**PERFORMANCE ASSESSMENT:**

**D206 – DATA CLEANING**

Michael Lawson

Western Governors University

D206 – Data Cleaning

Keiona Middleton

10/21/21

Michael C. Lawson | Student ID: #001479557 | Program: MS Data Analytics (5/1/21) |

Program Mentor: Emil Stoica | 859-265-1888 (EST) | [mlaws77@wgu.edu](mailto:mlaws77@wgu.edu)

**Part I: Research Question**

1. Are factors present that predict the likelihood of patient readmission?

**B**. The data set represents 10,000 patient records for a popular medical hospital chain with patients in almost every state in the United States.

**Part II: Data-Cleaning Plan**

**C. Explanation of plan for cleaning the data.**

1. **Plan proposal with specific steps & techniques for detecting anomalies in the data.**

To locate anomalies that will answer the question in Part I(A), the data must be cleaned and prepped for analysis. The following steps will be implemented for finding anomalies:

* + - Examine the .csv file contents to gain a general idea of the data.
    - Import data to Jupyter Python 3 using pandas.read\_csv().
    - Examine the entire data set.
      * Use df.shape to see how many columns and rows are in the data.
      * Use df.info() to see the column names, where nulls are present, and the data type of each column.
    - Detect missing data.
      * Use df.isnull().sum() to see where and how many nulls are present.
    - Treat missing data.
      * Use df.hist() on each column containing nulls to decide how to treat them.
        + Infill qualitative/categorical data columns using mode.
        + Infill symmetric and uniform numerical data using mean.
        + Infill skewed or bi-modal data using median.
    - Detect Outliers.
      * Z-Scores will be used to detect outliers. Anything with a +3/-3 score will be examined more closely.
      * If z-scores don’t appear to accurately represent the data, then boxplots or histograms may be used.
    - Treat Outliers.
      * Decide whether to remove, retain, exclude or replace.
    - Export data set as .csv file.
    - Re-express categorical variables.
      * Examine whether categorical data is nominal or ordinal.
      * Replace ordinal values with numerical.
    - Perform Principal Component Analysis.
    - Export cleaned data set as .csv file.

(Middleton, 2021)

**C2. Justification of approach to assessing the quality of the data.**

* **Characteristics of the data being assessed:**

The Medical Data set includes 10,000 rows and 53 columns. The first columns are used to identify patients or separate visits or procedures. The next columns represent demographical data for patients. The ‘ReAdmis’ column will be important for finding the causation of readmissions. The next columns represent information about the actions taken by the hospital and medical conditions of the patient. The final 8 columns represent a questionnaire filled out by patients. The data set is too large to see where all the missing values are using the .csv alone, so Python will be used. There are no completely empty values. All nulls are filled with “NA”.

* + **The approach used to assess the quality:**

Using df.isnull().sum() quickly shows specific columns with missing values. Most columns appeared in the .csv file to be complete. Counting nulls in python this way is a fast way to display all the columns at once and show exactly where the null values are. Z-scores are a good starting place for detecting outliers. Z-scores are straightforward and easy to understand. Boxplots and histograms are also effective methods for detecting outliers, and if for any reason the Z-scores are not clear, boxplots or histograms will be used.

**C3. Justification of programming language and libraries or packages.**

Python is the programming language that will be used to clean the data. Python is a multi-purpose language, much like C++ and Java, with a readable syntax that’s easy to learn (IBM.com). For a beginner in programming languages, easier to learn is the safer path to take. Python is also the best choice for someone interested in machine learning and data analysis within web-based applications. Python supports a wider variety of data types, including JSON sourced from the web. Python uses the data analysis library Pandas to explore data, data modeling libraries like Numpy for numerical modeling analysis, SciPy for scientific computing and calculations, and scikit-learn for machine learning algorithms (IBM.com).

Pandas will be installed because it is a powerful software library for Python specifically written for manipulating numerical tables. I chose the NumPy library for its ability to perform mathematical functions on multi-dimensional arrays. SciPy is another library that performs powerful mathematical functions is was specifically imported on this project to find Z-scores when detecting outliers. Sklearn is a library directed toward machine learning and graphing data. Sklearn was installed on this project for the principal component analysis portion. Matplotlib is a library packed full of plots and graphs for visualizing. Missingno is another library of graphs specifically for visualizing the presence and distribution of missing data.

**4.  Code to identify the anomalies in the data:**

*#Install needed packages*

**!**pip install pandas

**!**pip install numpy

**!**pip install scipy

**!**pip install sklearn

**!**pip install matplotlib

**!**pip install missingno

*#import full packages or separate tools needed from packages*

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib

**import** matplotlib.pyplot **as** pp

**%matplotlib** inline

**import** missingno **as** msno

**from** sklearn.preprocessing **import** scale

**from** sklearn.decomposition **import** PCA

**import** scipy.stats **as** stats

*#Import data from hard drive*

med **=** pd**.**read\_csv(r"C:\Users\mlaws\OneDrive - Western Governors University\Documents\WGU\D206\medical\_raw\_data.csv")

*#Display the number of records and columns in data set*

print (med**.**shape)

*# View column names & data types*

print (med**.**info())

*#Examine total nulls in each column*

med**.**isnull()**.**sum()

*#Histograms will be generated for columns with nulls to assess how to treat.*

*#'Soft\_drink' column created an error because the values are not numerical.*

*#The .unique command shows variables in the column.*

med['Soft\_drink']**.**unique()

*#Replace yes/no variables with 1/0 similar to True/False boolean.*

med**.**Soft\_drink**.**replace(('Yes','No'),(1,0),inplace**=True**)

*#Check that values were changed*

med['Soft\_drink']**.**unique()

*#Due to the error with 'Soft\_drink', the analyst decided to re-express categorical values first.*

*#Convert the remaining yes/no columns to avoid similar issues for other columns.*

med**.**ReAdmis**.**replace(('Yes','No'),(1,0),inplace**=True**)

med**.**HighBlood**.**replace(('Yes','No'),(1,0),inplace**=True**)

med**.**Stroke**.**replace(('Yes','No'),(1,0),inplace**=True**)

med**.**Arthritis**.**replace(('Yes','No'),(1,0),inplace**=True**)

med**.**Diabetes**.**replace(('Yes','No'),(1,0),inplace**=True**)

med**.**Hyperlipidemia**.**replace(('Yes','No'),(1,0),inplace**=True**)

med**.**BackPain**.**replace(('Yes','No'),(1,0),inplace**=True**)

med**.**Allergic\_rhinitis**.**replace(('Yes','No'),(1,0),inplace**=True**)

med**.**Reflux\_esophagitis**.**replace(('Yes','No'),(1,0),inplace**=True**)

