D206 Data Cleaning

Jeff Enumerables

Western Governors University

M.S. Data Analytics

**Part I: Research Question**

**A: Question or Decision**

What are the key factors from the patient's medical information in predicting higher rates of readmission? Reducing readmission rates for future analysis will allow hospitals to develop strategies to reduce readmission and penalties.

**B: Required Variables**

The data set contains 10,000 records and 50 variables containing various patient information, demographics, medical conditions, readmission, and survey responses. Examples are below:

| **Variable Name** | **Data Type** | **Description** | **Example** |
| --- | --- | --- | --- |
| Unnamed: 0 | Quantitative – Numerical | Unnamed variable | 1 |
| CaseOrder | Quantitative – Numerical | A placeholder of the original raw data file | 1 |
| Customer\_id | Quantitative – Numerical | Unique patient ID number | C412403 |
| Interaction | Quantitative – Numerical | Patient transactions, procedures, and admissions | 8cd49b13-f45a-4b47-a2bd-173ffa932c2f |
| UID | Quantitative – Numerical | Unique ID related to patients interaction | 3a83ddb66e2ae73798bdf1d705dc0932 |
| City | Qualitative – Categorical | The city of the patient resides | Eva |
| State | Qualitative – Categorical | The state of the patient resides | AL |
| County | Qualitative – Categorical | The county the patient resides. | Morgan |
| Zip | Quantitative – Numerical | The patient's zip code | 35621 |
| Lat | Quantitative – Numerical | The GPS latitude of the patient's residence | 34.3496 |
| Lng | Quantitative – Numerical | The GPS longitude of the patient's residence | -86.72508 |
| Population | Quantitative – Numerical | Population within a mile radius | 2951 |
| Area | Qualitative – Categorical | The type of area | Suburban |
| Timezone | Qualitative – Categorical | The patient time zone place of residence | America/Chicago |
| Job | Qualitative – Categorical | The patient job | Psychologist, sport and exercise |
| Children | Quantitative – Numerical | The number of children in the patient household | 1.0 |
| Age | Quantitative – Numerical | The age of the patient | 53.0 |
| Education | Qualitative – Categorical | Highest earned degree of the patient | Some College, Less than 1 Year |
| Employment | Qualitative – Categorical | Patient employment status | Full Time |
| Income | Quantitative – Numerical | Annual income of the patient | 86575.93 |
| Marital | Qualitative – Categorical | Patient martial status | Divorced |
| Gender | Qualitative – Categorical | Patient self-identification | Male |
| ReAdmis | Qualitative – Categorical | Indicates whether patient is readmitted within a month | No |
| VitD\_levels | Quantitative – Numerical | The patient's vitamin D levels measured in ng/ML | 17.802330 |
| Doc\_visits | Quantitative – Numerical | Number of times the primary physician visited the patient during initial hospitalization | 6 |
| Full\_meals\_eaten | Quantitative – Numerical | Meals eaten during hospital stay. Partial meals count as 0. Some patients had more than 3 meals a day if requested. | 0 |
| VitD\_supp | Quantitative – Numerical | The number of times vitamin D administered to the patient | 0 |
| Soft\_drink | Qualitative – Categorical | If the patient drinks three or more sodas a day | No |
| Initial\_admin | Qualitative – Categorical | Initial  hospital admission reason | Emergency Admission |
| HighBlood | Qualitative – Categorical | Indicates patient has high blood pressure | Yes |
| Stroke | Qualitative – Categorical | Indicates patient has had a stroke | No |
| Complication\_risk | Qualitative – Categorical | Level of complication risk assessed by the primary patient assessment | Medium |
| Overweight | Qualitative – Categorical | Considered overweight based on age, gender, and height | 0.0 |
| Arthritis | Qualitative – Categorical | Indicates patient has arthritis | Yes |
| Diabetes | Qualitative – Categorical | Indicates patient has diabetes | No |
| Hyperlipidemia | Qualitative – Categorical | Indicates patient has hyperlipidemia | No |
| BackPain | Qualitative – Categorical | Indicates patient has back pain | Yes |
| Anxiety | Qualitative – Categorical | Indicates patient has anxiety | 1.0 |
| Allergic\_rhinitis | Qualitative – Categorical | Indicates patient has allergic rhinitis | Yes |
| Reflux\_esophagitis | Qualitative – Categorical | Indicates patient has reflux esophagitis | No |
| Asthma | Qualitative – Categorical | Indicates patient has asthma | Yes |
| Services | Qualitative – Categorical | Primary service the patient received during hospitalization | Blood Work |
| Initial\_days | Quantitative – Numerical | The number of days the patient stayed in the hospital during initial admission | 10.585770 |
| TotalCharge | Quantitative – Numerical | Daily amount charged to the patient. The charge is an averaged per patient based on the total charge divided by number of days in the hospital. Represents typical charges billed, not including specialized treatments. | 3191.048774 |
| Additional\_charges | Quantitative – Numerical | Average amount charged to the patient for miscellaneous procedures, treatments, medicines, etc. | 17939.403420 |
| Item 1 | Qualitative – Categorical | Timely admission response to survey question asking patient to rate the importance of factors on a scale of 1 to 8. 1 being most important and 8 being the least important | 3 |
| Item 2 | Qualitative – Categorical | Timely treatment response to survey question asking patient to rate the importance of factors on a scale of 1 to 8. 1 being most important and 8 being the least important | 3 |
| Item 3 | Qualitative – Categorical | Timely visit's response to a survey question asking patient to rate the importance of factors on a scale of 1 to 8. 1 being most important and 8 being the least important | 2 |
| Item 4 | Qualitative – Categorical | Reliability response to survey question asking patient to rate the importance of factors on a scale of 1 to 8. 1 being most important and 8 being the least important | 2 |
| Item 5 | Qualitative – Categorical | Options response to survey question asking patient to rate the importance of factors on a scale of 1 to 8. 1 being most important and 8 being the least important | 4 |
| Item 6 | Qualitative – Categorical | Hours of treatment response to survey question asking patient to rate the importance of factors on a scale of 1 to 8. 1 being most important and 8 being the least important | 3 |
| Item 6 | Qualitative – Categorical | Courteous staff response to survey question asking patient to rate the importance of factors on a scale of 1 to 8. 1 being most important and 8 being the least important | 3 |
| Item 8 | Qualitative – Categorical | Evidence of active listening from doctor response to survey question asking patient to rate the importance of factors on a scale of 1 to 8. 1 being most important and 8 being the least important | 4 |

