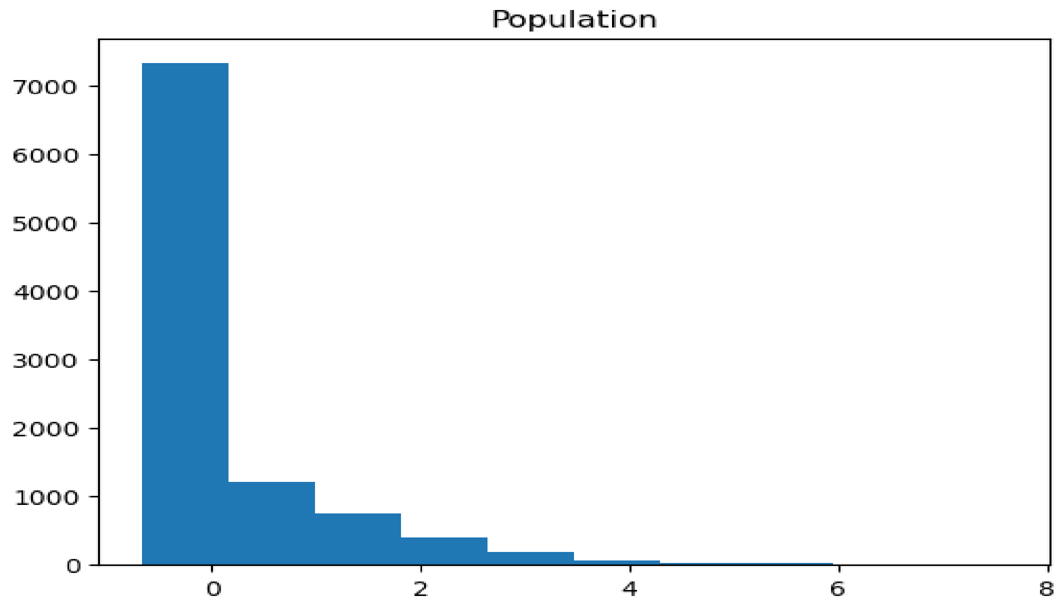


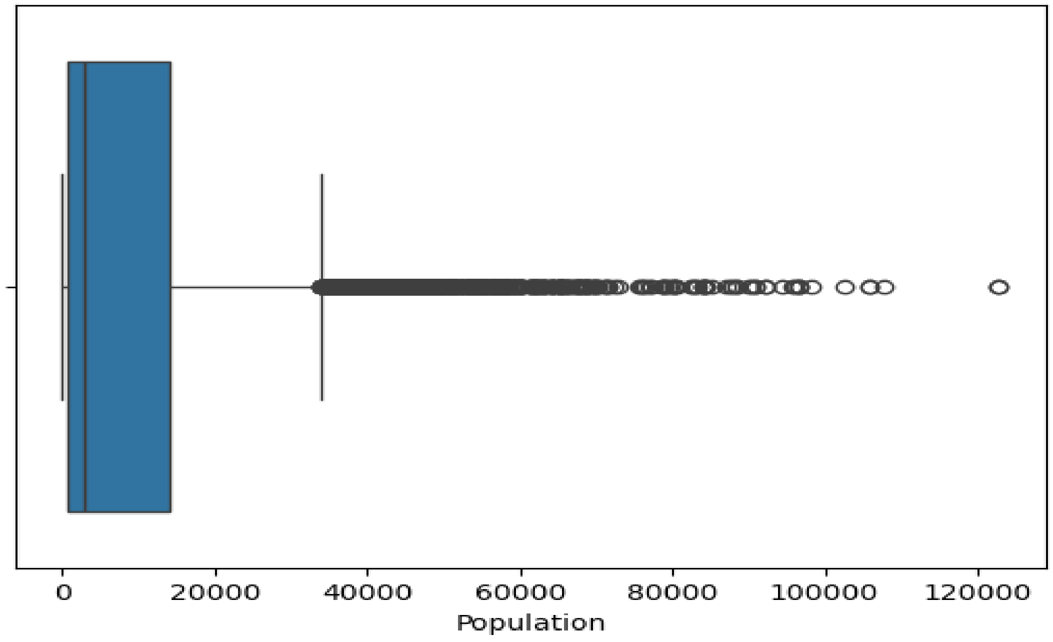
```
Index(['Unnamed: 0', 'CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City',
       'State', 'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area',
       'Timezone', 'Job', 'Children', 'Age', 'Education', 'Employment',
       'Income', 'Marital', 'Gender', 'ReAdmis', 'VitD_levels', 'Doc_visits',
       'Full_meals_eaten', 'VitD_supp', 'Soft_drink', 'Initial_admin',
       'HighBlood', 'Stroke', 'Complication_risk', 'Overweight', 'Arthritis',
       'Diabetes', 'Hyperlipidemia', 'BackPain', 'Anxiety',
       'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma', 'Services',
       'Initial_days', 'TotalCharge', 'Additional_charges', 'Item1', 'Item2',
       'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8'],
      dtype='object')
```

Outliers

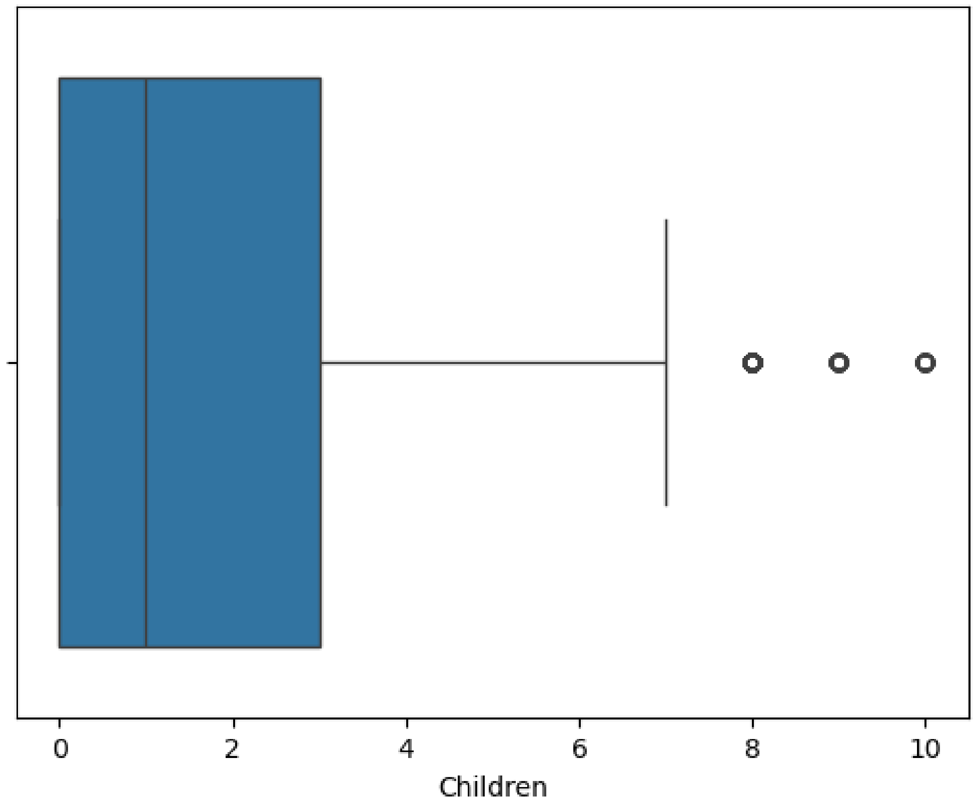
Below are the results of my outlier analysis:

```
++++=========Population========++++
Minimum value: 0
Maximum value: 122814
Value range: 122814
Values with a z-score over 3: 218
Values with a z-score under -3: 0
Values below IQR lower threshold: 0
Values above IQR upper threshold: 855
```

