Data Cleaning – D206
Western Governor's University
Performance Assessment

Matthew Morgan
Student ID: 010471280

9/7/2022

Part I: Research Question

A. For this project I chose to approach it with the following research question in mind: How can we modify patient outcomes to reduce the number of readmissions? In the information document provided for the medical data it was noted that, "many hospitals are overconfident and underprepared" to handle readmissions ultimately leading to fines. If the data can be used in a way to show a relationship between certain components that the hospital has control over and readmission rates we can begin looking for ways to prevent readmissions.

B. The following variables are contained in the dataset:

CaseOrder: Quantitative, A variable to maintain the original order of the data file (ex: 1)
Customer_id: Quantitative, A unique patient identifier (ex: C412403)
Interaction: Quantitative, Unique ID related to patient interaction (ex: 8cd49b13-f45a-4b47-a2bd-173ffa932c2f)
UID: Quantitative, Same as above (ex: 3a83ddb66e2ae73798bdf1d705dc0932)
City: Qualitative, Patient city of residence from billing statement (ex: Eva)
State: Qualitative, Patient stat of residence from billing statement (ex: AL)
County: Qualitative, Patient county of residence from billing statement (ex: Morgan)
Zip: Quantitative, Patient zip of residence from billing statement (ex: 35621)
Latitude: Quantitative, Latitudinal GPS coordinates of patient residence (ex: 34.3496)
Longitude: Quantitative, Longitudinal GPS coordinates of patient residence (ex: -86.7251)
Population: Quantitative, Based on census data, total population within a mile radius of patient (ex: 2951)
Area: Qualitative, Area type, based on unofficial census data (ex: Suburban)
TimeZone: Qualitative, Time of patient residence based on patient provided data (ex: America/Chicago)
Job: Qualitative, Patient's occupation or primary insurance holder (ex: Psychologist, sport and exercise)
Children: Quantitative, Number of children as reported in admissions (ex: 1)
Age: Quantitative, Age of patient as reported in admissions (ex: 53)
Education: Qualitative, Level of education as reported in admissions (ex: Master's Degree)
Income: Quantitative, Annual income as reported in admissions (ex: 86575.93)
Marital: Qualitative, Marital status of patient (ex: Married)
Gender: Qualitative, Patient self-identification (ex: Male)
ReAdmis: Qualitative, Whether patient ended up readmitted within a month of release (ex: No)
VitD_levels: Quantitative, Patient's vitamin D levels (ex: 17.80233)
Doc_visits: Quantitative, Number of times the patients primary physician visited during hospitalization (ex: 6)
Full_meals_eaten: Quantitative, How many full meals the patient had while hospitalized (partially eaten meals count as 0, patients could have more meals if requested) (ex: 0)
VitD_supp: Quantitative, Number of administrations of Vitamin D the patient received (ex: 0)
Soft_drink: Qualitative, Yes if patient drinks 3 or more soft drinks in a day, no otherwise (ex: No)
Initial_Admin: Qualitative, Why patient was admitted into the hospital (Emergency, elective, observation) (ex: Emergency Admission)
HighBlood: Qualitative, Yes if patient has high blood pressure, no otherwise (ex: Yes)
Stroke: Qualitative, Yes if patient has had a stroke, no otherwise (ex: Yes)