**Part II: Data-Cleaning Plan**

**C1: Plan to Find Anomalies**

My plan to for cleaning the data will follow the steps below:

1. The dataset is imported from the raw CSV file into a python pandas data frame.
2. Review detailed information about the variables, inspect the data frame structure, and ensure they align with the data dictionary.
3. Inspect data set field values, number of columns and rows, and data types.
4. Add an index field to the dataset if one does not already exist.
5. Drop redundant columns or columns not useful for data analysis.
6. Check for duplicate data entries in the data frame.
7. Examine categorical variables for misspellings, review unique characteristics, and treat misleading field values in the data set.
8. Reduce decimal precision of values and convert floating point numbers to integer values where needed
9. Detect missing values and examine the distribution of missing variables.
10. Perform imputation of missing values by filling null values with univariate imputation based on the distribution of values.
11. Identify outliers that exist using Z-score calculations, box plots, and histograms
12. Re-express categorical variables that will be used for principal component analysis.
13. Perform Principal Component Analysis
14. Export the cleaned data to a CSV file.

**C2: Justification of Approach**

I import the CSV file into pandas data frame to be read, which allows me to use advanced cleaning techniques rather than manually reviewing the CSV text file. Once imported, I review the variables and field values to ensure it is aligned with the data dictionary provided by the hospital. The structure of the entire data set is then reviewed to prepare for data analysis. Index is added to the data set. It services two purposes, it provides the data set with an ID field for data sets not containing one. It also tracks sort order of records and can be helpful to recover the original sort order if necessary (Larose, Chantal D., and Daniel T. Larose, 2019, Page 31). The redundant or unnecessary columns are dropped from the data frame. The 'Unnamed: 0', 'Customer\_id', 'Interaction', 'UID', 'Lat', 'Lng' are dropped from the data frame because they are not useful for analysis. Variables are renamed to follow a standard naming convention. The survey columns are renamed to provide an accurate description of the columns field values. The data dictionary will need to be updated to reflect the recent changes to the variable names.

The data set is checked for duplicates, misspellings, uniformity, unique characteristics, and missing values when assessing the data set. Duplicated data can alter the skewness of results, causing the analysis to be incorrect. Misspellings are reviewed to ensure accurate descriptions of the variables. It is important to treat misleading variables to make sure the data is consistent and has uniformity across the variables. Reviewing the uniqueness of the data enforces data integrity so that no two rows have the same values. The precision of values is reduced to eliminate unnecessary decimal points, along with data type constraints in string formats that need to be processed as integer data type. This brings uniformity across the variables.

A frequent problem when performing data analysis is finding data missing from the dataset. This issue can impact the conclusion of your analysis. (Larose, Chantal D., and Daniel T. Larose, 2019, Lesson 5: Missing Data). To treat missing values from the dataset, the fillna() method is used to impute numerical variables with the mean or median. For categorical variables, the use of mode to impute missing categorical variables because it provides the value that occurs with the greatest frequency. Utilizing these imputation methods will give us a complete data set to further analyze.

Outliers are identified by displaying graphical visualization to spot values separated from majority of the others (Larose, Chantal D., and Daniel T. Larose, 2019, Lesson 6: Outliers). Box plots and histograms are used to identify the abnormalities in the dataset with visual representations. Z-Scores are used to observe the value’s relationship to mean. Anomalies are identified by how far away it is from the normal distribution. Re-expression of categorical values transforms variables that can be used in principle categorical analysis. In PCA, the variables are grouped to find a meaningful correlation among the groupings.

**C3: Justification of Tools**

  I chose to use Python 3 in Jupyter notebook programming environment to analyze and clean the data set because it is an intuitive and readable syntax. It is also the most used and popular choice for data science professionals (Hayes, Bob, 2018). There are many popular libraries to incorporate graphical visualizations when displaying the code results. The Jupyter has a user-friendly environment, allows for documentation with Markdown format, and produces clean visual aids.

The following libraries are installed to support the data cleaning and analysis:

* Pandas – Load and clean the data set
* NumPy – Work with arrays
* SciPy – Statistical Functions
* Missingno - Visualization series
* Matplotlib – Plot Charts
* Seaborn - Box plots
* Scikit-learn – PCA

A specific example of loading the dataset using the read\_csv() argument from the pandas library. A variable is created for the data frame that can be used later in data analysis.

import pandas as pd

df = pd.read\_csv(r'C:\temp\D206\medical\_raw\_data.csv')

**C4: Provide the Code**

The following code used to identify anomalies in the data. Please see code uploaded.

*# Import python libraries used for analysis*  
import pandas as pd  
import scipy.stats as stats  
import numpy as np  
import missingno as msno  
import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.decomposition import PCA

*# Import raw CSV file into Python panda data frame*  
df = pd.read\_csv(r'C:\temp\D206\medical\_raw\_data.csv')

*# Display max columns to prevent truncation*  
pd.set\_option('display.max\_columns', None)

#### Review detailed information about the variables and structure

*# Review first 5 rows*  
df.head()

*# Print info about the data frame*  
df.info()

*# Create Index for the data frame*   
df['ID'] = pd.Series(range(0,10000))

*# Removed redundant column or columns not useful for analysis*  
df.drop(columns=['Unnamed: 0', 'Customer\_id', 'Interaction', 'UID', 'Lat', 'Lng'], axis=1, inplace=True)  
df.columns