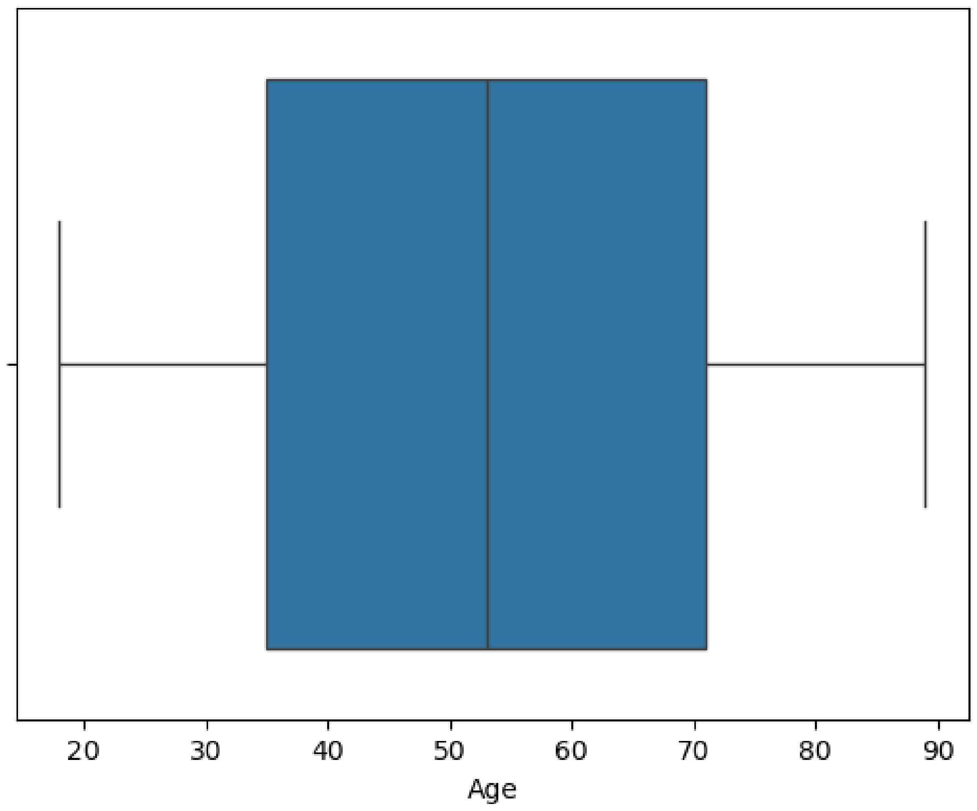




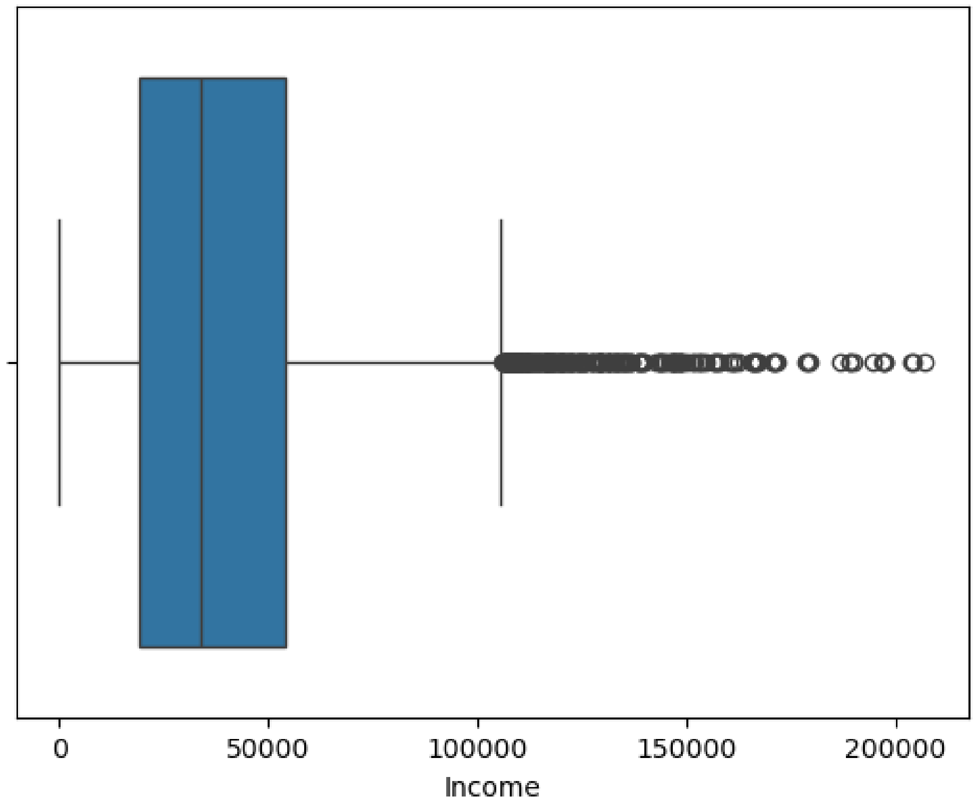
```
++++=========Children========++++
Minimum value: 0.0
Maximum value: 10.0
Value range: 10.0
Values with a z-score over 3: 0
Values with a z-score under -3: 0
Values below IQR lower threshold: 0
Values above IQR upper threshold: 0
```



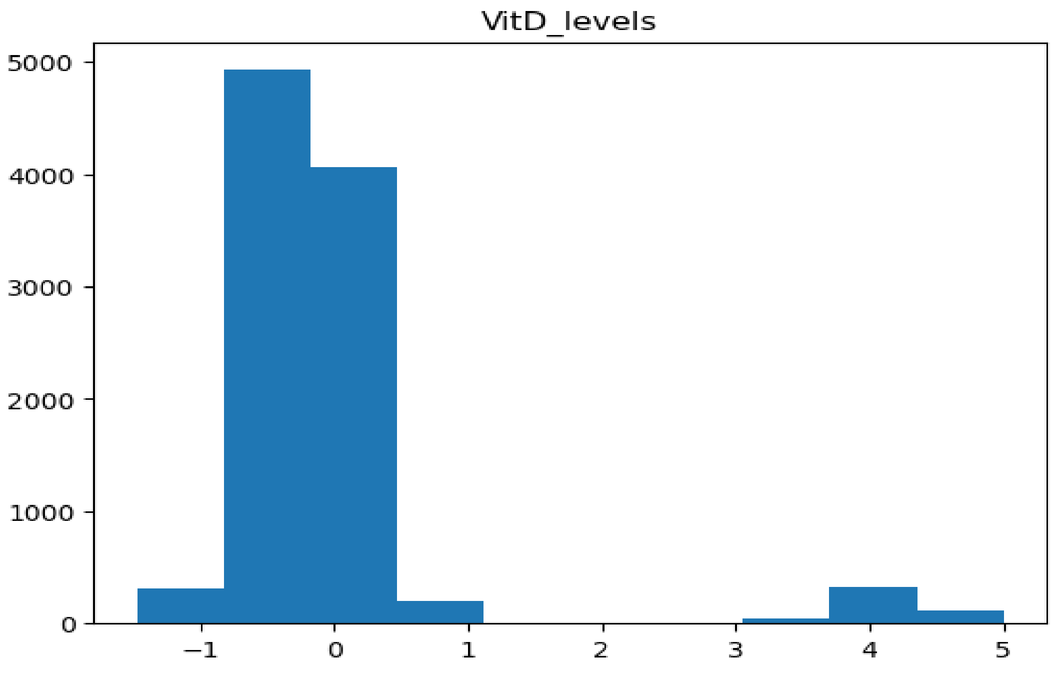
```
++++=========Age========++++

Minimum value: 18.0
Maximum value: 89.0
Value range: 71.0
Values with a z-score over 3: 0
Values with a z-score under -3: 0
Values below IQR lower threshold: 0
Values above IQR upper threshold: 0
```

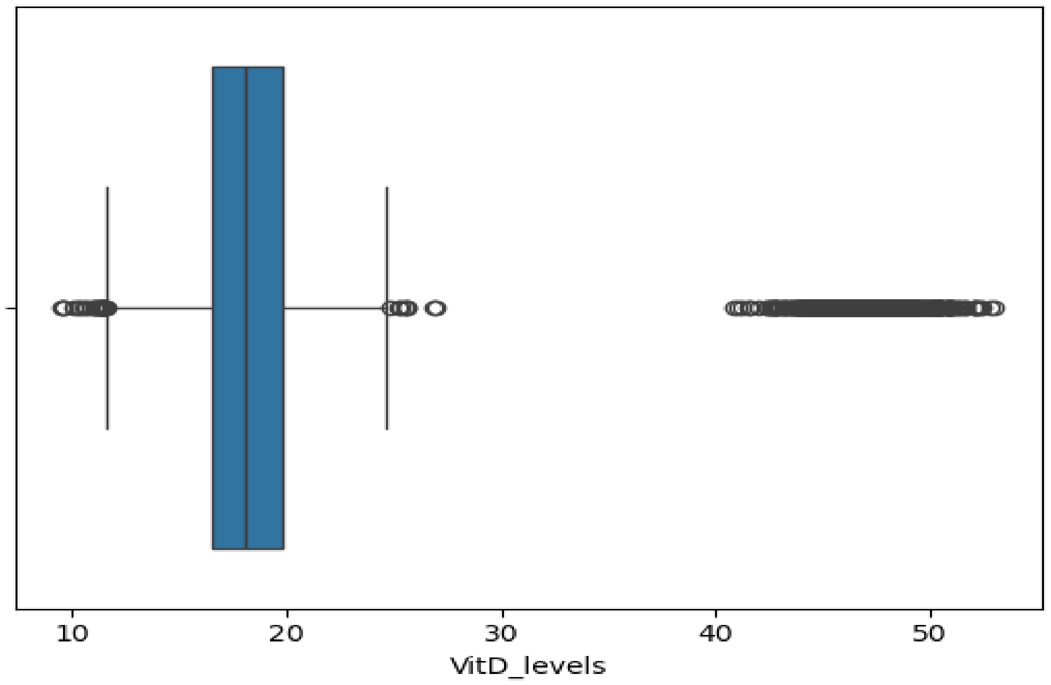


```
++++=========Income========++++
Minimum value: 154.08
Maximum value: 207249.13
Value range: 207095.05000000002
Values with a z-score over 3: 0
Values with a z-score under -3: 0
Values below IQR lower threshold: 0
Values above IQR upper threshold: 0
```