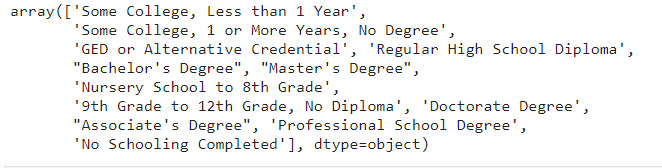
med**.**Asthma**.**replace(('Yes','No'),(1,0),inplace**=True**)

*#Rename survey answer columns to better represent data.*

med**.**rename(columns **=**{'Item1':'S1\_Timely\_admission', 'Item2':'S2\_Timely\_Treatment', 'Item3':'S3\_Timely\_Visits', 'Item4':'S4\_Reliability', 'Item5':'S5\_Options', 'Item6':'S6\_Hours\_of\_Treatment', 'Item7':'S7\_Courteous\_staff', 'Item8':'S8\_Dr\_Actively\_Listened'}, inplace**=True**)

*#Display each unique value in 'Education' column*

med**.**Education**.**unique()



*#Create a new education\_numeric column to convert 'Education' to ranked numeric.*

med['education\_numeric']**=** med['Education']

dict\_edu **=** {'education\_numeric':{'No Schooling Completed':0, 'Nursery School to 8th Grade':8, '9th Grade to 12th Grade, No Diploma':10, 'GED or Alternative Credential':12, 'Regular High School Diploma':12, 'Some College, Less than 1 Year':13, 'Some College, 1 or More Years, No Degree':13, "Associate's Degree" : 14, "Bachelor's Degree": 16, "Master's Degree": 20, 'Professional School Degree':20, 'Doctorate Degree':22, "unknown":np**.**NaN}}

med**.**replace(dict\_edu, inplace**=True**)

*#Verify that all values were converted to numbers.*

med**.**education\_numeric**.**unique()

*#Display each unique value in 'Complication\_risk' column.*

med**.**Complication\_risk**.**unique()

*#Replace ordinal values with numbers*

med**.**Complication\_risk**.**replace(('Low','Medium', 'High'),(0,1,2),inplace**=True**)

*#Verify that all values are now numeric.*

med**.**Complication\_risk**.**unique()

*#Display Unique Values of 'Area' column.*

med**.**Area**.**unique()

*#Rank 'Area' in respect to higher population.*

med**.**Area**.**replace(('Rural','Suburban', 'Urban'),(0,1,2),inplace**=True**)

*#Replace 'Timezone' values with numbers according to the relation to GMT.*

med['timezone\_numeric'] **=** med['Timezone']

dict\_time **=** {'timezone\_numeric':{'America/Chicago': **-**6, 'America/New\_York': **-**5, 'America/Los\_Angeles':**-**8, 'America/Indiana/Indianapolis': **-**5, ‘America/Detroit': **-**5, 'America/Denver': **-**7 , 'America/Nome': **-**9, 'America/Anchorage': **-**9, 'America/Phoenix': **-**7, 'America/Boise': **-**7, 'America/Puerto\_Rico': **-**4, 'America/Yakutat': **-**9, 'Pacific/Honolulu': **-**10, 'America/Menominee': **-**6, 'America/Kentucky/Louisville': **-**5, 'America/Indiana/Vincennes': **-**5, 'America/Toronto': **-**5, ‘America/Indiana/Marengo': **-**5, 'America/Indiana/Winamac': **-**5, 'America/Indiana/Tell\_City': **-**6, 'America/Sitka': **-**9, 'America/Indiana/Knox': **-**6, 'America/North\_Dakota/New\_Salem': **-**6, 'America/Indiana/Vevay': **-**5, 'America/Adak': **-**10, 'America/North\_Dakota/Beulah': **-**6}}

med**.**replace(dict\_time, inplace**=True**)

*#Verify 'Timezone' is numeric.*

med**.**timezone\_numeric**.**unique()

*#Create histograms for missing value fields.*

med**.**hist(['Age'])

med**.**hist(['Children'])

med**.**hist(['Income'])

med**.**hist(['Overweight'])

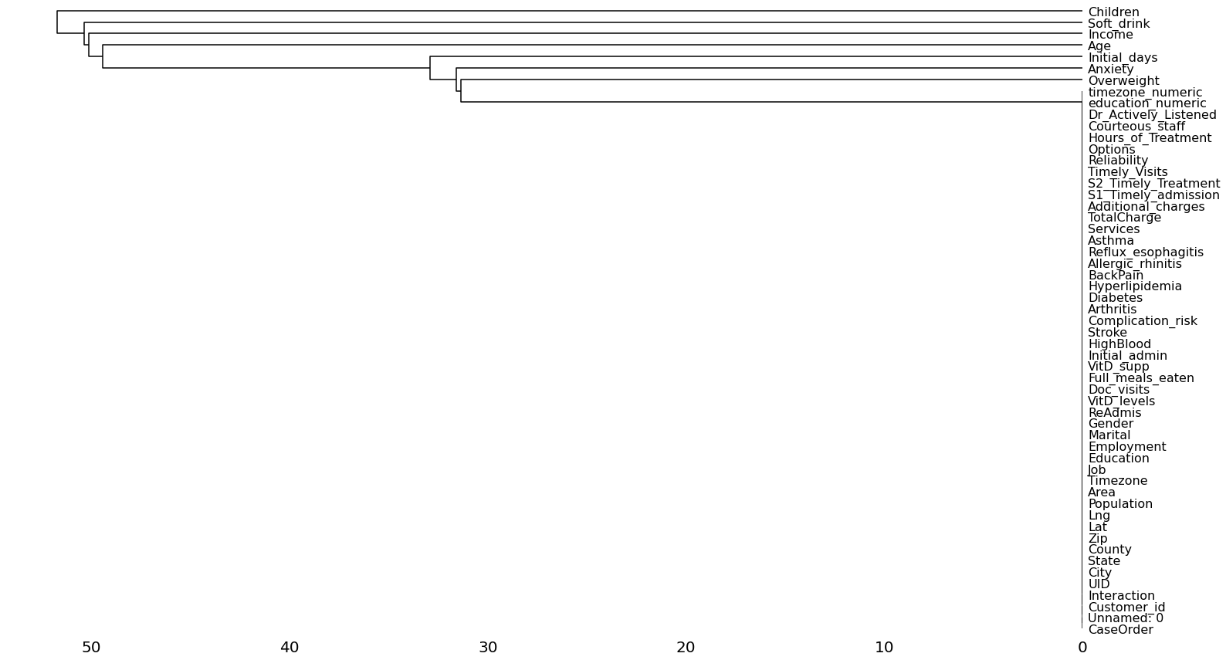
med**.**hist(['Anxiety'])

med**.**hist(['Initial\_days'])

med**.**hist(['Soft\_drink'])