*# Standardize column naming convention and Rename columns to be more descriptive.*  
df.rename( columns = {'CaseOrder':'Case\_order',  
 'Marital':'Marital\_status',  
 'ReAdmis':'Readmitted',  
 'VitD\_supp':'VitD\_supplements',  
 'Soft\_drink':'Habitual\_soft\_drink\_use',  
 'Initial\_admin':'Initial\_admission',  
 'HighBlood':'High\_blood',  
 'BackPain':'Back\_pain',  
 'TotalCharge':'Total\_charge',  
 'Item1':'Timely\_admission',   
 'Item2':'Timely\_treatment',  
 'Item3':'Timely\_visits',  
 'Item4':'Reliability',  
 'Item5':'Options',  
 'Item6':'Hours\_of\_treatment',  
 'Item7':'Courteous\_staff',  
 'Item8':'Evidence\_of\_active\_listening\_from\_doctor',}, inplace=True)  
df.columns

*# Examine duplicate columns*  
df.columns.duplicated().any()

*# Examine duplicate rows*  
df.duplicated().any()

#### Examine categorical variables for misspellings, review unique characteristics, and misleading field values in the data set.

df['State'].unique()

*# Indication of zip codes are not all 5 digits*  
len(df['Zip'].unique())

df['Area'].unique()

df['Timezone'].unique()

df['Timezone'].describe()

dict\_timezone = {'America/Chicago':'UTC -5:00',   
 'America/New\_York':'UTC -4:00',   
 'America/Los\_Angeles':'UTC -7:00',  
 'America/Indiana/Indianapolis':'UTC -4:00',   
 'America/Detroit':'UTC -4:00',  
 'America/Denver':'UTC -6:00',   
 'America/Nome':'UTC -8:00',   
 'America/Anchorage':'UTC -8:00',  
 'America/Phoenix':'UTC -7:00',   
 'America/Boise':'UTC -6:00',   
 'America/Puerto\_Rico':'UTC -4:00',  
 'America/Yakutat':'UTC -8:00',   
 'Pacific/Honolulu':'UTC -10:00',   
 'America/Menominee':'UTC -5:00',  
 'America/Kentucky/Louisville':'UTC -4:00',   
 'America/Indiana/Vincennes':'UTC -4:00',  
 'America/Toronto':'UTC -4:00',   
 'America/Indiana/Marengo':'UTC -4:00',  
 'America/Indiana/Winamac':'UTC -4:00',   
 'America/Indiana/Tell\_City':'UTC -5:00',  
 'America/Sitka':'UTC -8:00',   
 'America/Indiana/Knox':'UTC -5:00',  
 'America/North\_Dakota/New\_Salem':'UTC -5:00',   
 'America/Indiana/Vevay':'UTC -5:00',  
 'America/Adak':'UTC -9:00',   
 'America/North\_Dakota/Beulah':'UTC -5:00'}  
  
df['Timezone'].replace(dict\_timezone, inplace = True)  
df['Timezone']

df['Education'].unique()

df['Job'].unique()

*# High number of unique jobs*  
df['Job'].describe()

df['Employment'].unique()

df['Marital\_status'].unique()

*# Data dictionary indicates non-binary, field values are "Prefer not to answer"*  
df['Gender'].unique()

df['Readmitted'].unique()

df['Initial\_admission'].unique()

df['Services'].unique()

df['Complication\_risk'].unique()

#### Detecting Missing Data

df.isnull().sum()

*# Examine distribution for missing variables*  
df[['Children', 'Age', 'Income','Initial\_days']].hist(figsize = (5,5))  
plt.show()

df[['Anxiety', 'Overweight']].hist(figsize = (5,5))  
plt.show()

df['Habitual\_soft\_drink\_use'].hist(figsize = (5,5))  
plt.show()

#### Detect and Identify Outliers

*# Display histograms of numerical values*  
df.hist(figsize = (15,20))

*# Potential Outliers identified. Create a new data frame. (“How to select Rows and Columns in Pandas Using [],.Loc, 2022)*   
df\_train = df.loc[:,['Population', 'Children', 'Age', 'Income', 'VitD\_levels', 'Doc\_visits', 'Full\_meals\_eaten', 'VitD\_supplements', 'Initial\_days', 'Total\_charge', 'Additional\_charges']]  
df\_train.head()

boxplot=sns.boxplot(x='Population',data=df\_train)

*# Standardize Numeric Field, Query Outliers, Sort values, Plot Z-Score.*  
df\_train['Z\_Score\_Population'] = stats.zscore(df\_train['Population'])  
Population\_outlier = df\_train.query('Z\_Score\_Population > 3 | Z\_Score\_Population < -3')  
Population\_sorted = df\_train.sort\_values(['Z\_Score\_Population'], ascending = False)  
  
plt.hist(df\_train['Z\_Score\_Population'])  
plt.show()

Population\_sorted[['Z\_Score\_Population','Population']].head(n=15)

boxplot=sns.boxplot(x='Children',data=df\_train)

df\_train['Z\_Score\_Children'] = stats.zscore(df\_train['Children'])  
Children\_outliers = df\_train.query('Z\_Score\_Children > 3 | Z\_Score\_Children < -3')  
Children\_sorted = df\_train.sort\_values(['Z\_Score\_Children'], ascending = False)  
  
plt.hist(df\_train['Z\_Score\_Children'])  
plt.show()

Children\_sorted[['Z\_Score\_Children','Children']].head(n=15)

boxplot=sns.boxplot(x='Age',data=df\_train)

df\_train['Z\_Score\_Age'] = stats.zscore(df\_train['Age'])  
Age\_outliers = df\_train.query('Z\_Score\_Age > 3 | Z\_Score\_Age < -3')  
Age\_sorted = df\_train.sort\_values(['Z\_Score\_Age'], ascending = False)  
  
plt.hist(df\_train['Z\_Score\_Age'])  
plt.show()

Age\_sorted[['Z\_Score\_Age','Age']].head(n=15)

boxplot=sns.boxplot(x='Income',data=df\_train)

df\_train['Z\_Score\_Income'] = stats.zscore(df['Income'])  
Income\_outliers = df\_train.query('Z\_Score\_Income > 3 | Z\_Score\_Income < -3')  
Income\_sorted = df\_train.sort\_values(['Z\_Score\_Income'], ascending = False)  
  
plt.hist(df\_train['Z\_Score\_Income'])  
plt.show()