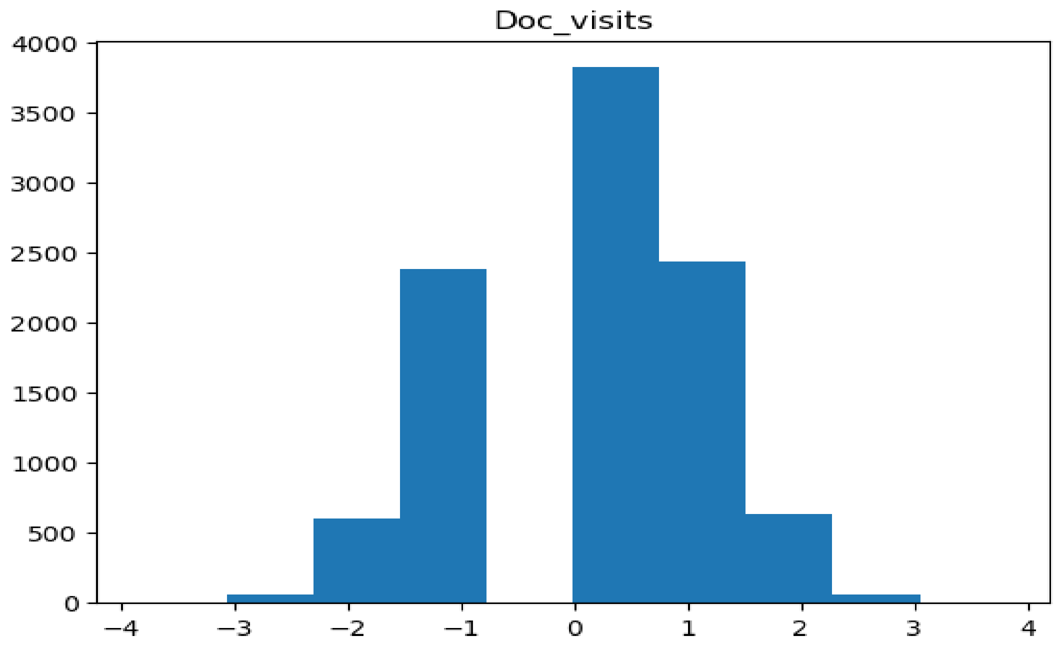


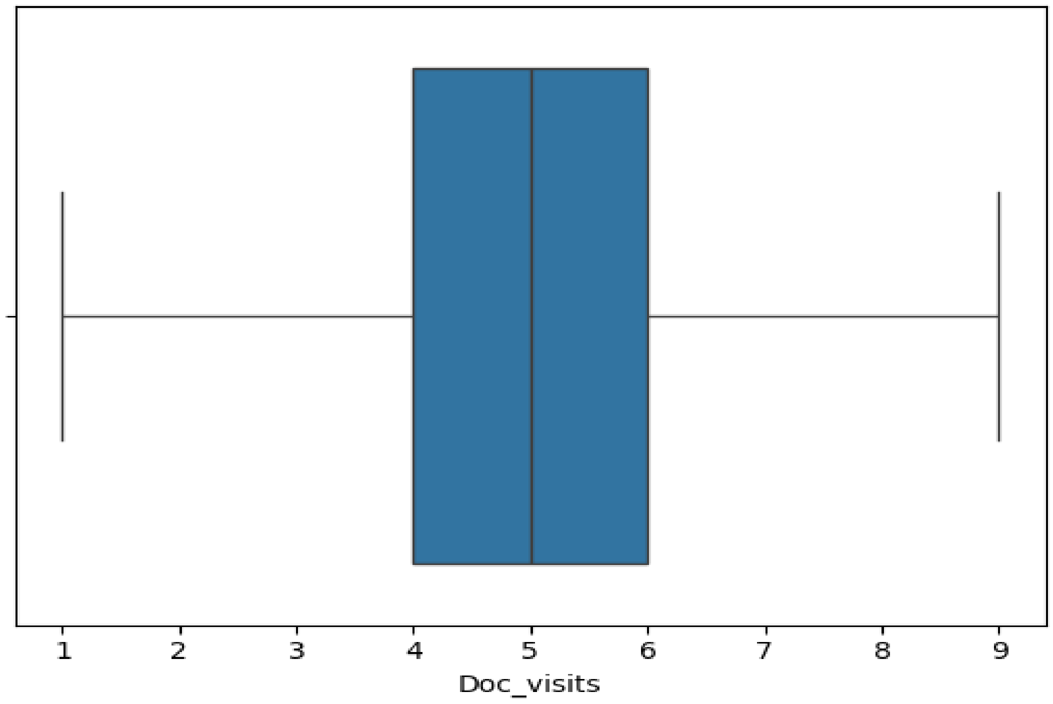
```
++++=========VitD_levels========++++
Minimum value: 9.519011638
Maximum value: 53.01912416
Value range: 43.500112521999995
Values with a z-score over 3: 500
Values with a z-score under -3: 0
Values below IQR lower threshold: 26
Values above IQR upper threshold: 508
```



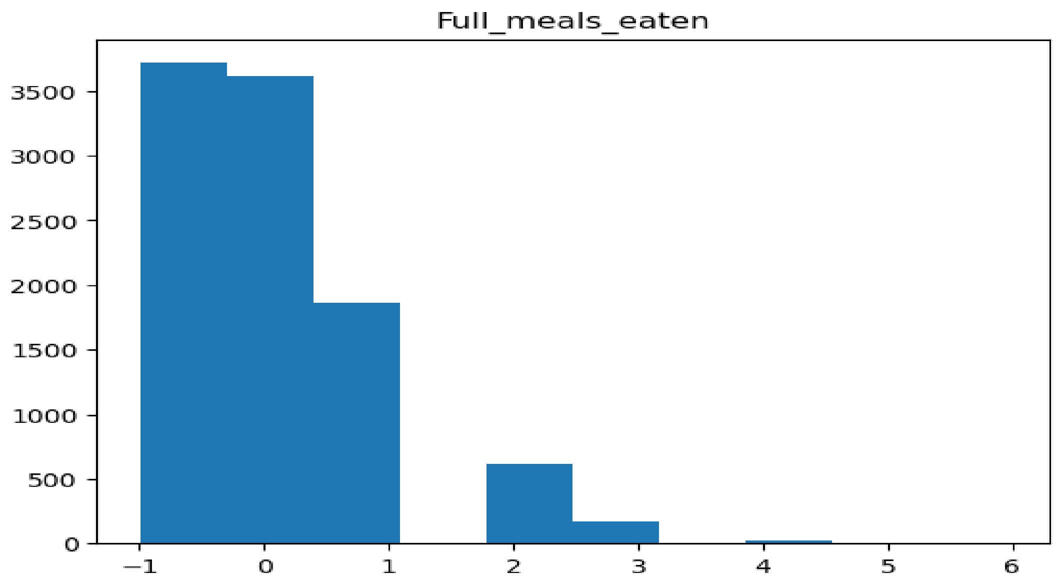


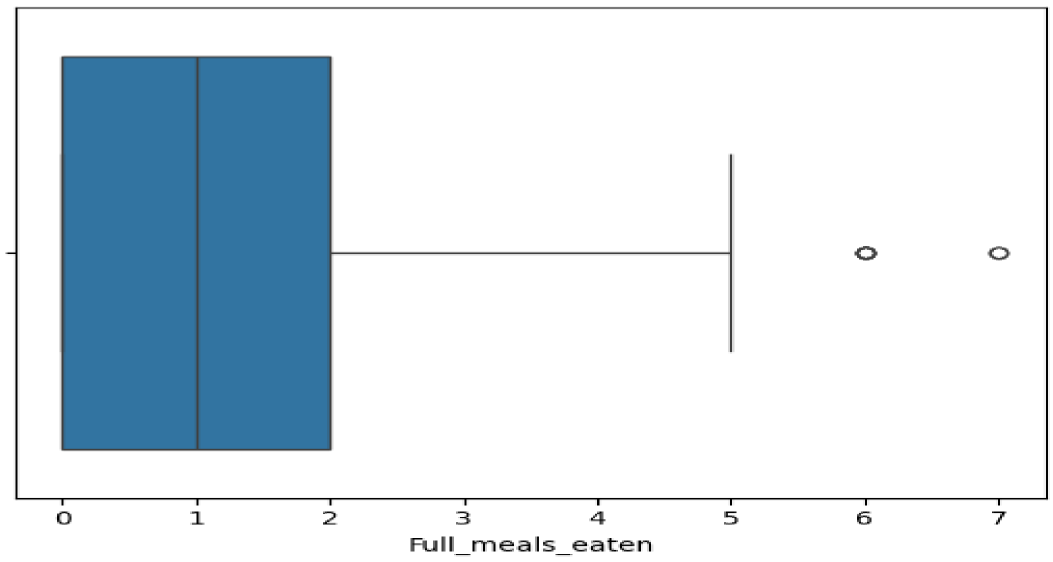
```
++++=========Doc_visits========++++
Minimum value: 1
Maximum value: 9
Value range: 8
Values with a z-score over 3: 2
Values with a z-score under -3: 6
Values below IQR lower threshold: 0
Values above IQR upper threshold: 0
```