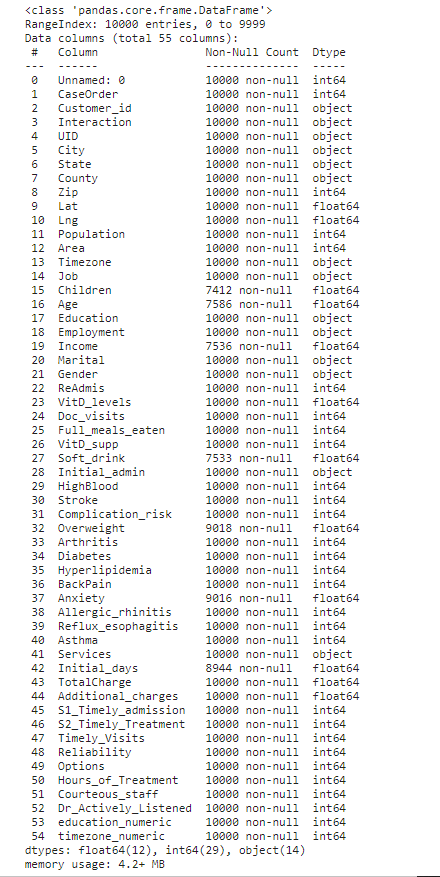
*#Create a dendrogram to see if there is any correlation in missing data that might change how missing values are treated.*

msno**.**dendrogram(med)

**

*#View what columns are still Dtype "object" to see if any ordinal value columns remain to be re-expressed.*

med**.**info()



*#Create Z-scores for numerical columns to detect outliers.*

med['children\_z'] **=** stats**.**zscore(med['Children'])

*#Separate Z-scores >3 & <-3 to separate outliers from the rest of the data set.*

children\_outliers **=** med**.**query('children\_z > 3 | children\_z < -3')

*#Display outliers in a histogram*

*#Out of 10,000 patients, it is possible that this number of patients had 8,9, or 10 children. Outliers will be retained.*

pp**.**hist(children\_outliers['Children'])

*#Zip Z-score*

med['zip\_z'] **=** stats**.**zscore(med['Zip'])

zip\_outliers **=** med**.**query('zip\_z > 3 | zip\_z < -3')

*#No outliers detected. No action taken.*

pp**.**hist(zip\_outliers['Zip'])

**

*#Population Z-score*

med['pop\_z'] **=** stats**.**zscore(med['Population'])

pop\_outliers **=** med**.**query('pop\_z > 3 | pop\_z < -3')

*#These values are realistic populations. These outliers will be retained and possibly adjusted later in the analysis cycle.*

pp**.**hist(pop\_outliers['Population'])

*#Age Z-scores*

med['age\_z'] **=** stats**.**zscore(med['Age'])

age\_outliers **=** med**.**query('age\_z > 3 | age\_z < -3')

*# No outliers detected*

pp**.**hist(age\_outliers['Age'])

**

*#Income Z-scores*

med['income\_z'] **=** stats**.**zscore(med['Income'])

income\_outliers **=** med**.**query('income\_z > 3 | income\_z < -3')

*#These incomes are possible for the size of the data set. These outliers will be retained.*

pp**.**hist(income\_outliers['Income'])

*#VitD\_levels Z-scores.*

med['VitD\_levels\_z'] **=** stats**.**zscore(med['VitD\_levels'])

vitd\_lev\_outliers **=** med**.**query('VitD\_levels\_z > 3 | VitD\_levels\_z < -3')

*# A quick Google search revealed vitamin D levels between 25-40 are normal.*

*#The highest recorded vitamin D level is over 200, so these levels and the number of values seem possible.*

*#Outliers will be retained.*

pp**.**hist(vitd\_lev\_outliers['VitD\_levels'])

*#Doc\_visits Z-scores.*

med['doc\_vis\_z'] **=** stats**.**zscore(med['Doc\_visits'])

doc\_vis\_outliers **=** med**.**query('doc\_vis\_z > 3 | doc\_vis\_z < -3')

*#6 patients had only one doctor visit and 2 had 9 visits. Out of 10,000 patients, this seems completely possible*

*#Outliers will be retained.*

pp**.**hist(doc\_vis\_outliers['Doc\_visits'])

*#Full\_meals\_eaten Z-scores.*

med['meals\_z'] **=** stats**.**zscore(med['Full\_meals\_eaten'])

meals\_outliers **=** med**.**query('meals\_z > 3 | meals\_z < -3')

*#The histogram is not clear on the values given.*

pp**.**hist(meals\_outliers['Full\_meals\_eaten'])

*#Unique values of Full\_meals\_eaten*

*#The column represents meals for the entire stay. the dictionary says patients are allowed to request more than 3 meals per day.*

*#Outliers will be retained.*

print(meals\_outliers['Full\_meals\_eaten']**.**unique())

[5 7 6]

*#VitD\_supp Z-scores*

med['vitd\_supp\_z'] **=** stats**.**zscore(med['VitD\_supp'])

vitd\_supp\_outliers **=** med**.**query('vitd\_supp\_z > 3 | vitd\_supp\_z < -3')

*#The dictionary states this represent total number of vitamin D supplements provided during stay.*

*#Values appear to be plausible. Outliers will be retained.*

pp**.**hist(vitd\_supp\_outliers['VitD\_supp'])

*#Initial\_days Z-scores.*

med['initial\_days\_z'] **=** stats**.**zscore(med['Initial\_days'])

initial\_days\_outliers **=** med**.**query('initial\_days\_z > 3 | initial\_days\_z < -3')

*#No outliers detected.*

pp**.**hist(initial\_days\_outliers['Initial\_days'])

**

*#TotalCharge Z-scores*

med['total\_charge\_z'] **=** stats**.**zscore(med['TotalCharge']

total\_charge\_outliers **=** med**.**query('total\_charge\_z > 3 | total\_charge\_z < -3')

*#There are enough values at these levels that they may be accurate.*

*#The values will be retained until further investigation in the data analysis cycle.*

pp**.**hist(total\_charge\_outliers['TotalCharge'])

*#Additional\_charges Z-scores.*

med['add\_charge\_z'] **=** stats**.**zscore(med['Additional\_charges'])

add\_charge\_outliers **=** med**.**query('add\_charge\_z > 3 | add\_charge\_z < -3')

*#No outliers detected.*

pp**.**hist(add\_charge\_outliers['Additional\_charges'])