Income\_sorted[['Z\_Score\_Income','Income']].head(n=15)

boxplot=sns.boxplot(x='VitD\_levels',data=df\_train)

df\_train['Z\_Score\_VitD\_levels'] = stats.zscore(df\_train['VitD\_levels'])  
VitD\_levels\_outliers = df\_train.query('Z\_Score\_VitD\_levels > 3 |Z\_Score\_VitD\_levels < -3')  
VitD\_levels\_sorted = df\_train.sort\_values(['Z\_Score\_VitD\_levels'], ascending = False)  
  
plt.hist(df\_train['Z\_Score\_VitD\_levels'])  
plt.show()

VitD\_levels\_sorted[['Z\_Score\_VitD\_levels','VitD\_levels']].head(n=15)

boxplot=sns.boxplot(x='Doc\_visits',data=df\_train)

df\_train['Z\_Score\_Doc\_visits'] = stats.zscore(df\_train['Doc\_visits'])  
Doc\_visits\_outliers = df\_train.query('Z\_Score\_Doc\_visits > 3 |Z\_Score\_Doc\_visits < -3')  
 *# Ascending to see less than -3*  
Doc\_visits\_sorted = df\_train.sort\_values(['Z\_Score\_Doc\_visits'], ascending = True)   
  
plt.hist(df\_train['Z\_Score\_Doc\_visits'])  
plt.show()

Doc\_visits\_sorted[['Z\_Score\_Doc\_visits','Doc\_visits']].head(n=15)

boxplot=sns.boxplot(x='Full\_meals\_eaten',data=df\_train)

df\_train['Z\_Score\_Full\_meals\_eaten'] = stats.zscore(df\_train['Full\_meals\_eaten'])  
Full\_meals\_eaten\_outliers = df\_train.query('Z\_Score\_Full\_meals\_eaten > 3 | Z\_Score\_Full\_meals\_eaten < -3')  
Full\_meals\_eaten\_sorted = df\_train.sort\_values(['Z\_Score\_Full\_meals\_eaten'], ascending = False)  
  
plt.hist(df\_train['Z\_Score\_Full\_meals\_eaten'])  
plt.show()

Full\_meals\_eaten\_sorted[['Z\_Score\_Full\_meals\_eaten','Full\_meals\_eaten']].head(n=15)

boxplot=sns.boxplot(x='VitD\_supplements',data=df\_train)

df\_train['Z\_Score\_VitD\_supplements'] = stats.zscore(df\_train['VitD\_supplements'])  
VitD\_supplements\_outliers = df\_train.query('Z\_Score\_VitD\_supplements > 3 | Z\_Score\_VitD\_supplements < -3')  
VitD\_supplements\_sorted = df\_train.sort\_values(['Z\_Score\_VitD\_supplements'], ascending = False)  
  
plt.hist(df\_train['Z\_Score\_VitD\_supplements'])  
plt.show()

VitD\_supplements\_sorted[['Z\_Score\_VitD\_supplements','VitD\_supplements']].head(n=15)

boxplot=sns.boxplot(x='Initial\_days',data=df\_train)

df\_train['Z\_Score\_Initial\_days'] = stats.zscore(df\_train['Initial\_days'])  
Initial\_days\_outliers = df\_train.query('Z\_Score\_Initial\_days > 3 | Z\_Score\_Initial\_days < -3')  
Initial\_days\_sorted = df\_train.sort\_values(['Z\_Score\_Initial\_days'], ascending = False)  
  
plt.hist(df\_train['Z\_Score\_Initial\_days'])  
plt.show()

Initial\_days\_sorted[['Z\_Score\_Initial\_days','Initial\_days']].head(n=15)

boxplot=sns.boxplot(x='Total\_charge',data=df\_train)

df\_train['Z\_Score\_Total\_charge'] = stats.zscore(df\_train['Total\_charge'])  
Total\_charge\_outliers = df\_train.query('Z\_Score\_Total\_charge > 3 | Z\_Score\_Total\_charge < -3')  
Total\_charg\_sorted = df\_train.sort\_values(['Z\_Score\_Total\_charge'], ascending = False)  
  
plt.hist(df\_train['Z\_Score\_Total\_charge'])  
plt.show()

Total\_charg\_sorted[['Z\_Score\_Total\_charge','Total\_charge']].head(n=15)

#### Re-expression of catagorical varibles:

*# Yes or No will be converted to 1 or 0. Categorical columns expressed ordinally.*

df.loc[:,['Education','Readmitted','Habitual\_soft\_drink\_use','High\_blood','Stroke','Arthritis','Diabetes','Hyperlipidemia','Back\_pain','Allergic\_rhinitis','Reflux\_esophagitis','Asthma','Complication\_risk','Anxiety','Overweight']]

df = df.replace(to\_replace={'Readmitted':{'Yes':1,'No':0}})

df = df.replace(to\_replace={'Habitual\_soft\_drink\_use':{'Yes':1,'No':0}})

df = df.replace(to\_replace={'High\_blood':{'Yes':1,'No':0}})

df = df.replace(to\_replace={'Stroke':{'Yes':1,'No':0}})

df = df.replace(to\_replace={'Arthritis':{'Yes':1,'No':0}})

df = df.replace(to\_replace={'Diabetes':{'Yes':1,'No':0}})

df = df.replace(to\_replace={'Hyperlipidemia':{'Yes':1,'No':0}})

df = df.replace(to\_replace={'Back\_pain':{'Yes':1,'No':0}})

df = df.replace(to\_replace={'Allergic\_rhinitis':{'Yes':1,'No':0}})

df = df.replace(to\_replace={'Reflux\_esophagitis':{'Yes':1,'No':0}})

df = df.replace(to\_replace={'Asthma':{'Yes':1,'No':0}})