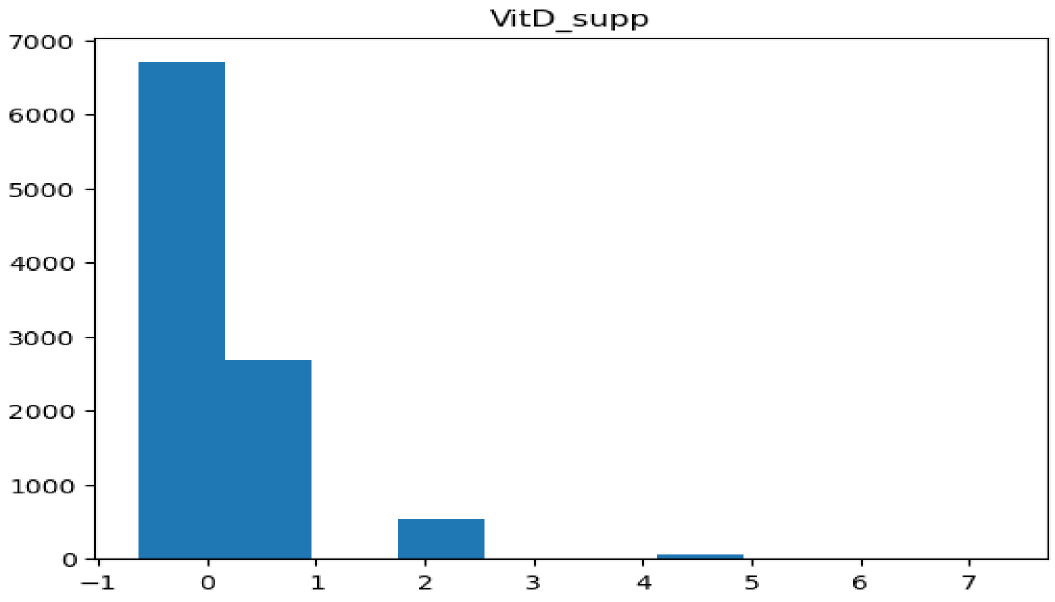


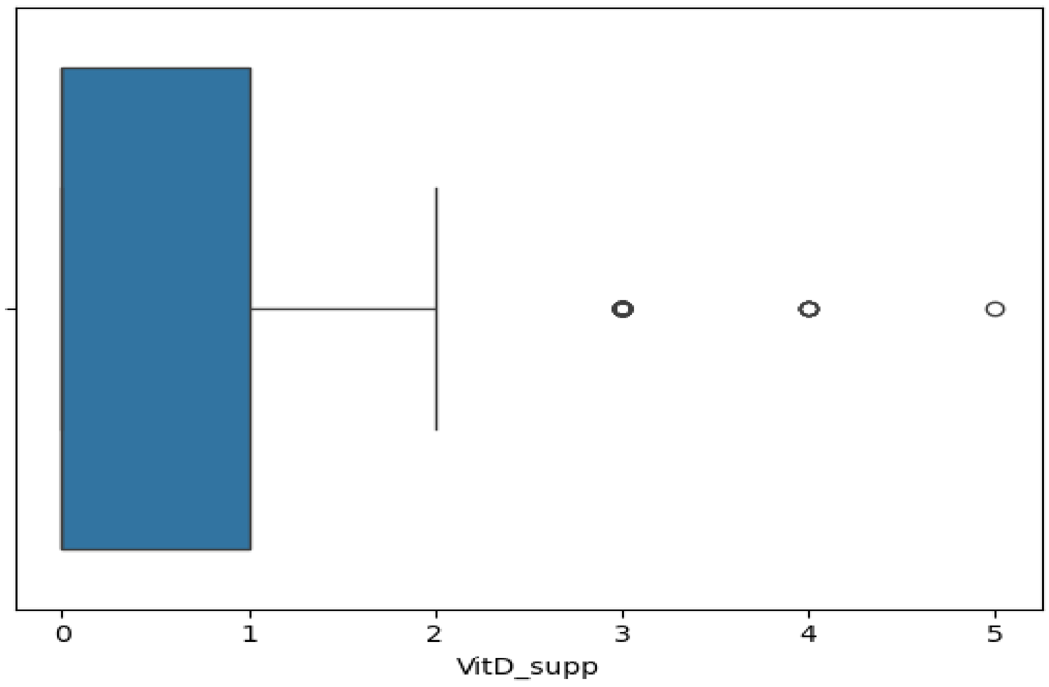
```
++++=========Full_meals_eaten========++++
Minimum value: 0
Maximum value: 7
Value range: 7
Values with a z-score over 3: 33
Values with a z-score under -3: 0
Values below IQR lower threshold: 0
Values above IQR upper threshold: 8
```



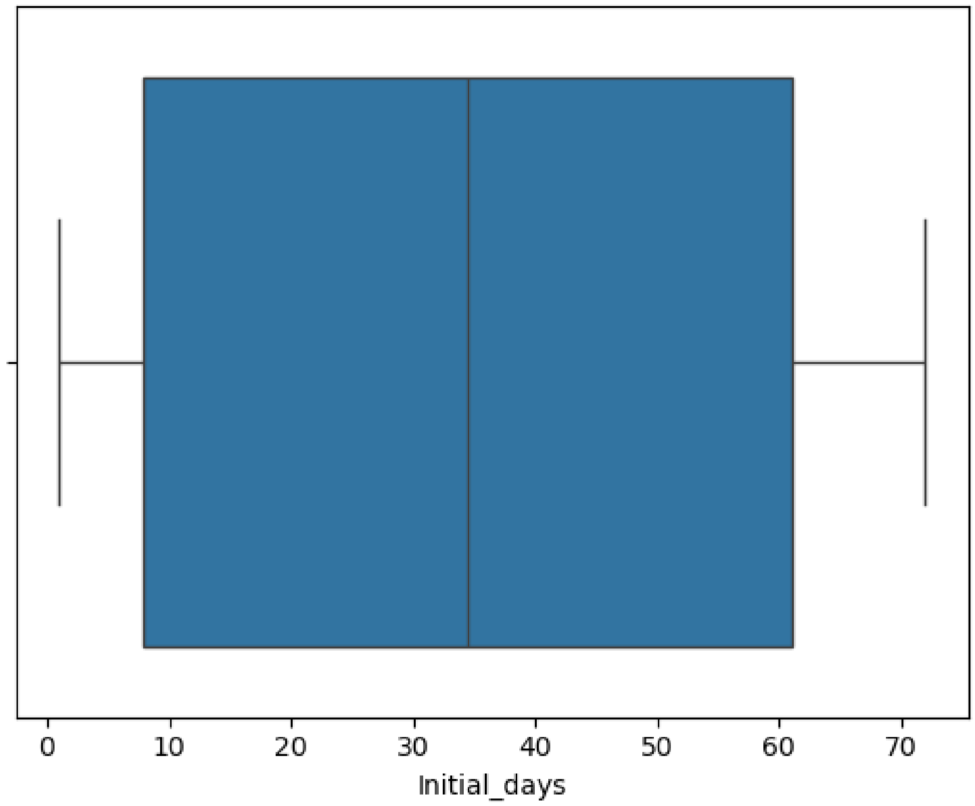


```
++++=========VitD_supp========++++
Minimum value: 0
Maximum value: 5
Value range: 5
Values with a z-score over 3: 70
Values with a z-score under -3: 0
Values below IQR lower threshold: 0
Values above IQR upper threshold: 70
```

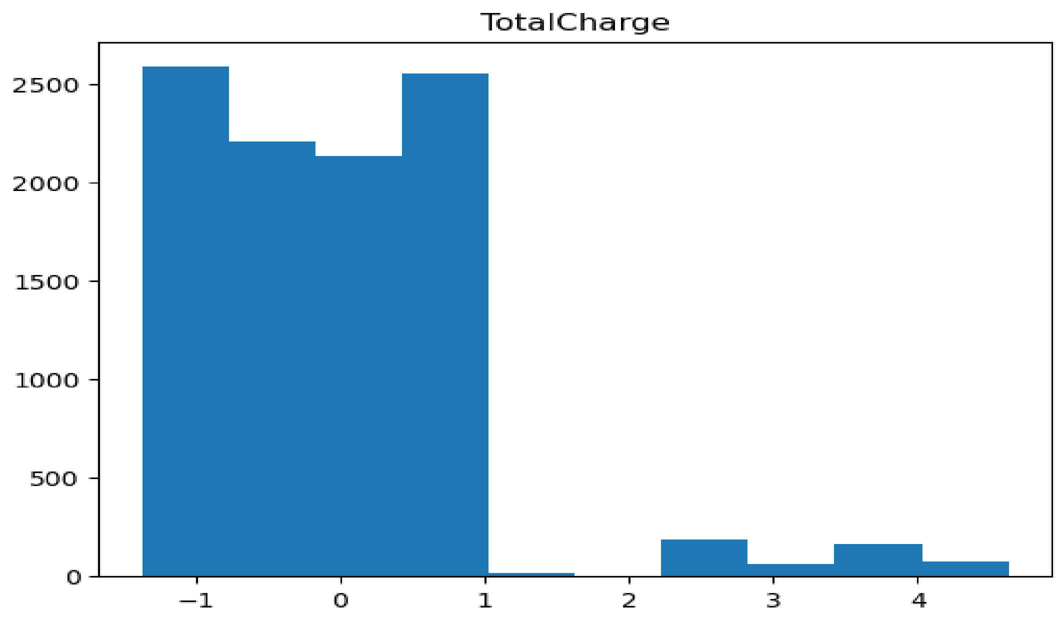


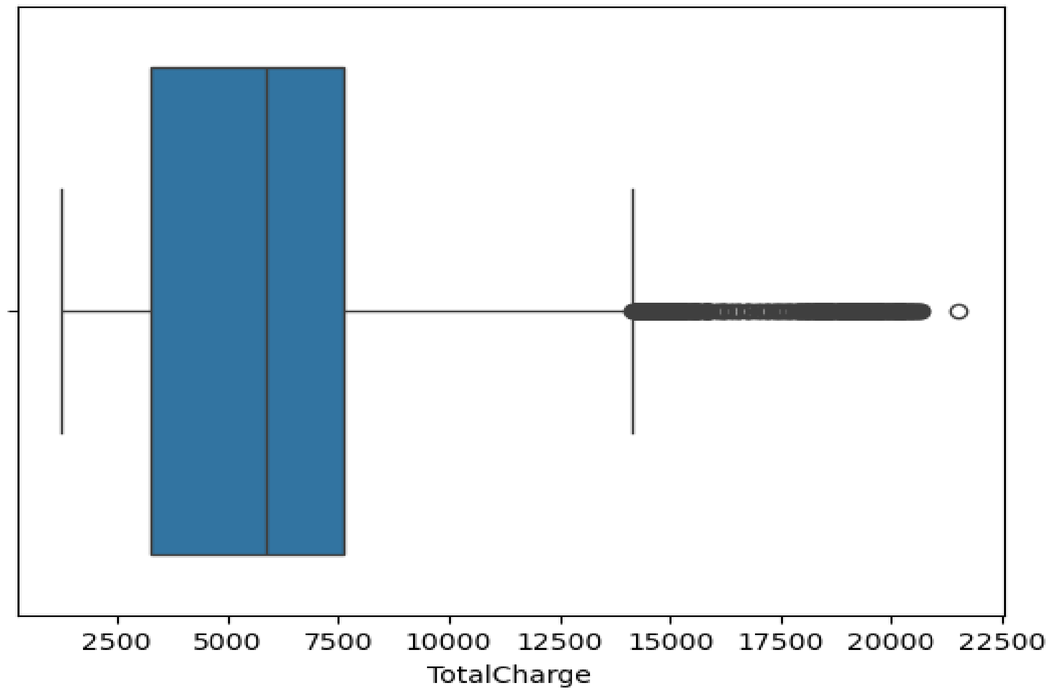


```
++++=========Initial_days========++++
Minimum value: 1.001980919
Maximum value: 71.9814863
Value range: 70.979505381
Values with a z-score over 3: 0
Values with a z-score under -3: 0
Values below IQR lower threshold: 0
Values above IQR upper threshold: 0
```