**Part III: Data cleaning**

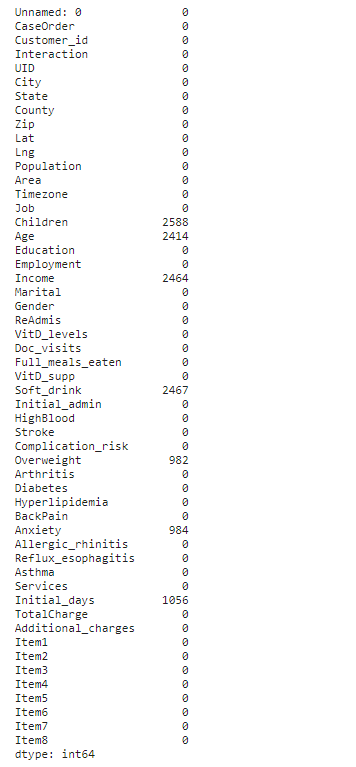
**D.**

**D1.**

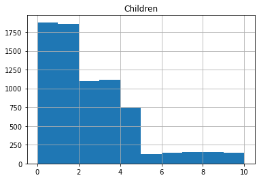
Cleaning the data with Python detected multiple columns with missing data and outliers.

* **Missing values were detected in seven columns.**

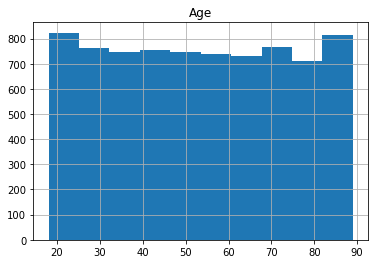
df.isnull().sum() revealed seven columns included the null value “NA”:



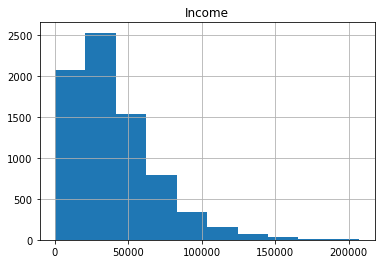
* The ‘Children’ column had 2588 null values.
* The data is skewed.

**

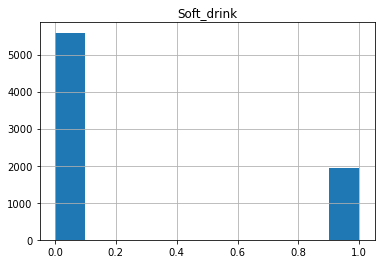
* The ‘Age’ column had 2414 null values.
* The data is uniform.

**

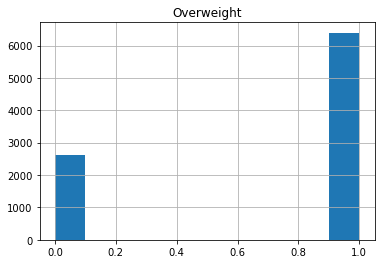
* The ‘Income’ column had 2464 null values.
* The data is skewed.

**

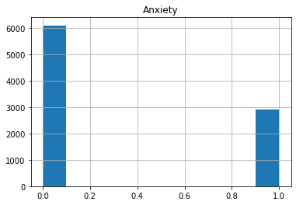
* The Soft\_drink column had 2467 null values.
* The data is skewed.

**

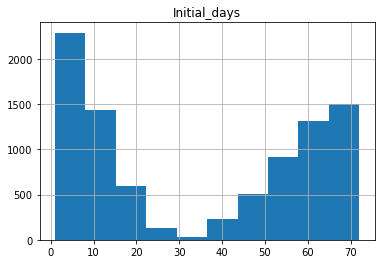
* The ‘Overweight’ column had 982 null values.
* The data is skewed.

**

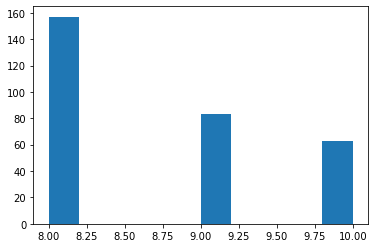
* The ‘Anxiety’ column had 984 null values.

**

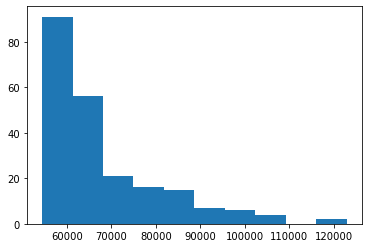
* The ‘Initial\_days column had1056 null values.

**

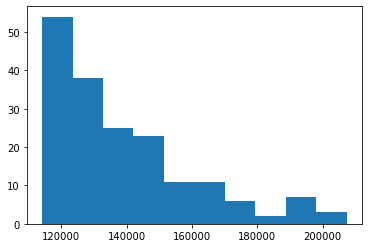
* **Outliers were detected in eight columns with Z-scores > 3 and < -3:**
  + The ‘Children’ column had outliers of 8, 9 & 10 children.

**

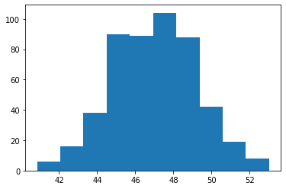
* + The ‘Population’ column had outliers with cities showing 60,000 to 120,000 people. The highest numbers were in the lower ranges, and only a few cities had over 100,000 people.

**

* + The ‘Income’ column had outliers ranging from 120,000 to 200,000. The highest density was in the lower side of this range, and only a few incomes were higher than 160,000.

**

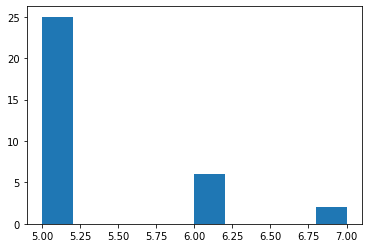
* + The VitD\_levels’ column showed a group of outliers with levels between 42-52. The outliers were symmetrically distributed with the highest density at 48. 25-40 is normal, and while 52 is high, it is common to see at a hospital. (Addy’s book, pg. 941)

**

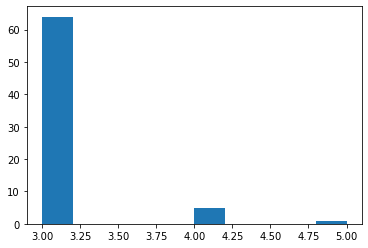
* + The ‘Doc\_visits’ column had six outlier values of 1, and 2 outlier values of 9.

**

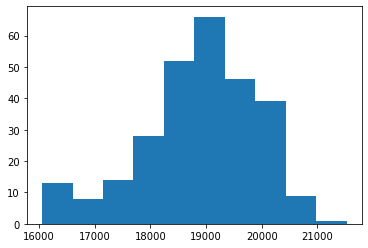
* + The ‘Full\_meals\_eaten’ column had twenty-five outlier values of 5 meals, six values of 6 meals, and two values of 7 meals.