df = df.replace(to\_replace={'Complication\_risk':{'Low':1,'Medium':2,'High':3}})

dict\_edu = {'Doctorate Degree': 11,  
 "Master's Degree": 10,  
 "Bachelor's Degree": 9,  
 "Associate's Degree": 8,  
 'Professional School Degree': 7,  
 'Some College, 1 or More Years, No Degree': 6,  
 'Some College, Less than 1 Year': 5,  
 'Regular High School Diploma': 4,  
 'GED or Alternative Credential': 3,  
 '9th Grade to 12th Grade, No Diploma': 2,  
 'Nursery School to 8th Grade': 1,  
 'No Schooling Completed': 0}  
  
df['Education'] = df['Education'].replace(dict\_edu)

df.loc[:,['Education','Readmitted','Habitual\_soft\_drink\_use','High\_blood','Stroke','Arthritis','Diabetes','Hyperlipidemia','Back\_pain','Allergic\_rhinitis','Reflux\_esophagitis','Asthma','Complication\_risk','Anxiety','Overweight']]

**Part III: Data Cleaning**

**D1: Cleaning Findings**

 Analysis of the data set produced the following observations:

* I identified six variables that were redundant or unnecessary for further analysis. Several of the variable names did not follow a naming convention or did not sufficiently describe the context of the column names. The survey response columns were listed as Item 1 through Item 8, instead of a more descriptive variable name.
* An index was not present in the data frame and was added so that it could be used for further analysis.
* Duplicated columns and rows were checked, and none were found. No misspellings of categorical variables found.
* After inspecting the data set for unique characteristics, the gender column produced different field values compared to the data dictionary. The value of “prefer not to answer” did not match with “non-binary,” as specified in the data dictionary.
* The time zone variable contained twenty-six unique values. Many of the time zones appear to be granular locations for the time zone. They were reduced to UTC time zones. (“timeanddate.com”,2022)
* The job variable contained 639 unique values. Further exploratory analysis will be need to performed to determine the validity of the jobs titles.
* The Children, Age, Overweight, and Anxiety columns are incorrectly expressed as float data types. The Overweight and Anxiety variables are represented as yes or no in the data dictionary.
* The decimal precision for VitD\_levels, Initial\_days, Total\_charge, and Additional\_charges need to be reduced to 2 decimal places.
* The Zip code column did not display the leading zeros. They were cross-checked with the United States zip codes to confirm they were valid zip codes (“U.S. ZIP Codes “, 2022<https://www.unitedstateszipcodes.org/>).
* There is a large number of missing values for Age, Children, Income, Initial\_days, Anxiety, Overweight, and Soft\_drink columns.
* Eleven potential outliers were detected using histograms, Z-score, and box plots for quantitative variables. Nine variables were found to contain outliers.

**D2: Justification of Mitigation Methods**

 The six columns, Unnamed: 0, Customer\_id, Interaction, UID, Lat, Lng do not provide any meaningful information for data analysis, so they were dropped from the data frame. A new variable named ID was used as the index to track sort order of records and can be helpful to recover the original sort order for future analysis. The columns CaseOrder, Marital, ReAdmis, VitD\_Supp, Soft\_drink, HighBlood, BackPain, and TotalCharge were renamed to follow a standard naming convention across all variables. The survey columns were renamed to provide an accurate description of the variables.

List of original and renamed column names:

| **Original Variable Name** | **New Variable Name** |
| --- | --- |
| CaseOrder | Case\_order |
| Marital | Marital\_status |
| ReAdmis | Readmitted |
| VitD\_supp | VitD\_supplements |
| Soft\_drink | Habitual\_soft\_drink\_use |
| Initial\_admin | Initial\_admission |
| HighBlood | High\_blood |
| BackPain | Back\_pain |
| TotalCharge | Total\_charge |
| Item1 | Timely\_admission |
| Item2 | Timely\_treatment |
| Item3 | Timely\_visits |
| Item4 | Reliability |
| Item5 | Options |
| Item6 | Hours\_of\_treatment |
| Item7 | Courteous\_staff |
| Item8 | Evidence\_of\_active\_listening\_from\_doctor |

The time zone variable contained twenty-six unique values; they are similar but with finer details of the location. Accurate time zones could also be acquired by using the City, State, and Zip code columns. The values were replaced to UTC time zones for standardization.

The Children, Age, Overweight, and Anxiety columns were incorrectly categorized as float data types, so they were converted to integers data types. The precision of decimals was reduced for numerical values of VitD\_levels, Initial\_days, Total\_charge, and Additional\_charges for standardization and appropriate currency values. Leading zeros for zip codes were added to accurately display the correct US zip code.

Missing values were treated with univariate imputation. The mean was used to impute Age due to its uniform distribution of data. The median was used to impute numerical values for Children, Income and Initial\_days based on skewed distribution. For categorical variables for Anxiety, Overweight, and Habitual\_soft\_drink\_use, mode was used to impute missing values.

A new data frame named df\_train was created to store the outliers. The numerical variables were displayed using histograms to quickly identify the eleven potential outliers. Three variables, Age, Doc\_Visits, and Initial\_days contained no outliers. The eight remaining variables did contain outliers, they were 'Population, Children, Income, VitD\_levels, Full\_meals\_eaten, VitD\_supplements, Total\_charge, and Additional\_charges. For each variable, the outliers numerical value were first standardized. It was then queried to find the Z-score of +/-3, to identify values outside the range of the normal distribution. Box plots were used to visualize the distribution of values that fall outsize the minimum and maximum values. The eight outliers are determined to be within a logical range and will be kept in the data set. Categorical variables that contained yes or no values were re-expressed as numerical values, with values of 1 or 0 respectively. This will allow statistical analysis used in PCA.