Complication_risk: Qualitative, Assessed for level of complication risk as low, medium, or high (ex: Low)
Overweight: Qualitative, Yes if patient is overweight based on age, gender, and height, no otherwise (ex: Yes)
Arthritis: Qualitative, Yes if patient has arthritis, no otherwise (ex: Yes)
Diabetes: Qualitative, Yes if patient has diabetes, no otherwise (ex: Yes)
Hyperlipidemia: Qualitative, Yes if patient has hyperlipidemia, no otherwise (ex: Yes)
BackPain: Qualitative, Yes if patient has chronic back pain, no otherwise (ex: Yes)
Anxiety: Qualitative, Yes if patient has anxiety disorder, no otherwise (ex: Yes)
Allergic_rhinitis: Qualitative, Yes if patient has allergic rhinitis, no otherwise (ex: Yes)
Reflux_esophagitis: Qualitative, Yes if patient has reflux esophagitis, no otherwise (ex: Yes)
Asthma: Qualitative, Yes if patient has asthma, no otherwise (ex: Yes)
Services: Qualitative, Primary service rendered to patient (blood work, intravenous, CT Scan, MRI) (ex: Blood Work)
Initial_days: Quantitative, Number of days spent in hospital during initial visit (ex: 10.58576971)
TotalCharge: Quantitative, Amount charged to patient daily. Calculated by averaging over the patients entire visit. Does not include specialized treatments (ex: 3191.049)
Additional_charges: Quantitative, Average amount charged for specialized treatments (ex: 17939.4)
Items 1-8 represent a survey given to patients after their stay where they rate the most important factors on a scale of 1-8 (1 being most important, 8 being least important)
Item 1: Quantitative, Time to admit (ex: 1)
Item 2: Quantitative, Time to treatment (ex: 2)
Item 3: Quantitative, Timely visits (ex: 3)
Item 4: Quantitative, Reliability (ex: 4)
Item 5: Quantitative, Options (ex: 5)
Item 6: Quantitative, Hours of treatment (ex: 6)
Item 7: Quantitative, Courteous staff (ex: 7)
Item 8: Quantitative, Evidence of active listening from doctor (ex: 8)

Part II: Data-Cleaning Plan

C1. To begin cleaning the data first I had to take an inventory of what is in the data set. I started with df.info() To give me a readout of the columns, and tell me the data types they contain. After that I ran df.duplicated() to detect for duplicate rows. This reported no duplicate rows so I knew I could proceed without having to drop any rows in the dataset. Following that I checked for missing values with df.isnull().sum() which sums up the total null values in each column. After that I produced a visualization for missing data in the dataset and also produced box plots for all the quantitative variables in the data set. I used the boxplots to check for outliers. Finally I produced bar graphs for all of the columns with null values to visualize the distribution shape so I could appropriately fill in the missing data later. I also had to re-express variables in a lot of columns including Education, Marital, Gender, ReAdmis, Soft_drink, Initial_admin, HighBlood, Stroke, Complication_risk, Arthritis, Diabetes, Hyperlipidemia, BackPain, Allergic_rhinitis, Reflux_esophagitis, Asthma, Services. The original variables were all qualitative variables and not able to be used in PCA. I converted them into quantitative variables so they could be used in PCA if wanted or needed.

C2. I used the methods mentioned in C1 based on 2 sources. The data camp learning modules

and the WGU course material (Larose & Larose, 2019). Between the two they gave me a solid foundation for tackling a project like this and helping me explore the dataset before making any modifications.

The essential steps for cleaning this dataset was to get an overall look at the data with df.info(), check for duplicates and fix any that occur, count null values and fill in the missing data, and re-express categorical variables. The ultimate purpose of this approach is to make sure all data is quantitative in nature to eventually be used to run an analysis. If this approach is not followed and principle data is not expressed quantitatively an analysis cannot be run.

C3. For this assignment I used Python as I am more familiar with it and have been learning it since before starting the program. I used the following packages pandas, numpy, missingno, scipy.stats, seaborn, and PCA.

Pandas and numpy provide libraries to automate a lot of processes involved in exploring and modifying data sets.

Missingno provided me library to produce the missing data matrix.

Scipy.stats provided a library with which to perform statistical functions.

Seaborn allowed me to produce boxplots and explore the possibility of outliers.

Finally, PCA from sklearn.decomposition provided me a library with which to perform my final PCA with.