**

* + The ‘VitD\_supp’ column had outlier Z-scores for over sixty values of 3, and less than ten values of 4 and 5.

**

* + The ‘TotalCharge’ column had outliers between 16,000 – 22,000. The outliers were symmetrically distributed with the highest density at 19,000.

**

**D2. Justification of methods of mitigating each type of anomaly.**

Histograms were generated for each of the columns containing null values. The distribution of the existing data helped determine the best way to treat the missing data. Only 4 columns out of 53 were missing close to 25% of the data for that column. The other 3 columns with missing data were only missing around 10% in each column. Deleting rows was not a viable option with so little data missing. Imputation was chosen to keep as much data as possible from the original data set.

Multivariate imputation was another method that was considered, but univariate using mean and median is a simpler method and equally accurate for filling missing values in the data set provided. Multivariate imputation would have hidden where the null values existed. In later steps of analysis, it could be helpful to know exactly where data was imputed into the data. Multivariate imputation using a method such as MICE would distribute the imputation across too many variables to track . Since deletion was not a viable option, univariate imputation was the best choice to both fill the missing values with something that represented the existing data set and keep those imputations in a trackable location.

Mean and median averages were imputed into the null data fields. The mean average was imputed into missing values in the ‘Age’ column because the histogram revealed a uniform distribution. Mean average is the most distributed type of average measurement, so it is the best choice for a uniform distribution of data. The ‘Children’, ‘Income’, Soft\_drink’, ‘Overweight’, ‘Anxiety’, and ‘Initial\_days’ columns are either skewed or bimodal, so the median average of existing data for each column was imputed into null values. The median average is the best choice for imputation in bi-modal and skewed data sets because it normally maintains the original shape of the data set (Algore, 2021).

All outliers were retained. Retention is a treatment of outliers for data that appears to be possible or legitimate. Values should not be changed without a solid justification. Deleting rows that contain outliers is a last resort method when the outliers are corrupting the presentation of data. None of the outliers detected appeared to be impossible or extremely different than any other values in the data set. No values were completely alone or extreme enough to justify excluding or changing them. They may be treated differently in the future if other steps of the data analysis life cycle reveal more information, but currently there is no justification for changing existing values that were detected as outliers using Z-scores. Z-scores were used for detecting outliers because of how simple they are to understand at a more precise level than boxplots, and how easily they are converted into histograms or other graphs to visualize the outliers. Methods other than histograms for visualizing outliers are possible with Z-scores, but with this data set histograms were substantial for deciding how to treat the outliers.

**D3. Summary of the outcome from the implementation of data-cleaning.**

All columns now show a “0” value for the total number of nulls in each column. Histograms were generated a second time for the seven columns that contained nulls.

The histograms have retained the same basic distribution curve that they originally had, so the imputation methods chosen were accepted as accurate representations of the original data. Retaining the outliers resulted in no changes at all to the data in relation to outliers.

Table

Description automatically generated with medium confidence

*Chart, histogram

Description automatically generatedChart, histogram

Description automatically generatedA picture containing text, shoji, clipart

Description automatically generatedA picture containing text, shoji, clipart

Description automatically generatedA picture containing text, shoji, public, clipart

Description automatically generatedChart

Description automatically generatedChart, histogram

Description automatically generated*

**D4. Code to treat anomalies:**

*#Fill null values of balances data with mean average*

med['Age']**.**fillna(med['Age']**.**median(), inplace**=True**)

*#Fill null values of skewed or bi-modal data with median.*

med['Children']**.**fillna(med['Children']**.**median(), inplace**=True**)

med['Income']**.**fillna(med['Income']**.**median(), inplace**=True**)

med['Overweight']**.**fillna(med['Overweight']**.**median(), inplace**=True**)

med['Anxiety']**.**fillna(med['Anxiety']**.**median(), inplace**=True**)

med['Initial\_days']**.**fillna(med['Initial\_days']**.**median(), inplace**=True**)

med['Soft\_drink']**.**fillna(med['Soft\_drink']**.**median(), inplace**=True**)

*#Check that null values are full.*

med**.**isnull()**.**sum()

*#Histograms of columns that have had missing values imputed.*

*#This will display if the distribution of data has greatly varied from the original data set.*

med**.**hist(['Age'])

med**.**hist(['Children'])

med**.**hist(['Income'])

med**.**hist(['Overweight'])

med**.**hist(['Anxiety'])med**.**hist(['Initial\_days'])

med**.**hist(['Soft\_drink'])

**D5.**

#Export clean data set.

med**.**to\_csv(r"C:\Users\mlaws\OneDrive - Western Governors University\Documents\WGU\D206\medical\_clean\_data.csv")

**D6. Summary of the limitations of the data cleaning process.**

Imputation was the method chosen for filling null values.

* Imputation is a guess.
* It is possible the imputed data doesn’t match the real data at all.
* Imputation can change the shape of the data.

Retention was the chosen method for all outliers.

* Outliers, even if accurate data, can skew the overall picture of the data.
* The possibility remains that the outliers are indeed inaccurate and not properly representing the data.

**D7. Discuss the limitations in D6.**

Univariate imputation was the chosen method for filling the missing data. Imputation is a guess. The goal of using the mean and median average is to impute the best guess that represents the existing original data. Even the most accurate guess is not the same as having real data collected with the original data set. The goal with imputation was to keep the data shape as close to the original shape as possible. The ‘Age’ column revealed a limitation of univariate imputation. Nearly 25% of the data was missing, so no matter what single number was entered, the uniform shape of the histogram has one column that spikes up above all the other values. MICE was considered as an option to spread this spike out using multivariate imputation, but as discussed earlier, having multiple variables imputed would have made tracking the data that wasn’t real or original much harder. The averages used still represented the original data.

Retention was the treatment used for outliers in the data set. Retaining outliers could possibly skew the data. Retention could possibly change how the data looks in later steps. If outliers are not accurate information meant to be in the data, they could change the predictions and answers to questions about the data. Without more information, the outliers appeared to be accurate and possible data. They may be dealt with differently in future steps of the analysis process. The data was retained because it appeared to represent true values of the data, but the existence of any values far separated from the rest of the data could change how the data looks in future steps.

**E. Principal Component Analysis (PCA)**

**E1. What are the principal components?**

* + Survey Questions
    1. S1\_Timely\_admission, S2\_Timely\_Treatment, S3\_Timely\_Visits, S6\_Hours\_of\_Treatment, S7\_Courteous\_Staff, S8\_Dr\_Actively\_Listened
  + Location
    1. Lng, Timezone

Python found 39 principal components. 8 of those were clearly above 1 eigenvalue on a scree plot, and 7 more were level with or slightly above 1 eigenvalue. A clear definition of every principal component was not as easy as the first two, listed above.