**D3: Summary of the Outcomes**

* Six columns not useful or redundant were dropped from the data frame.
* The column names were standardized and changed to provide a more descriptive variable name.
* A new column ID was created and used as the index. This provides the ability to track sort order of records and recover from original sort order in future analysis
* Duplicates, misspellings, unique characteristics, and misleading field values were examined to ensure data consistency and integrity of the data set.
* Time zone categorical values were updated to UTC time zone.
* Float data type converted to integers for accuracy of context.
* Decimal precision reduced to two decimal places.
* Leading zeros added to zip codes to display the accurate patient zip codes.
* Missing numerical values were replaced with the mean, median, or mode allows for consistency in statistical analysis and will reduce errors encountered in later analysis
* Eleven potential outliers were identified, nine variables contained outliers.
* Categorical variables were re-expressed to numerical values that can be used for Principal Component Analysis.
* The data dictionary will need to be updated to show the new variable names and numerical values.
* The decision to leave the outliers in the data set is due to the values being within a valid logical range.
* The data set has been cleaned to provide better clarity that can be used for further exploratory data analysis.

**D4: Mitigation Code**

The following code used to treat anomalies in the data. Please see code uploaded.

*# Import python libraries used for analysis*  
import pandas as pd  
import scipy.stats as stats  
import numpy as np  
import missingno as msno  
import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.decomposition import PCA

*# Import raw CSV file into Python panda data frame*  
df = pd.read\_csv(r'C:\temp\D206\medical\_raw\_data.csv')

#### Treat missing values by filling nulls with mean, median, or mode.

*# Use mean to impute uniform distribution*  
df['Age'].fillna(df['Age'].mean(), inplace=True)  
  
*# Use median to impute numerical variable skewed*  
df['Children'].fillna(df['Children'].median(), inplace=True)  
df['Income'].fillna(df['Income'].median(), inplace=True)  
df['Initial\_days'].fillna(df['Initial\_days'].median(), inplace=True)  
  
*# Use mode to impute missing categorical variables.*  
df['Anxiety'] = df['Anxiety'].fillna(df['Anxiety'].mode()[0])  
df['Overweight'] = df['Overweight'].fillna(df['Overweight'].mode()[0])  
df['Habitual\_soft\_drink\_use'] = df['Habitual\_soft\_drink\_use'].fillna(df['Habitual\_soft\_drink\_use'].mode()[0])

*# Visualization of variables*  
msno.matrix(df, fontsize=9,labels=True)  
plt.title('Complete Dataset')  
plt.show()

#### Convert data type Floats to Ints

df[['Children','Age','Overweight','Anxiety']].head()

df['Children'] = df['Children'].astype(int)

df['Age'] = df['Age'].astype(int)

df['Overweight'] = df['Overweight'].astype(int)

df['Anxiety'] = df['Anxiety'].astype(int)  
  
df[['Children','Age','Overweight','Anxiety']].head()

#### Convert decimal precision 2 decimal places

df[['VitD\_levels','Initial\_days','Total\_charge','Additional\_charges']].head()

df['VitD\_levels'] = df['VitD\_levels'].round(decimals = 2)

df['Initial\_days'] = df['Initial\_days'].round(decimals = 2)

df['Total\_charge'] = df['Total\_charge'].round(decimals = 2)

df['Additional\_charges'] = df['Additional\_charges'].round(decimals = 2)  
  
df[['VitD\_levels','Initial\_days','Total\_charge','Additional\_charges']].head()

#### Search for zip codes with less than 5 digits

*# Convert Zip to string data type*   
df['Zip'] = df['Zip'].astype(str)  
  
*#df['Zip'][df['Zip'].apply(len) == 5] # 9,277 zip codes length of 5*  
df['Zip'][df['Zip'].apply(len) != 5] *# 723 zip codes not equal to length of 5*

#### Fill leading zeros that have been omitted.

df['Zip'] = df['Zip'].str.zfill(5)  
print(df['Zip'])

*# Validate the changes*  
df['Zip'][df['Zip'].apply(len) != 5]

#### Detect and Identify Outliers

*# Display histograms of numerical values*  
df.hist(figsize = (15,20))

*# Potential Outliers identified. Create a new data frame.*  
df\_train = df.loc[:,['Population', 'Children', 'Age', 'Income', 'VitD\_levels', 'Doc\_visits', 'Full\_meals\_eaten', 'VitD\_supplements', 'Initial\_days', 'Total\_charge', 'Additional\_charges']]  
df\_train.head()

boxplot=sns.boxplot(x='Population',data=df\_train)

*# Standardize Numeric Field, Query Outliers, Sort values, Plot Z-Score.*  
df\_train['Z\_Score\_Population'] = stats.zscore(df\_train['Population'])  
Population\_outlier = df\_train.query('Z\_Score\_Population > 3 | Z\_Score\_Population < -3')  
Population\_sorted = df\_train.sort\_values(['Z\_Score\_Population'], ascending = False)  
  
plt.hist(df\_train['Z\_Score\_Population'])  
plt.show()

Population\_sorted[['Z\_Score\_Population','Population']].head(n=15)

boxplot=sns.boxplot(x='Children',data=df\_train)

df\_train['Z\_Score\_Children'] = stats.zscore(df\_train['Children'])  
Children\_outliers = df\_train.query('Z\_Score\_Children > 3 | Z\_Score\_Children < -3')  
Children\_sorted = df\_train.sort\_values(['Z\_Score\_Children'], ascending = False)  
  
plt.hist(df\_train['Z\_Score\_Children'])  
plt.show()

Children\_sorted[['Z\_Score\_Children','Children']].head(n=15)

boxplot=sns.boxplot(x='Age',data=df\_train)

df\_train['Z\_Score\_Age'] = stats.zscore(df\_train['Age'])  
Age\_outliers = df\_train.query('Z\_Score\_Age > 3 | Z\_Score\_Age < -3')  
Age\_sorted = df\_train.sort\_values(['Z\_Score\_Age'], ascending = False)  
  
plt.hist(df\_train['Z\_Score\_Age'])  
plt.show()

Age\_sorted[['Z\_Score\_Age','Age']].head(n=15)

boxplot=sns.boxplot(x='Income',data=df\_train)

df\_train['Z\_Score\_Income'] = stats.zscore(df['Income'])  
Income\_outliers = df\_train.query('Z\_Score\_Income > 3 | Z\_Score\_Income < -3')  
Income\_sorted = df\_train.sort\_values(['Z\_Score\_Income'], ascending = False)  
  
plt.hist(df\_train['Z\_Score\_Income'])  
plt.show()