C 4. See code/script attached as well as the code pasted here:

```
import pandas as pd

import numpy as np

import missingno as msno



import matplotlib.pyplot as plt

from pandas import DataFrame

import scipy.stats as stats

import seaborn

from sklearn.decomposition import PCA

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

```
#Object or string is qualitative, int64 or float64 is quantitative

df.info()

#Detect duplicate rows

print(df.duplicated())

#Detect null values

print(df.isnull().sum())

#Missing data matrix showing where null values exist

msno.matrix(df, labels=True)

plt.title('Missing Data Matrix')

plt.show()

#Detection of outliers for quantitative values

boxplot=seaborn.boxplot(x='Children',data=df)

plt.show()

boxplot=seaborn.boxplot(x='Age',data=df)

plt.show()

boxplot=seaborn.boxplot(x='Income',data=df)

plt.show()

boxplot=seaborn.boxplot(x='VitD_levels',data=df)

plt.show()

boxplot=seaborn.boxplot(x='Doc_visits',data=df)

plt.show()

boxplot=seaborn.boxplot(x='Full_meals_eaten',data=df)

plt.show()

boxplot=seaborn.boxplot(x='VitD_supp',data=df)
```

```
plt.show()

boxplot=seaborn.boxplot(x='Overweight',data=df)

plt.show()

boxplot=seaborn.boxplot(x='Anxiety',data=df)

plt.show()

boxplot=seaborn.boxplot(x='Initial_days',data=df)

plt.show()

boxplot=seaborn.boxplot(x='TotalCharge',data=df)

plt.show()

boxplot=seaborn.boxplot(x='Additional_charges',data=df)

plt.show()

boxplot=seaborn.boxplot(x='Item1',data=df)

plt.show()

boxplot=seaborn.boxplot(x='Item2',data=df)

plt.show()

boxplot=seaborn.boxplot(x='Item3',data=df)

plt.show()

boxplot=seaborn.boxplot(x='Item4',data=df)

plt.show()

boxplot=seaborn.boxplot(x='Item5',data=df)

plt.show()

boxplot=seaborn.boxplot(x='Item6',data=df)

plt.show()

boxplot=seaborn.boxplot(x='Item7',data=df)
```

```
plt.show()

boxplot=seaborn.boxplot(x='Item8',data=df)

plt.show()


#Visualizing distribution shapes

print('Children: ')

plt.hist(df['Children'])

plt.show()

print('Age: ')

plt.hist(df['Age'])

plt.show()

print('Income: ')

plt.hist(df['Income'])

plt.show()

print('Overweight: ')

plt.hist(df['Overweight'])

plt.show()

print('Anxiety: ')

plt.hist(df['Anxiety'])

plt.show()

print('Initial_days: ')

plt.hist(df['Initial_days'])

plt.show()
```

```
#Check for unique values to re-express categorical variables

print('Education: ', df.Education.unique(), "\n")

print('Marital: ', df.Marital.unique())

print('Gender: ', df.Gender.unique())

print('ReAdmis: ', df.ReAdmis.unique())

print('Soft_drink: ', df.Soft_drink.unique())

print('Initial_admin: ', df.Initial_admin.unique())

print('HighBlood: ', df.HighBlood.unique())

print('Stroke: ', df.Stroke.unique())

print('Complication_risk: ', df.Complication_risk.unique())

print('Arthritis: ', df.Arthritis.unique())

print('Diabetes: ', df.Diabetes.unique())

print('Hyperlipidemia: ', df.Hyperlipidemia.unique())

print('BackPain: ', df.BackPain.unique())

print('Allergic_rhinitis: ', df.Allergic_rhinitis.unique())

print('Reflux_esophagitis: ', df.Reflux_esophagitis.unique())

print('Ashtma: ', df.Asthma.unique())

print('Services: ', df.Services.unique())