The following variables were loaded into the data frame created to perform data analysis because they are completely filled with numerical values:

'Zip', 'Lat', 'Lng', 'Population', 'Area', 'Children', 'Age', 'education\_numeric', 'Income', 'ReAdmis', 'VitD\_levels', 'Doc\_visits', 'Full\_meals\_eaten','VitD\_supp', 'Soft\_drink','HighBlood','Stroke','Complication\_risk', 'Overweight','Arthritis', 'Diabetes', 'Hyperlipidemia','BackPain', 'Anxiety'

**E2. How PCA was identified.**

Principle components identified using eigenvalues that loaded in Python (Figure 2a) as shown below. These values were then loaded into a scree plot (Figure 2b) to have a visualization of how many principal components landed above an eigenvalue of 1.

Figure 2a:

Table

Description automatically generated

Figure 2b:

Chart, line chart

Description automatically generated

The scree plot clearly shows the first 8 principal components above 1 eigenvalue. The next 7 values are very close to 1 but are in the “rubble” of the scree plot. The variables in the third principal component were :ReAdmis, VitD\_levels, Initial\_days, TotalCharge. A name for this group was not created because it was not clear yet how these columns relate. This may change further in the analysis life cycle.

The code used for PCA is as follows:

med\_pca = med[['Zip', 'Lat', 'Lng', 'Population', 'Area', 'Children', 'Age', 'education\_numeric', 'Income', 'ReAdmis', 'VitD\_levels', 'Doc\_visits', 'Full\_meals\_eaten','VitD\_supp', 'Soft\_drink','HighBlood','Stroke','Complication\_risk', 'Overweight','Arthritis', 'Diabetes', 'Hyperlipidemia','BackPain', 'Anxiety'

med\_normalized = (med\_pca - med\_pca.mean())/med\_pca.std()

pca = PCA(n\_components = med\_pca.shape[1])

pca.fit(med\_normalized)



loadings = pd.DataFrame(pca.components\_.T,

columns = ['PCA1', 'PCA2', 'PCA3', 'PCA4','PCA5','PCA6', 'PCA7', 'PCA8','PCA9','PCA10','PCA11','PCA12','PCA13','PCA14','PCA15','PCA16','PCA17','PCA18','PCA19','PCA20','PCA21','PCA22','PCA23','PCA24','PCA25','PCA26','PCA27','PCA28','PCA29','PCA30','PCA31','PCA32','PCA33','PCA34','PCA35','PCA36','PCA37','PCA38','PCA39'],

index=med\_normalized.columns)

loadings

cov\_matrix = np.dot(med\_normalized.T, med\_normalized)/med\_pca.shape[0]

eigenvalues = [np.dot(eigenvector.T, np.dot(cov\_matrix, eigenvector)) for eigenvector in pca.components\_]

pp.plot(eigenvalues, 'ro-')

pp.xlabel('Number of Components')

pp.ylabel('Eigenvalues')

pp.show()

**E3. Describe how the organization can benefit from PCA.**

The organization can benefit from PCA because the data can be grouped into fewer dimensions that are easier to analyze. Some variables are so closely related, the data will look the same whether the variables are grouped together or not. The big picture isn’t changed, and the number of variables is reduced which helps with ease and speed of manipulating the data, presenting it, and in the end answering the question about the data. PCA helped emphasize attention should be given to the columns of the data that represent the survey.

**Part IV. Supporting Documents.**

**F. Panapto Recording:**

[Friday, October 22, 2021 at 16:02:00 (panopto.com)](https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=7d0cfe15-86a7-4ffd-a798-adc9014a463c)

**G. Web Sources**

IBM Cloud Team, IBM Cloud (2021, March 23) *Python vs. R: What’s the Difference?* <https://www.ibm.com/cloud/blog/python-vs-r>

Middleton, Keiona (2021, October) *Getting Started with “Re-Expression of Categorical Variables” Data cleaning D206* [Cisco Webex Meetings - Replay Recorded Meeting](https://wgu.webex.com/recordingservice/sites/wgu/recording/184840d81356103a96590050568f9b64/playback)

Middleton, Keiona (2021, August) *Understanding PCA; Data cleaning D206.* [Cisco Webex Meetings - Replay Recorded Meeting](https://wgu.webex.com/recordingservice/sites/wgu/recording/af6491cce754103997fd00505681be2e/playback)

**H. Sources**

Algore, Matt (2021, January 6*) Python For Data Analysis: The Ultimate and Definitive Manual to Learn Data Science and Coding With Python. Master The basics of Machine Learning, to Clean Code and Improve Artificial Intelligence.* (Independently published)

Larose, C.D. & Larose, D.T (2019. Data Science: Using Python and R. (John Wiley & Sons, Inc.)

***Raw code used for performance assessment:***

*#Install needed packages*

**!**pip install pandas

**!**pip install numpy

**!**pip install scipy

**!**pip install sklearn

**!**pip install matplotlib

**!**pip install missingno

*#import full packages or separate tools needed from packages*

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib

**import** matplotlib.pyplot **as** pp

**%matplotlib** inline

**import** missingno **as** msno

**from** sklearn.preprocessing **import** scale

**from** sklearn.decomposition **import** PCA

**import** scipy.stats **as** stats

*#Import data from hard drive*

med **=** pd**.**read\_csv(r"C:\Users\mlaws\OneDrive - Western Governors University\Documents\WGU\D206\medical\_raw\_data.csv")

*#Display the number of records and columns in data set*

print (med**.**shape)

*# View column names & data types*

print (med**.**info())

*#histograms will be generated for columns with nulls to assess how to treat.*

*#'Soft\_drink' column created an error because the values are not numerical.*

*#The .unique command shows variables in the column.*

med['Soft\_drink']**.**unique()

*#Replace yes/no variables with 1/0 similar to True/False boolean.*

med**.**Soft\_drink**.**replace(('Yes','No'),(1,0),inplace**=True**)

*#Check that values were changed*

med['Soft\_drink']**.**unique()

*#Due to the error with 'Soft\_drink', the analyst decided to re-express categorical values first.*