Income\_sorted[['Z\_Score\_Income','Income']].head(n=15)

boxplot=sns.boxplot(x='VitD\_levels',data=df\_train)

df\_train['Z\_Score\_VitD\_levels'] = stats.zscore(df\_train['VitD\_levels'])  
VitD\_levels\_outliers = df\_train.query('Z\_Score\_VitD\_levels > 3 |Z\_Score\_VitD\_levels < -3')  
VitD\_levels\_sorted = df\_train.sort\_values(['Z\_Score\_VitD\_levels'], ascending = False)  
  
plt.hist(df\_train['Z\_Score\_VitD\_levels'])  
plt.show()

VitD\_levels\_sorted[['Z\_Score\_VitD\_levels','VitD\_levels']].head(n=15)

boxplot=sns.boxplot(x='Doc\_visits',data=df\_train)

df\_train['Z\_Score\_Doc\_visits'] = stats.zscore(df\_train['Doc\_visits'])  
Doc\_visits\_outliers = df\_train.query('Z\_Score\_Doc\_visits > 3 |Z\_Score\_Doc\_visits < -3')  
 *# Ascending to see less than -3*  
Doc\_visits\_sorted = df\_train.sort\_values(['Z\_Score\_Doc\_visits'], ascending = True)   
  
plt.hist(df\_train['Z\_Score\_Doc\_visits'])  
plt.show()

Doc\_visits\_sorted[['Z\_Score\_Doc\_visits','Doc\_visits']].head(n=15)

boxplot=sns.boxplot(x='Full\_meals\_eaten',data=df\_train)

df\_train['Z\_Score\_Full\_meals\_eaten'] = stats.zscore(df\_train['Full\_meals\_eaten'])  
Full\_meals\_eaten\_outliers = df\_train.query('Z\_Score\_Full\_meals\_eaten > 3 | Z\_Score\_Full\_meals\_eaten < -3')  
Full\_meals\_eaten\_sorted = df\_train.sort\_values(['Z\_Score\_Full\_meals\_eaten'], ascending = False)  
  
plt.hist(df\_train['Z\_Score\_Full\_meals\_eaten'])  
plt.show()

Full\_meals\_eaten\_sorted[['Z\_Score\_Full\_meals\_eaten','Full\_meals\_eaten']].head(n=15)

boxplot=sns.boxplot(x='VitD\_supplements',data=df\_train)

df\_train['Z\_Score\_VitD\_supplements'] = stats.zscore(df\_train['VitD\_supplements'])  
VitD\_supplements\_outliers = df\_train.query('Z\_Score\_VitD\_supplements > 3 | Z\_Score\_VitD\_supplements < -3')  
VitD\_supplements\_sorted = df\_train.sort\_values(['Z\_Score\_VitD\_supplements'], ascending = False)  
  
plt.hist(df\_train['Z\_Score\_VitD\_supplements'])  
plt.show()

VitD\_supplements\_sorted[['Z\_Score\_VitD\_supplements','VitD\_supplements']].head(n=15)

boxplot=sns.boxplot(x='Initial\_days',data=df\_train)

df\_train['Z\_Score\_Initial\_days'] = stats.zscore(df\_train['Initial\_days'])  
Initial\_days\_outliers = df\_train.query('Z\_Score\_Initial\_days > 3 | Z\_Score\_Initial\_days < -3')  
Initial\_days\_sorted = df\_train.sort\_values(['Z\_Score\_Initial\_days'], ascending = False)  
  
plt.hist(df\_train['Z\_Score\_Initial\_days'])  
plt.show()

Initial\_days\_sorted[['Z\_Score\_Initial\_days','Initial\_days']].head(n=15)

boxplot=sns.boxplot(x='Total\_charge',data=df\_train)

df\_train['Z\_Score\_Total\_charge'] = stats.zscore(df\_train['Total\_charge'])  
Total\_charge\_outliers = df\_train.query('Z\_Score\_Total\_charge > 3 | Z\_Score\_Total\_charge < -3')  
Total\_charg\_sorted = df\_train.sort\_values(['Z\_Score\_Total\_charge'], ascending = False)  
  
plt.hist(df\_train['Z\_Score\_Total\_charge'])  
plt.show()

Total\_charg\_sorted[['Z\_Score\_Total\_charge','Total\_charge']].head(n=15)

#### Re-expression of catagorical varibles:

*# Yes or No will be converted to 1 or 0. Categorical columns expressed ordinally.*

df.loc[:,['Education','Readmitted','Habitual\_soft\_drink\_use','High\_blood','Stroke','Arthritis','Diabetes','Hyperlipidemia','Back\_pain','Allergic\_rhinitis','Reflux\_esophagitis','Asthma','Complication\_risk','Anxiety','Overweight']]

df = df.replace(to\_replace={'Readmitted':{'Yes':1,'No':0}})

df = df.replace(to\_replace={'Habitual\_soft\_drink\_use':{'Yes':1,'No':0}})

df = df.replace(to\_replace={'High\_blood':{'Yes':1,'No':0}})

df = df.replace(to\_replace={'Stroke':{'Yes':1,'No':0}})

df = df.replace(to\_replace={'Arthritis':{'Yes':1,'No':0}})

df = df.replace(to\_replace={'Diabetes':{'Yes':1,'No':0}})

df = df.replace(to\_replace={'Hyperlipidemia':{'Yes':1,'No':0}})

df = df.replace(to\_replace={'Back\_pain':{'Yes':1,'No':0}})

df = df.replace(to\_replace={'Allergic\_rhinitis':{'Yes':1,'No':0}})

df = df.replace(to\_replace={'Reflux\_esophagitis':{'Yes':1,'No':0}})

df = df.replace(to\_replace={'Asthma':{'Yes':1,'No':0}})