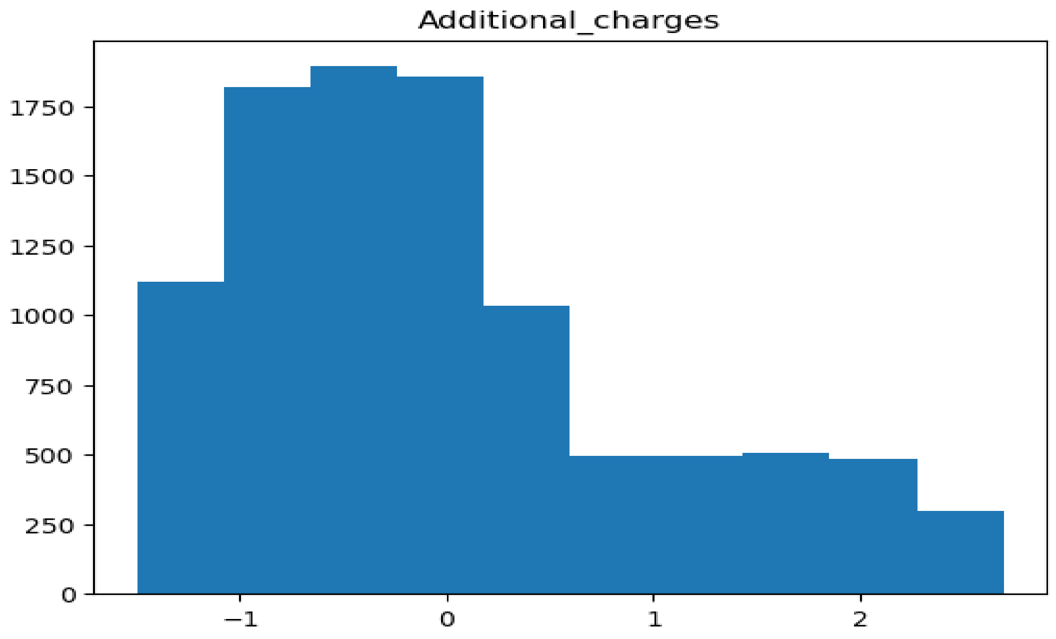


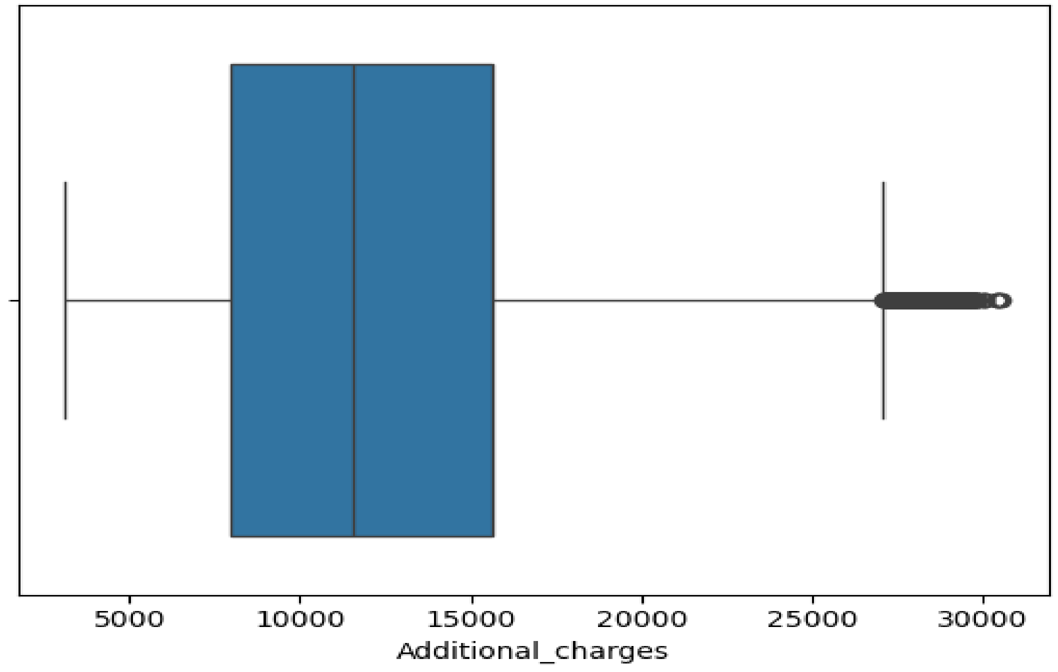
```
++++=========TotalCharge========++++
Minimum value: 1256.751699
Maximum value: 21524.22421
Value range: 20267.472511
Values with a z-score over 3: 276
Values with a z-score under -3: 0
Values below IQR lower threshold: 0
Values above IQR upper threshold: 466
```





```
++++=========Additional_charges========++++
Minimum value: 3125.702716
Maximum value: 30566.07313
Value range: 27440.370414
Values with a z-score over 3: 0
Values with a z-score under -3: 0
Values below IQR lower threshold: 0
Values above IQR upper threshold: 424
```





Potential candidates for outliers include those values that express values that:

1. Feature a z-score above 3 or below -3
2. Feature a value that is outside of the range: (Q1 – 1.5 * IQR) – (Q3 + 1.5 * IQR)
3. Generate a histogram with visual markers (isolated columns)
4. Generate a boxplot with visual markers (values outside of the bounds of the outer edges of the plot)

The variables displaying signs of containing outliers were:

Population:

- 218 values with a z-score above 3
- 855 values above IQR upper threshold
- Visual markers include a large number of values to the right of the plot edge

Children

- NaN: To do
- Visual markers include several values to the right of the plot edge

Age

- No z-scores found above or below threshold
- No values above or below IQR threshold
- No visual indicators of values outside of norm

Income

- Nan: To do
- Visual markers include many values to the right of the plot edge

VitD_levels

- 500 values above z-score threshold
- 26 values below IQR threshold
- 508 values above IQR threshold
- Visual indicators in the histogram feature isolated columns
- Visual indicators in the boxplot features values to the left and right of the plot edges

Doc_visits

- 2 values below the z-score threshold
- 6 values above the z-score threshold
- No values above or below the IQR threshold
- No visual indicators of values outside of norm

Full_meals_eaten

- 33 values below the z-score threshold
- 8 values found above the IQR threshold
- Visual indicators in the histogram include isolated columns
- Visual indicators in the box-plot include values to the right of the plot edge

VitD_supp

- 70 values above the z-score threshold
- 70 values above the IQR threshold
- Visual indicators in the histogram include isolated columns
- Visual indicators in the box-plot include values to the right of the plot edge

Initial_days

- NaN to do
- No visual indicators found in box-plot

TotalCharge

- 276 values above the z-score threshold
- 466 values above the IQR threshold
- Visual indicators in the histogram include isolated columns
- Visual indicators in the box-plot include many values to the right of the plot edge

Additional_charges

- No values found above or below z-score threshold
- 424 values found above IQR threshold
- Visual indicators in the histogram include many values to the right of the plot edge

Re-Expression of Categorical Variables

Below are the results of my analysis of categorical variables:

```
Education : ['Some College, Less than 1 Year'
 'Some College, 1 or More Years, No Degree'
 'GED or Alternative Credential' 'Regular High School Diploma'
 "Bachelor's Degree" "Master's Degree" 'Nursery School to 8th Grade'
 '9th Grade to 12th Grade, No Diploma' 'Doctorate Degree'
 "Associate's Degree" 'Professional School Degree'
 'No Schooling Completed']
Complication_risk : ['Medium' 'High' 'Low']
Item1 : [3 2 4 1 5 7 6 8]
Item2 : [3 4 5 1 2 6 7]
Item3 : [2 3 4 5 1 6 7 8]
Item4 : [2 4 3 5 6 1 7]
Item5 : [4 3 5 2 6 1 7]
Item6 : [3 4 5 2 6 1 7]
Item7 : [3 5 4 2 6 1 7]
Item8 : [4 3 5 6 2 1 7]
ReAdmis : ['No' 'Yes']
Soft_drink : [nan 'No' 'Yes']
HighBlood : ['Yes' 'No']
Stroke : ['No' 'Yes']
Overweight : [ 0.  1. nan]
Arthritis : ['Yes' 'No']
Diabetes : ['Yes' 'No']
Hyperlipidemia : ['No' 'Yes']
BackPain : ['Yes' 'No']
Anxiety : [ 1. nan  0.]
Allergic_rhinitis : ['Yes' 'No']
Reflux_esophagitis : ['No' 'Yes']
Asthma : ['Yes' 'No']
```

What I found was an inconsistency between the coding of dichotomous values i.e. 'Yes' or 'No' vs '0' or '1'. There was also an inconsistency in the values represented by the rankings of ordinal variables i.e. 'Low', 'Medium', 'High' vs. '1', '2', '3'.

C2. Justification for Mitigation Methods

Duplicates

No duplicates were found, so no mitigation was necessary.

Missing Values

For all variables in the set, imputation was found to be the most appropriate mitigations technique. No individual column had a percentage of missing values that rose above the upper limit of 60%, and therefore none were candidates for removal. Likewise, all missing values for each column represented more than 5% of the set, so removal of records with missing data was not optimal either.

My process for each variable in the set was:

1. Determine type (Numeric, Categorical)

2. Examine distribution
3. Choose imputation method based on type and distribution
4. Apply imputation
5. Verify distribution did not significantly change during imputation

For all variables in the set, univariate imputation methods proved to be appropriate.