*#Convert the remaining yes/no columns to avoid similar issues for other columns.*

med**.**ReAdmis**.**replace(('Yes','No'),(1,0),inplace**=True**)

med**.**HighBlood**.**replace(('Yes','No'),(1,0),inplace**=True**)

med**.**Stroke**.**replace(('Yes','No'),(1,0),inplace**=True**)

med**.**Arthritis**.**replace(('Yes','No'),(1,0),inplace**=True**)

med**.**Diabetes**.**replace(('Yes','No'),(1,0),inplace**=True**)

med**.**Hyperlipidemia**.**replace(('Yes','No'),(1,0),inplace**=True**)

med**.**BackPain**.**replace(('Yes','No'),(1,0),inplace**=True**)

med**.**Allergic\_rhinitis**.**replace(('Yes','No'),(1,0),inplace**=True**)

med**.**Reflux\_esophagitis**.**replace(('Yes','No'),(1,0),inplace**=True**)

med**.**Asthma**.**replace(('Yes','No'),(1,0),inplace**=True**)

*#Rename survey answer columns to better represent data.*

med**.**rename(columns **=**{'Item1':'S1\_Timely\_admission', 'Item2':'S2\_Timely\_Treatment', 'Item3':'S3\_Timely\_Visits', 'Item4':'S4\_Reliability', 'Item5':'S5\_Options', 'Item6':'S6\_Hours\_of\_Treatment', 'Item7':'S7\_Courteous\_staff', 'Item8':'S8\_Dr\_Actively\_Listened'}, inplace**=True**)

*#Display each unique value in 'Education' column*

med**.**Education**.**unique()

array(['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'], dtype=object)

*#Create a new education\_numeric column to convert 'Education' to ranked numeric.*

med['education\_numeric']**=** med['Education']

dict\_edu **=** {'education\_numeric':{'No Schooling Completed':0, 'Nursery School to 8th Grade':8, '9th Grade to 12th Grade, No Diploma':10, 'GED or Alternative Credential':12, 'Regular High School Diploma':12, 'Some College, Less than 1 Year':13, 'Some College, 1 or More Years, No Degree':13, "Associate's Degree" : 14, "Bachelor's Degree": 16, "Master's Degree": 20, 'Professional School Degree':20, 'Doctorate Degree':22, "unknown":np**.**NaN}}

med**.**replace(dict\_edu, inplace**=True**)

*#Verify that all values were converted to numbers.*

med**.**education\_numeric**.**unique()

*#Display each unique value in 'Complication\_risk' column.*

med**.**Complication\_risk**.**unique()

array(['Medium', 'High', 'Low'], dtype=object)

*#Replace ordinal values with numbers*

med**.**Complication\_risk**.**replace(('Low','Medium', 'High'),(0,1,2),inplace**=True**)

*#Verify that all values are now numeric.*

med**.**Complication\_risk**.**unique()

array([1, 2, 0], dtype=int64)

*#Display Unique Values of 'Area' column.*

med**.**Area**.**unique()

array(['Suburban', 'Urban', 'Rural'], dtype=object)

*#Rank 'Area' in respect to higher population.*

med**.**Area**.**replace(('Rural','Suburban', 'Urban'),(0,1,2),inplace**=True**)

*#Replace 'Timezone' values with numbers according to the relation to GMT.*

med['timezone\_numeric'] **=** med['Timezone']

dict\_time **=** {'timezone\_numeric':{'America/Chicago': **-**6, 'America/New\_York': **-**5, 'America/Los\_Angeles':**-**8,

'America/Indiana/Indianapolis': **-**5, 'America/Detroit': **-**5,

'America/Denver': **-**7 , 'America/Nome': **-**9, 'America/Anchorage': **-**9,

'America/Phoenix': **-**7, 'America/Boise': **-**7, 'America/Puerto\_Rico': **-**4,

'America/Yakutat': **-**9, 'Pacific/Honolulu': **-**10, 'America/Menominee': **-**6,

'America/Kentucky/Louisville': **-**5, 'America/Indiana/Vincennes': **-**5,

'America/Toronto': **-**5, 'America/Indiana/Marengo': **-**5,

'America/Indiana/Winamac': **-**5, 'America/Indiana/Tell\_City': **-**6,

'America/Sitka': **-**9, 'America/Indiana/Knox': **-**6,

'America/North\_Dakota/New\_Salem': **-**6, 'America/Indiana/Vevay': **-**5,

'America/Adak': **-**10, 'America/North\_Dakota/Beulah': **-**6}}

med**.**replace(dict\_time, inplace**=True**)

*#Verify 'Timezone' is numeric.*

med**.**timezone\_numeric**.**unique()

*#Create histograms for missing value fields.*

med**.**hist(['Age'])

med**.**hist(['Children'])

med**.**hist(['Income'])

med**.**hist(['Overweight'])

med**.**hist(['Anxiety'])

med**.**hist(['Initial\_days'])

med**.**hist(['Soft\_drink'])

*#Create a dendrogram to see if there is any correlation in missing data that might change how missing values are treated.*

msno**.**dendrogram(med)

*#View what columns are still Dtype "object" to see if any ordinal value columns remain to be re-expressed.*

med**.**info()

*#Fill null values of balances data with mean average*

med['Age']**.**fillna(med['Age']**.**median(), inplace**=True**)

*#Fill null values of skewed or bi-modal data with median.*

med['Children']**.**fillna(med['Children']**.**median(), inplace**=True**)

med['Income']**.**fillna(med['Income']**.**median(), inplace**=True**)

med['Overweight']**.**fillna(med['Overweight']**.**median(), inplace**=True**)

med['Anxiety']**.**fillna(med['Anxiety']**.**median(), inplace**=True**)

med['Initial\_days']**.**fillna(med['Initial\_days']**.**median(), inplace**=True**)

med['Soft\_drink']**.**fillna(med['Soft\_drink']**.**median(), inplace**=True**)

*#Check that null values are full*

med**.**isnull()**.**sum()

*#Histograms of columns that have had missing values imputed.*

*#This will display if the distribution of data has greatly varied from the original data set.*

med**.**hist(['Age'])

med**.**hist(['Children'])

med**.**hist(['Income'])

med**.**hist(['Overweight'])

med**.**hist(['Anxiety'])

med**.**hist(['Initial\_days'])

med**.**hist(['Soft\_drink'])

*#Create Z-scores for numerical columns to detect outliers.*

med['children\_z'] **=** stats**.**zscore(med['Children'])

*#Separate Z-scores >3 & <-3 to separate outliers from the rest of the data set.*

children\_outliers **=** med**.**query('children\_z > 3 | children\_z < -3')

*#Display outliers in a histogram*

*#Out of 10,000 patients, it is possible that this number of patients had 8,9, or 10 children. Outliers will be retained.*

pp**.**hist(children\_outliers['Children'])

*#Zip Z-score*

med['zip\_z'] **=** stats**.**zscore(med['Zip'])

zip\_outliers **=** med**.**query('zip\_z > 3 | zip\_z < -3')

*#No outliers detected. No action taken.*

pp**.**hist(zip\_outliers['Zip'])