df = df.replace(to\_replace={'Complication\_risk':{'Low':1,'Medium':2,'High':3}})

dict\_edu = {'Doctorate Degree': 11,  
 "Master's Degree": 10,  
 "Bachelor's Degree": 9,  
 "Associate's Degree": 8,  
 'Professional School Degree': 7,  
 'Some College, 1 or More Years, No Degree': 6,  
 'Some College, Less than 1 Year': 5,  
 'Regular High School Diploma': 4,  
 'GED or Alternative Credential': 3,  
 '9th Grade to 12th Grade, No Diploma': 2,  
 'Nursery School to 8th Grade': 1,  
 'No Schooling Completed': 0}  
  
df['Education'] = df['Education'].replace(dict\_edu)

df.loc[:,['Education','Readmitted','Habitual\_soft\_drink\_use','High\_blood','Stroke','Arthritis','Diabetes','Hyperlipidemia','Back\_pain','Allergic\_rhinitis','Reflux\_esophagitis','Asthma','Complication\_risk','Anxiety','Overweight']]

**D5: Clean Data**

Cleaned data set has been exported to CSV file and uploaded.

**D6: Limitations**

The biggest limitation when treating the data is not being able to discuss issues or ask questions from a hospital resource. Not being able to discuss and determine why missing data is present in the data set. Simply, it could be that data errors exist, and training needs to be provided or do they exist because that is how they are consumed by another application. Clarifying why column names are the way they are and determining a naming convention. The medical data would be more meaningful if a medical professional would help interpret or assess the outliers. I would be able to gain clarity in the data set, which could change the approach of mitigating the data.

**D7: Impact of the Limitations**

 The impact of not being able to discuss the data issues with a hospital resource could impact how the data is mitigated. Without having any medical professional experience, I am the only one deciding if the anomalies should be present in the data set, how the data is cleaned, variable names changed. This could impact others who consume the data into another application that expects certain values or data types in the data set, even though they are inaccurately described.

 There is no indication as to what factors caused the patient to be readmitted to the hospital. It would be difficult to determine if the readmission is due to the key medical conditions or a result of chronic condition. It can provide greater insight as to what medical conditions are causing the higher rates of readmission.

**Part IV: PCA**

**E1: Principal Components**

List of the principal components in the data set.

* Population
* Children
* Age
* Education
* Income
* VitD\_levels
* Doc\_visits
* Full\_meals\_eaten
* VitD\_supplements
* Initial\_days
* Total\_charge
* Additional\_charges
* Timely\_admission
* Timely\_treatment
* Timely\_visits
* Reliability
* Options
* Hours\_of\_treatment
* Courteous\_staff
* Evidence\_of\_active\_listening\_from\_doctor

**E2: Criteria Used**

 There are twenty continuous numerical variables selected for principal component analysis. The variables were normalized, then converted to components used for analysis. The components Loadings are then outputted to table. Components are then selected by defining eigenvalues, then performing a scree plot. Seven principal components with eigenvalues of 1.00 or greater are retained and assigned to a new pca\_reduced variable.

**E3: Benefits**

 The organization can benefit from the result of PCA by reducing the number of variables to the most important related groupings of variables (Larose, Chantal D., and Daniel T. Larose, 2019, Lesson 7: PCA). This provides a more manageable numerical grouping of values that have components in predicating crucial factors in predicting a patient's readmission. This will help make analysis and machine learning algorithms more efficient. It can help the organization to quickly produce valuable insights for decision-making and strategic planning.

**Part IV. Supporting Documents**

**F: Video**

Panapto Video Link

**G: Sources for Third-Party Code**

“How to Select Rows and Columns in Pandas Using [ ], .Loc, Iloc, .at and .Iat.” KDnuggets, https://www.kdnuggets.com/how-to-select-rows-and-columns-in-pandas-using-loc-iloc-at-and-iat.html/. Accessed 20 June 2022.

Loops - Learn Python - Free Interactive Python Tutorial. https://www.learnpython.org/. Accessed 1 July 2022.

How to Replace Values in Pandas DataFrame - Data to Fish. 10 May 2020, https://datatofish.com/replace-values-pandas-dataframe/. Accessed 1 July 2022.

Python Dictionaries. https://www.w3schools.com/python/python\_dictionaries.asp. Accessed 15 July 2022.

Li, Guangyang. “Answer to ‘Add Leading Zeros to Strings in Pandas Dataframe.’” Stack Overflow, https://stackoverflow.com/a/31553791. 22 July 2015,

**H: Sources**

Hayes, Bob (2018, January 2) Most Used Data Science Tools and Technologies in 2017 and What to Expect for 2018. http://businessoverbroadway.com/2018/01/02/most-used-data-science-tools-and-technologies-in-2017-and-what-to-expect-for-2018/. Accessed 10, May 2022.

Larose, Chantal D., and Daniel T. Larose. Data Science Using Python and R. John Wiley & Sons, Inc, 2019. Page 31. Retrieved June 1, 2022.

Larose, Chantal D., and Daniel T. Larose. Data Science Using Python and R. John Wiley & Sons, Inc, 2019. Lesson 5: Missing Data. Retrieved June 2, 2022.

Larose, Chantal D., and Daniel T. Larose. Data Science Using Python and R. John Wiley & Sons, Inc, 2019. Lesson 6: Outlier Detection. Retrieved June 28, 2022.

Larose, Chantal D., and Daniel T. Larose. Data Science Using Python and R. John Wiley & Sons, Inc, 2019. Lesson 6: Outlier Detection. Retrieved June 28, 2022.

Larose, Chantal D., and Daniel T. Larose. Data Science Using Python and R. John Wiley & Sons, Inc, 2019. Lesson 7: Principal Component Analysis. Retrieved June 30, 2022.

Timeanddate.com. https://www.timeanddate.com/. Accessed 1 July 2022.

“U.S. ZIP Codes: Free ZIP Code Map and Zip Code Lookup.” UnitedStatesZipCodes, https://www.unitedstateszipcodes.org. Accessed 1 July 2022.