#Turn categorical values into quantitative data

df['Education_numeric'] = df['Education']

dict_edu = {"Education_numeric": {"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,"Associate's Degree": 7,"Professional School Degree": 8,"Bachelor's Degree": 9,"Master's Degree": 10,"Doctorate Degree": 11 }}
```

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



df['Marital_numeric'] = df['Marital']

dict_marital = {"Marital_numeric": {"Never Married": 0,"Separated": 1,"Widowed": 2,"Divorced": 3,
"Married": 4}}

df.replace(dict_marital, inplace=True)



df['Gender_numeric'] = df['Gender']

dict_gender = {"Gender_numeric": {"Prefer not to answer": 0,"Male": 1,"Female": 2}}

df.replace(dict_gender, inplace=True)



df['ReAdmis_numeric'] = df['ReAdmis']

dict_ReAdmis = {"ReAdmis_numeric": {"No": 0, "Yes": 1}}

df.replace(dict_ReAdmis, inplace=True)



df['Soft_drink_numeric'] = df['Soft_drink']

dict_Soft_drink = {"Soft_drink_numeric": {"No": 0, "Yes": 1}}

df.replace(dict_Soft_drink, inplace=True)



df['Initial_admin_numeric'] = df['Initial_admin']

dict_Initial_admin = {"Initial_admin_numeric": {"Emergency Admission": 0, "Elective Admission":
1,"Observation Admission": 2}}

df.replace(dict_Initial_admin, inplace=True)



df['HighBlood_numeric'] = df['HighBlood']
```

```
dict_HighBlood = {"HighBlood_numeric": {"No": 0, "Yes": 1}}

df.replace(dict_HighBlood, inplace=True)


df['Stroke_numeric'] = df['Stroke']

dict_stroke = {"Stroke_numeric": {"No": 0, "Yes": 1}}

df.replace(dict_stroke, inplace=True)


df['Complication_risk_numeric'] = df['Complication_risk']

dict_complication = {"Complication_risk_numeric": {"Low": 0, "Medium": 1, "High": 2}}

df.replace(dict_complication, inplace=True)


df['Arthritis_numeric'] = df['Arthritis']

dict_arthritis = {"Arthritis_numeric": {"No": 0, "Yes": 1}}