Age

Age was determined to be a numeric value, meaning the choice for imputation came down to either median or mean. Upon examination, Age had shown to display a uniform distribution. For this reason, I chose the imputation method of mean. Below is the code I ran to perform the mitigation.

```
#Examine mean and median for age before imputation

print('++++----Age Before Imputation----++++')

print(f'Mean: {df['Age'].mean()}')

print(f'Median: {df['Age'].median()}')

print('++++--------++++')


#Fill age with mean

df.fillna({'Age' : int(df['Age'].mean())}, inplace = True)


#Examine mean and median for age after imputation

print('++++----Age After Imputation----++++')

print(f'Mean: {df['Age'].mean()}')

print(f'Median: {df['Age'].median()}')

print('++++--------++++')
```

Above and below the line I use the fillna() method are print statements that allow me to examine the mean and median of the value before and after imputation to ensure that the distribution did not change significantly. Below are the results of the statement.

```
++++----Age Before Imputation----++++
Mean: 53.29567624571579
Median: 53.0
++++--------++++
++++----Age After Imputation----++++
Mean: 53.2243
Median: 53.0
++++--------++++
```

Children, Income, Initial_days

Children, Income, and Initial_days were all determined to be numeric. Each of these variables exhibited signs of skewdness, therefore the median was the chosen imputation method. Below is the code I used to perform the imputation.

```
#Loop through each numeric column and fill with median value

for column in numeric_columns:

  print(f'++++----{column} Before Imputation----++++')

  print(f'Mean: {df[column].mean()}')

  print(f'Median: {df[column].median()}')

  print('++++--------++++')

  #Fill with median value

  df.fillna({column : int(df[column].median())}, inplace = True)

  print(f'++++----{column} After Imputation----++++')

  print(f'Mean: {df[column].mean()}')

  print(f'Median: {df[column].median()}')

  print('++++--------++++')
```

Like with Age, I included debug statements in the loop that allowed me to examine the mean and median before an after imputations to detect significant changes to the distribution of values. Below are the results for each of the three included variables.

```
++++----Children Before Imputation----++++
Mean: 2.0982191041554237
Median: 1.0
++++--------++++
++++----Children After Imputation----++++
Mean: 1.814
Median: 1.0
++++--------++++
++++----Income Before Imputation----++++
Mean: 40484.4382683121
Median: 33942.28
++++--------++++
++++----Income After Imputation----++++
Mean: 38872.381479
Median: 33942.0
++++--------++++
++++----Initial_days Before Imputation----++++
Mean: 34.432081888043044
Median: 34.44694129
++++--------++++
++++----Initial_days After Imputation----++++
Mean: 34.3864540406657
```

```
Median: 34.0
++++--------++++
```

Soft_drink, Overweight, Anxiety

Soft_drink, Overweight, and Anxiety were all found to be categorical values. Because of this, mode was determined to be the most appropriate imputation method for mitigating missing values. Below is the code I used to achieve this.

#Loop through each categorical column and fill with mode value

for column in categorical_columns:

  #Fill with mode

  df.fillna({column : df[column].mode()}, inplace = True)

When all of this was done, I used the following line to store the cleaned data into a new file titled 'medical_cleaned_data.csv':

#Export cleaned file to csv

df.to_csv('medical_cleaned_data.csv')

Outliers

The following variables were found to contain outliers:

Population

- Treatment: Retain
- Reason: Population ranges were determined to be reasonable. About 1.1% of values were found to be above the defined threshold. These values represent real data and imputing values would reduce the range and accuracy of the set.

Children

- Treatment: Imputation
- Reason: With the mean and median value hovering around 1.0, the maximum outlier value of 10.0 did feel outside of the bounds of the norm. Looking at the minimum outlier value of 7.0, we have a relatively narrow outlier range of 3.0. These values represent 0.0457% of the set, so I determined it would be appropriate to mitigate values above 7.0 by imputing them with the median using the code below:
    - ***def remove_outliers(column, minimum_accepted_value, maximum_accepted_value):***
    - ***df[column] = np.where(df[column] < minimum_accepted_value, np.nan, df[column])***
    - ***df[column] = np.where(df[column] > maximum_accepted_value, np.nan, df[column])***
    - ***df.fillna({column : int(df[column].median())}, inplace = True)***
    - ***remove_outliers('Children', 0, 7)***

Income

- Treatment: Retain
- Reason: The income ranges described (154.08 – 207,249.13 with a total range of 207,095.05) and the outlier ranges described (80,233.66 – 207249.13 with a total range of 127,015.47) were found to be statistically relevant for future analysis. The outlier range accounts for more than half of the total range of values. While these may seem extreme, there is nothing to suggest that these incomes fall outside of expected boundaries or are the result of an error.

VitD_levels

- Treatment: Retain
- Reason: This is an area where domain knowledge would be vital. I cannot determine which values of VitD_levels would appear to be erroneous. Purely from an analytical standpoint the range of values expressed does seem valid. There are no values so far beyond the norm that they would appear to be erroneous. Moreover, the cluster of values that appear beyond the bounds of the established thresholds may yield important insights for the analysis of factors impacting readmissions.

Full_meals_eaten

- Treatment: Retain
- Reason: The range of values is 0 – 7 with a range of outliers being 6 – 7. These outliers also represent a negligible percentage of the set. With the detected values being incredible close to the established normal range and the lack of occurrences of the outliers, other mitigation methods should not be necessary and would not make a significant difference on analysis.

VitD_supp

- Treatment: Retain
- Reason: This is another area where domain knowledge would be beneficial. The total range of values is 0 – 5 with outliers accounting for nearly half of that range (3 - 5). 0.007% of all values in the set were found to be within this range. Retention was the course of action taken because these values do not appear to be extreme or erroneous.

TotalCharge

- Treatment: Retain
- Reason: The minimum value of TotalCharge is 1,256.75. The maximum value is 21,524.22. These represent expected values given the cost of healthcare generally. No value is so far removed that it would suggest some kind of error.

Additional_charges

- Treatment: Retain
- Reason: The range of values with Additional_charges are similar to TotalCharge, with the minimum value being 3,125.70 and the maximum value being 30,566.07. These, like those

in TotalCharge, were deemed to be expected values, due to there being no indication of error, nor any belief that the values would not be expected.

Re-Expression of Categorical Variables

I elected to use ordinal encoding for the variables where I found inconsistent value expression. These were:

Education

I used the following code to map the ordinal Education column to corresponding numeric values.

#Separate the dichotomous variables from the rest

dichotomous_columns = ['ReAdmis', 'Soft_drink', 'HighBlood', 'Stroke', 'Arthritis', 'Diabetes', 'Hyperlipidemia',

'BackPain', 'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma']

#Create a numeric dictionary for the ordinal Education

dict = {'Education' : {'No Schooling Completed' : 0, 'Nursery School to 8th Grade' : 1, '9th Grade to 12th Grade, No Diploma' : 2,

'GED or Alternative Credential' : 3, 'Regular High School Diploma' : 4,

'Some College, Less than 1 Year' : 5, 'Some College, 1 or More Years, No Degree' : 6,

'Professional School Degree' : 7, "Associate's Degree" : 8, "Bachelor's Degree" : 9, "Master's Degree" : 10,

'Doctorate Degree' : 11}}

#Replace values with those from the dictionary

df.replace(dict, inplace=True)

This mapping allowed for the expression of the ordinal data as numeric, potentially assisting with future statistical analysis.

Complication_risk

I used the following code to map the ordinal Complication_risk to corresponding numeric values.

#Create a numeric dictionary for the ordinal Complication_risk

dict = {'Complication_risk' : {'Low' : 0, 'Medium' : 1, 'High' : 2}}

#Replace values with those from the dictionary

df.replace(dict, inplace=True)

Like with Education, this mapping allowed for the expression of the ordinal data as numeric, potentially assisting with future statistical analysis.

Remaining Variables

The remaining variables that were candidates for re-expression were:

- ReAdmis
- Soft_drink
- HighBlood
- Stroke
- Arthritis
- Diabetes
- Hyperlipidemia
- BackPain
- Allergic_rhinitis
- Reflux_esophagitis
- Asthma

Each of these represented a dichotomous “Yes”/”No” value in the data set. They were all treated the same way, mapping a “Yes” to a 1 and a “No” to a 0. Below is the code used to perform this step.

```
#Loop through dichotomous variables and give them a value of 0 for False and 1 for True

for column in dichotomous_columns:

  dict = {column : {'No' : 0, 'Yes' : 1}}

  df.replace(dict, inplace=True)
```

C3. Summary of Outcomes for Each Step

Duplicates

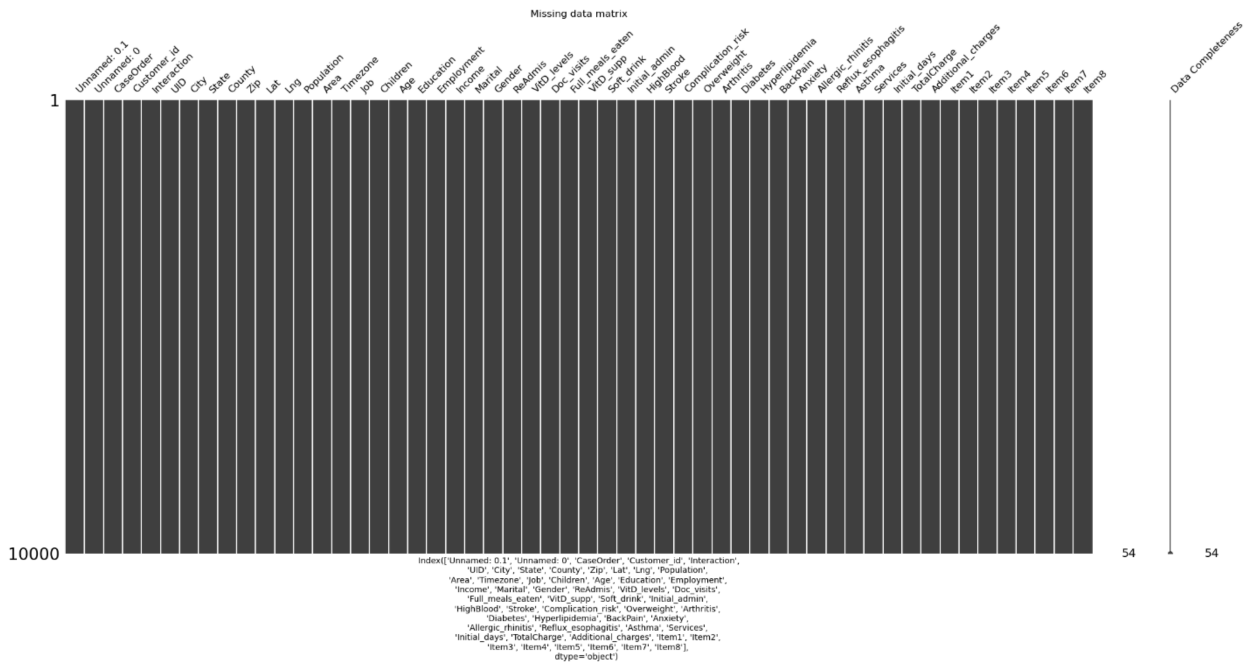
No duplicates were found, so no mitigation was necessary.