*#Population Z-score*

med['pop\_z'] **=** stats**.**zscore(med['Population'])

pop\_outliers **=** med**.**query('pop\_z > 3 | pop\_z < -3')

*#These values are realistic populations. These outliers will be retained and adjusted if needed later on in the analysis cycle.*

pp**.**hist(pop\_outliers['Population'])

*#Age Z-scores*

med['age\_z'] **=** stats**.**zscore(med['Age'])

In [605]:

age\_outliers **=** med**.**query('age\_z > 3 | age\_z < -3')

*#Income Z-scores*

med['income\_z'] **=** stats**.**zscore(med['Income'])

income\_outliers **=** med**.**query('income\_z > 3 | income\_z < -3')

*#These incomes are possible for the size of the data set. These outliers will be retained.*

pp**.**hist(income\_outliers['Income'])

*#VitD\_levels Z-scores.*

med['VitD\_levels\_z'] **=** stats**.**zscore(med['VitD\_levels'])

vitd\_lev\_outliers **=** med**.**query('VitD\_levels\_z > 3 | VitD\_levels\_z < -3')

*# A quick Google search revealed vitamin D levels between 25-40 are normal.*

*#The highest recorded vitamin D level is over 200, so these levels and the number of values seem possible.*

*#Outliers will be retained.*

pp**.**hist(vitd\_lev\_outliers['VitD\_levels'])

*#Doc\_visits Z-scores.*

med['doc\_vis\_z'] **=** stats**.**zscore(med['Doc\_visits'])

doc\_vis\_outliers **=** med**.**query('doc\_vis\_z > 3 | doc\_vis\_z < -3')

*#6 patients had only one doctor visit and 2 had 9 visits. Out of 10,000 patients, this seems completely possible*

*#Outliers will be retained.*

pp**.**hist(doc\_vis\_outliers['Doc\_visits'])

*#Full\_meals\_eaten Z-scores.*

med['meals\_z'] **=** stats**.**zscore(med['Full\_meals\_eaten'])

meals\_outliers **=** med**.**query('meals\_z > 3 | meals\_z < -3')

*#The histogram is not clear on the values given.*

pp**.**hist(meals\_outliers['Full\_meals\_eaten'])

*#Unique values of Full\_meals\_eaten*

*#The column represents meals for the entire stay. the dictionary says patients are allowed to request more than 3 meals per day.*

*#Outliers will be retained.*

print(meals\_outliers['Full\_meals\_eaten']**.**unique())

*#VitD\_supp Z-scores*

med['vitd\_supp\_z'] **=** stats**.**zscore(med['VitD\_supp'])

vitd\_supp\_outliers **=** med**.**query('vitd\_supp\_z > 3 | vitd\_supp\_z < -3')

*#The disctionary states this represent total number of vitamin D supplements provided during stay.*

*#Values appear to be plausible. Outliers will be retained.*

pp**.**hist(vitd\_supp\_outliers['VitD\_supp'])

*#Initial\_days Z-scores.*

med['initial\_days\_z'] **=** stats**.**zscore(med['Initial\_days'])

initial\_days\_outliers **=** med**.**query('initial\_days\_z > 3 | initial\_days\_z < -3')

*#No outliers detected.*

pp**.**hist(initial\_days\_outliers['Initial\_days'])

*#TotalCharge Z-scores*

med['total\_charge\_z'] **=** stats**.**zscore(med['TotalCharge'])

total\_charge\_outliers **=** med**.**query('total\_charge\_z > 3 | total\_charge\_z < -3')

In [628]:

*#There are enough values at these levels that they may be accurate for multiple day stays.*

*#They will be retained until further investigation in the data analysis cycle.*

pp**.**hist(total\_charge\_outliers['TotalCharge'])

*#Additional\_charges Z-scores.*

med['add\_charge\_z'] **=** stats**.**zscore(med['Additional\_charges'])

add\_charge\_outliers **=** med**.**query('add\_charge\_z > 3 | add\_charge\_z < -3')

*#No outliers detected.*

pp**.**hist(add\_charge\_outliers['Additional\_charges'])

med**.**to\_csv(r"C:\Users\mlaws\OneDrive - Western Governors University\Documents\WGU\D206\medical\_clean\_data.csv")

med**.**info()

med\_pca **=** med[['Zip', 'Lat', 'Lng', 'Population', 'Area', 'Children', 'Age', 'education\_numeric', 'Income', 'ReAdmis', 'VitD\_levels', 'Doc\_visits', 'Full\_meals\_eaten','VitD\_supp', 'Soft\_drink','HighBlood','Stroke','Complication\_risk', 'Overweight','Arthritis', 'Diabetes', 'Hyperlipidemia','BackPain', 'Anxiety', 'Allergic\_rhinitis', 'Reflux\_esophagitis', 'Asthma', 'Initial\_days','TotalCharge','Additional\_charges','S1\_Timely\_admission','S2\_Timely\_Treatment','S3\_Timely\_Visits','S4\_Reliability', 'S5\_Options','S6\_Hours\_of\_Treatment','S7\_Courteous\_staff','S8\_Dr\_Actively\_Listened', 'timezone\_numeric']]

med\_normalized **=** (med\_pca **-** med\_pca**.**mean())**/**med\_pca**.**std()

pca **=** PCA(n\_components **=** med\_pca**.**shape[1])

pca**.**fit(med\_normalized)

PCA(n\_components=39)

loadings **=** pd**.**DataFrame(pca**.**components\_**.**T,

columns **=** ['PCA1', 'PCA2', 'PCA3', 'PCA4','PCA5','PCA6', 'PCA7', 'PCA8','PCA9','PCA10','PCA11','PCA12','PCA13','PCA14','PCA15','PCA16','PCA17','PCA18','PCA19','PCA20','PCA21','PCA22','PCA23','PCA24','PCA25','PCA26','PCA27','PCA28','PCA29','PCA30','PCA31','PCA32','PCA33','PCA34','PCA35','PCA36','PCA37','PCA38','PCA39'],

index**=**med\_normalized**.**columns)

loadings

cov\_matrix **=** np**.**dot(med\_normalized**.**T, med\_normalized)**/**med\_pca**.**shape[0]

eigenvalues **=** [np**.**dot(eigenvector**.**T, np**.**dot(cov\_matrix, eigenvector)) **for** eigenvector **in** pca**.**components\_]

pp**.**plot(eigenvalues, 'ro-')

pp**.**xlabel('Number of Components')

pp**.**ylabel('Eigenvalues')

pp**.**show()

*#Final Export*

med**.**to\_csv(r"C:\Users\mlaws\OneDrive - Western Governors University\Documents\WGU\D206\medical\_clean\_data\_withpca.csv")