Missing Values

Following the mitigation process, examining the cleaned data using the previously stated detection methods yielded these results:

```
                    Missing Count  Missing Percent
Unnamed: 0.1                    0              0.0
Unnamed: 0                      0              0.0
CaseOrder                       0              0.0
Customer_id                     0              0.0
Interaction                     0              0.0
UID                             0              0.0
City                            0              0.0
State                           0              0.0
County                          0              0.0
Zip                             0              0.0
```

```
Lat                                0              0.0
Lng                                0              0.0
Population                         0              0.0
Area                               0              0.0
Timezone                           0              0.0
Job                                0              0.0
Children                           0              0.0
Age                                0              0.0
Education                          0              0.0
Employment                         0              0.0
Income                             0              0.0
Marital                            0              0.0
Gender                             0              0.0
ReAdmis                            0              0.0
VitD_levels                        0              0.0
Doc_visits                         0              0.0
Full_meals_eaten                   0              0.0
VitD_supp                          0              0.0
Soft_drink                         0              0.0
Initial_admin                      0              0.0
HighBlood                          0              0.0
Stroke                             0              0.0
Complication_risk                  0              0.0
Overweight                         0              0.0
Arthritis                          0              0.0
Diabetes                           0              0.0
Hyperlipidemia                     0              0.0
BackPain                           0              0.0
Anxiety                            0              0.0
Allergic_rhinitis                  0              0.0
Reflux_esophagitis                 0              0.0
Asthma                             0              0.0
Services                           0              0.0
Initial_days                       0              0.0
TotalCharge                        0              0.0
Additional_charges                 0              0.0
Item1                              0              0.0
Item2                              0              0.0
Item3                              0              0.0
Item4                              0              0.0
Item5                              0              0.0
Item6                              0              0.0
Item7                              0              0.0
Item8                              0              0.0
```



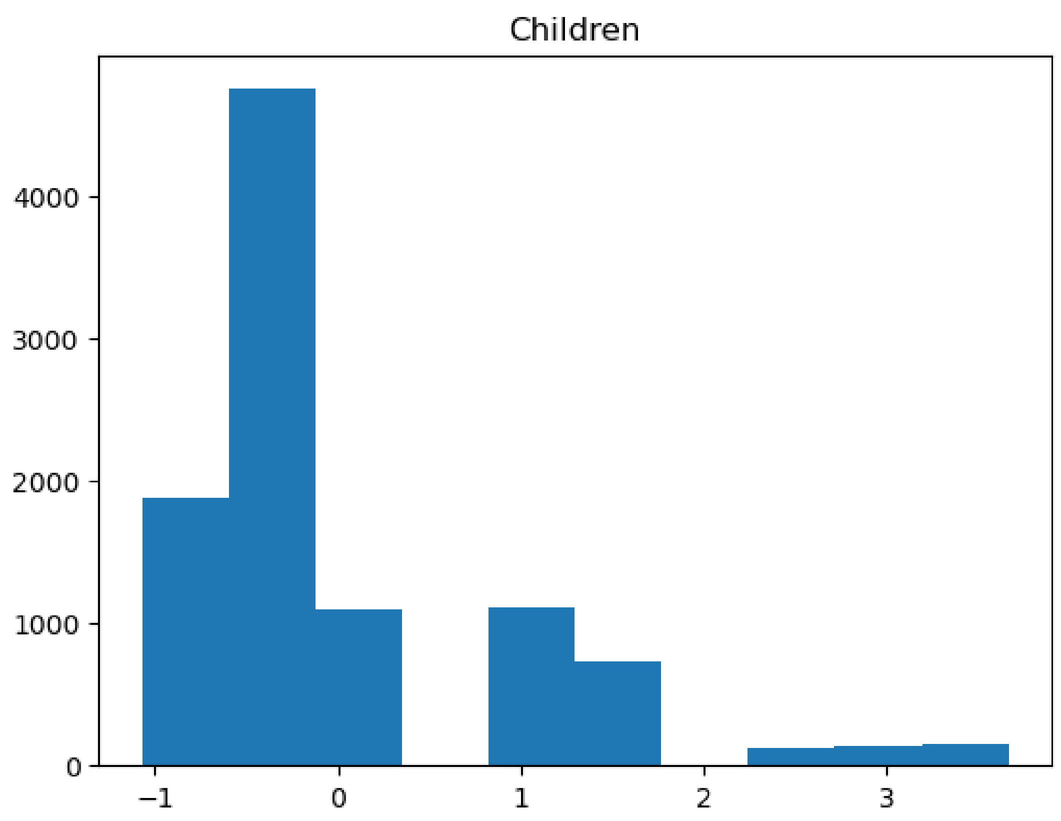
```
Index(['Unnamed: 0.1', 'Unnamed: 0', 'CaseOrder', 'Customer_id', 'Interaction',
       'UID', 'City', 'State', 'County', 'Zip', 'Lat', 'Lng', 'Population',
       'Area', 'Timezone', 'Job', 'Children', 'Age', 'Education', 'Employment',
       'Income', 'Marital', 'Gender', 'ReAdmis', 'VitD_levels', 'Doc_visits',
       'Full_meals_eaten', 'VitD_supp', 'Soft_drink', 'Initial_admin',
       'HighBlood', 'Stroke', 'Complication_risk', 'Overweight', 'Arthritis',
       'Diabetes', 'Hyperlipidemia', 'BackPain', 'Anxiety',
       'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma', 'Services',
       'Initial_days', 'TotalCharge', 'Additional_charges', 'Item1', 'Item2',
       'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8'],
      dtype='object')
```

Using the mitigation technique of imputation filled all of the missing values. Verification performed in the previous step showed no significant changes to the distribution of values within each of the variables.

Outliers

Following the mitigation method of filling values above 7.0 with the median for the Children column, my updated detection yielded these results:

```
++++=========Children========++++
Minimum value: 0.0
Maximum value: 7.0
Value range: 7.0
Mean Value: 1.581
Values with a z-score over 3: 154
Values with a z-score under -3: 0
Values below IQR lower threshold: 0
Values above IQR upper threshold: 1164
Minimum Outlier Value: 4.0
Maximum Outlier Value: 7.0
Range of Outlier Values: 3.0
```



While outliers still remain, they are within a reasonable and expected range.

Re-Expression of Categorical Variables

Following mitigation efforts, an examination of the unique variables in the treated tables appears as:

```
Education : [ 5  6  3  4  9 10  1  2 11  8  7  0]
Complication_risk : [1 2 0]
Item1 : [3 2 4 1 5 7 6 8]
Item2 : [3 4 5 1 2 6 7]
Item3 : [2 3 4 5 1 6 7 8]
Item4 : [2 4 3 5 6 1 7]
Item5 : [4 3 5 2 6 1 7]
Item6 : [3 4 5 2 6 1 7]
Item7 : [3 5 4 2 6 1 7]
Item8 : [4 3 5 6 2 1 7]
ReAdmis : [0 1]
Soft_drink : [0 1]
HighBlood : [1 0]
Stroke : [0 1]
Overweight : [0. 1.]
Arthritis : [1 0]
Diabetes : [1 0]
Hyperlipidemia : [0 1]
BackPain : [1 0]
Anxiety : [1. 0.]
```

```
Allergic_rhinitis : [1 0]
Reflux_esophagitis : [0 1]
Asthma : [1 0]
```

In the printing of the above variables, Overweight and Anxiety appear to feature decimal points in their expression. Upon examination of the data output, this was found not to be the case. The decimal points do not exist in the data.

C4. Limitations of the Data-Cleaning Process

Duplicates

While the included libraries have great built-in functions for detecting and removing duplicates, there are some potential drawbacks. The main one being that the duplicated() function only returns records that are exact matches. If a record was keyed with minor errors and a second record was created for the same patient due to this, the second record would not be detected by the duplicated() function, in spite of the fact that the two records could potentially be nearly identical.

Missing Values

Missing values in a data set pose a problem with a handful of imperfect solutions. One of the solutions is the removal of data with missing values. The problem here is that removing variables or rows limits both the sample size and scope of your analysis and the quality of your analysis suffers as a result. For this project, I was able to use imputation across the board, but there are issues with imputation as well.

Imputation comes down to being an inference of a missing value based on the existing values within the set. The problem is that imputed values may not accurately represent the population. Imputing large numbers of values could also skew the distribution and make your analysis less accurate. That is why it's vital to ensure that distribution does not change significantly after performing imputation.

Outliers

Of the four methods of mitigating outliers covered within the course: Impute, Retain, Exclude, and Remove, only two of these methods were found to be appropriate. These were Impute and Retain. Each method comes with its own share of pros and cons however. With imputation, bias can be introduced within the variable because the values are calculated from the remaining values, rather than occurring naturally. This method of "filling in the blanks" can distort the accuracy of the data.

The primary issue with retention is that certain statistical tests cannot be performed on variables that do not have a certain degree of normality. Retaining the data also retains the skew that comes with having a large number of outliers.

C5. Impact of Limitations on Analysis

There are some potential issues that could come from the way I approached the cleaning process for the data set. With outlier treatment, I decided to retain the vast majority of variables that displayed outliers. With these variables not following a normal distribution, there may be limitations in the analysis that can be performed on the variable. The skewness could present an issue when evaluating the relationship between variables.

Another issue could come from the way I mitigated the missing values. By nature, imputation is an attempt to fill gaps in data with values that do not skew the distribution or interpretation of that data. But the missing values themselves are part of a complete picture of the data set and inserting values does dilute the overall quality of the data. While it is easier to process, it is less reflective of the population.

D. Detection and Mitigation Code

D1. Annotated Code for Data Cleaning

See code attached – D206Task1.ipynb

D2. Cleaned Data Set

See file attached – medical_cleaned_data.csv

E. Principal Component Analysis (PCA)

E1. Principal Components and Loading Matrix

I used the following code to perform PCA and generate the loading matrix:

```
#Define variables in new data frame

variables = df[['Income', 'VitD_levels', 'TotalCharge', 'Additional_charges', 'Lat', 'Lng',
'Population', 'Children', 'Age', 'Doc_visits', 'Full_meals_eaten', 'VitD_supp', 'Initial_days']]


#Normalize variables

variables_normalized = (continuous_variables - continuous_variables.mean()) /
continuous_variables.std()


#Determine and print the number of principle components

pca = PCA(n_components = variables.shape[1])

print(pca)


#Perform PCA

pca.fit(variables_normalized)

continuous_pca = pd.DataFrame(pca.transform(variables_normalized), columns = ['PC1',
'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10', 'PC11', 'PC12', 'PC13'])


#Load PCs
```

loadings = pd.DataFrame(pca.components_.T, columns=['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10', 'PC11', 'PC12', 'PC13'], index = continuous_variables.columns)

#Print loading matrix

print(loadings)

Below is the generated loading matrix:

```
                          PC1       PC2       PC3       PC4       PC5  \
Income              -0.006431 -0.005553  0.044831  0.231205  0.351403
VitD_levels          0.540036 -0.052496 -0.056867 -0.299146  0.278389
TotalCharge          0.701946 -0.078146 -0.023375 -0.019051  0.001727
Additional_charges   0.083558  0.701028  0.001608  0.030573  0.000434
Lat                 -0.021694 -0.007934 -0.714375  0.143073 -0.025457
Lng                 -0.005398  0.017308  0.269484 -0.525099 -0.438038
Population           0.025505 -0.025444  0.630730  0.234582  0.245436
Children            -0.000287 -0.011304 -0.002193  0.158477 -0.250239
Age                  0.082970  0.701059  0.008127  0.018021 -0.025988
Doc_visits          -0.005388  0.012760  0.014048  0.185194  0.401867
Full_meals_eaten    -0.009317  0.036268 -0.105367 -0.390089  0.455684
VitD_supp            0.033960  0.010698  0.031817  0.424594 -0.064666
Initial_days         0.446437 -0.073839  0.032437  0.335178 -0.331099

                          PC6       PC7       PC8       PC9      PC10  \
Income               0.504639 -0.251608  0.534689  0.094697 -0.453546
VitD_levels          0.137056  0.055812 -0.024285 -0.462751  0.046076
TotalCharge          0.004339 -0.010335 -0.006499  0.011368 -0.013026
Additional_charges  -0.011880 -0.018216 -0.002541 -0.013204 -0.011823
Lat                  0.006952  0.015457 -0.063795 -0.037999 -0.027504
Lng                  0.511617  0.086948  0.009756  0.202336  0.035482
Population          -0.284251 -0.047844 -0.066792 -0.130648  0.042920
Children             0.173157 -0.785268 -0.010339 -0.254251  0.447165
Age                 -0.011918 -0.006490 -0.006016 -0.001668 -0.025766
Doc_visits           0.453557  0.005535 -0.696498  0.298348  0.143786
Full_meals_eaten    -0.208221 -0.156878  0.317195  0.473670  0.479350
VitD_supp            0.282001  0.523217  0.344246 -0.094512  0.576230
Initial_days        -0.158356 -0.089249  0.021744  0.574617 -0.069954

                         PC11      PC12      PC13
Income               0.048907 -0.007349 -0.001062
VitD_levels          0.001205 -0.023084  0.544148
TotalCharge          0.000611  0.021062 -0.706641
Additional_charges   0.020709  0.706221  0.025891
Lat                  0.679145 -0.014890 -0.000519
Lng                  0.384368  0.009766  0.001900
Population           0.615559 -0.015472 -0.001693
Children            -0.026406 -0.009178 -0.003833
Age                 -0.002319 -0.706684 -0.016469
Doc_visits          -0.056993 -0.005394 -0.000189
Full_meals_eaten     0.069754 -0.009569 -0.001491
VitD_supp           -0.021716 -0.005001 -0.001503
Initial_days        -0.000273 -0.005755  0.451212
```

E2. Number of PCs Retained

To determine the number of PCs to retain, I used the following code to calculate eigenvalues and generate a scree plot:

#Calculate eigenvalues

cov_matrix = np.dot(continuous_variables_normalized.T, continuous_variables_normalized) / continuous_variables.shape[0]

eigenvalues = [np.dot(eigenvector.T, np.dot(cov_matrix, eigenvector)) for eigenvector in pca.components_]

#Plot eigenvalues in a scree plot

plt.plot(eigenvalues)

plt.xlabel('Number of Components')

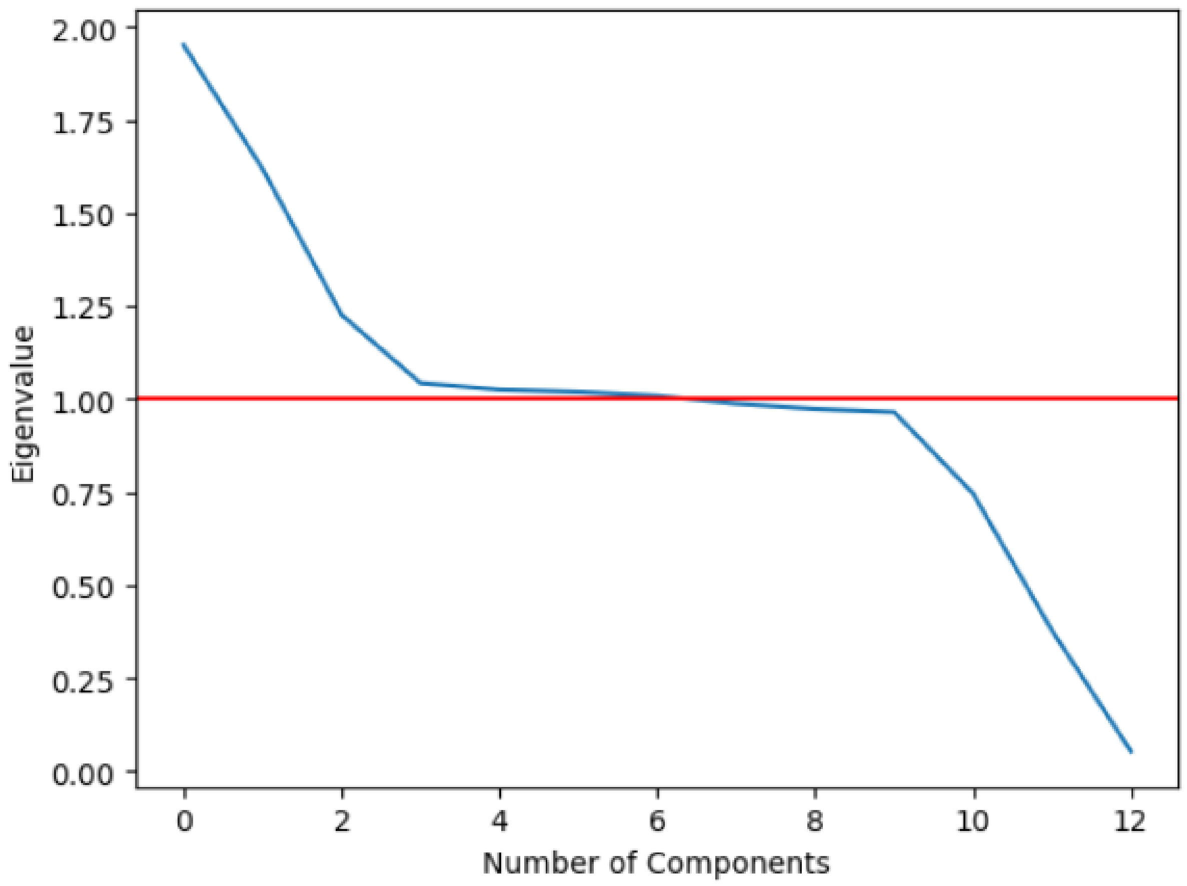
plt.ylabel('Eigenvalue')

plt.axhline(y=1, color='red')

plt.show()

```
The eigenvalues we[1.951554959538993, 1.6188633038310039, 1.226682173061143,
1.0432164857684547, 1.025229348027964, 1.0192269197641872,
1.0088630747223108, 0.9878536160126398, 0.9737904231091672,
0.9652487428459343, 0.7465038618818874, 0.37809819547966,
0.05356889595661104]
```
re:

The results of the scree plot were:



Principle Components:

PC1, PC2, PC3, PC4, PC5, PC6, PC7

E3. Benefits of PCA

The primary benefit of principle component analysis comes in the reduction of dimensionality. There are many benefits that result from conducting analysis on a reduced number of variables. While beyond the scope of this course, this reduction in variables is paramount in enabling efficient unsupervised analysis and training.

F. Panopto Video

See Video Link

G. Sources

Middleton, K. (n.d.). *Getting Started with D206 | Detecting and Treating Duplicates*. Panopto. https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=6eedfad4-240e-4c5c-8eab-b058003d3e6b

Middleton, K. (n.d.). *Getting Started with D206 | Detecting and Treating Missing Values*. Panopto. https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=767749d2-ba19-4f94-bec8-b058017b2f5e

Middleton, K. (n.d.). *Getting Started with D206 | Detecting and Treating Outliers*. Panopto. https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=19c24c56-0f37-408e-bb1f-b059002a77ac

Middleton, K. (n.d.). *Getting Started with Re-Expression of Categorical Variables*. Panopto. https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=bd7b8541-77ba-42e0-80a4-b059010bc790

Middleton, K. (n.d.). *Getting Started with D206 | Principal Component Analysis (PCA)*. Panopto. https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=3bcc452f-fa35-43be-b69f-b05901356f95