df.replace(dict_arthritis, inplace=True)


df['Diabetes_numeric'] = df['Diabetes']

dict_diabetes = {"Diabetes_numeric": {"No": 0, "Yes": 1}}

df.replace(dict_diabetes, inplace=True)


df['Hyperlipidemia_numeric'] = df['Hyperlipidemia']

dict_hyperlipidemia = {"Hyperlipidemia_numeric": {"No": 0, "Yes": 1}}

df.replace(dict_hyperlipidemia, inplace=True)


df['BackPain_numeric'] = df['BackPain']
```

```
dict_backpain = {"BackPain_numeric": {"No": 0, "Yes": 1}}

df.replace(dict_backpain, inplace=True)


df['Allergic_rhinitis_numeric'] = df['Allergic_rhinitis']

dict_allergies = {"Allergic_rhinitis_numeric": {"No": 0, "Yes": 1}}

df.replace(dict_allergies, inplace=True)


df['Reflux_esophagitis_numeric'] = df['Reflux_esophagitis']

dict_reflux = {"Reflux_esophagitis_numeric": {"No": 0, "Yes": 1}}

df.replace(dict_reflux, inplace=True)


df['Asthma_numeric'] = df['Asthma']

dict_asthma = {"Asthma_numeric": {"No": 0, "Yes": 1}}

df.replace(dict_asthma, inplace=True)


df['Services_numeric'] = df['Services']

dict_services = {"Services_numeric": {"Blood Work": 0, "Intravenous": 1, "CT Scan": 2, "MRI": 3}}

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

Part III: Data Cleaning (Treatment)

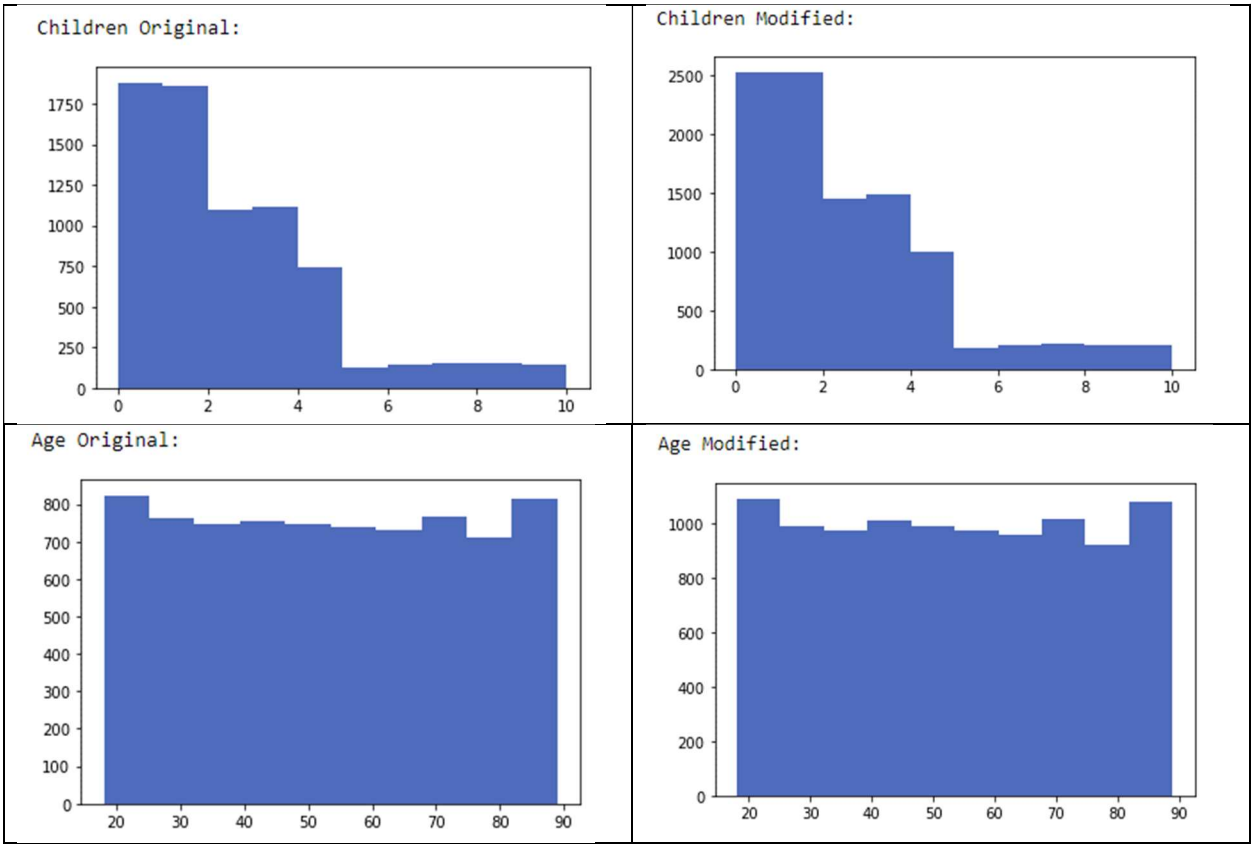
D 1. After checking for duplicates I found no duplicate rows. After checking for missing values I found that the columns for Children (2,588), Age (2,414), Income (2,464), Soft_drink (2,467), Overweight (982), Anxiety (984), and Initial_days (1,056) had missing data above the 5% threshold and would need to be addressed.

I ran boxplots for the following columns Children, Age, Income, VitD_levels, Doc_visits, Full_meals_eaten, Vitd_supp, Overweight, Anxiety, Initial_days, TotalCharge, Additional_charges, and Items 1-8. In Children, Full_meals_eaten, VitD_supp, TotalCharge, and Items 1-8 I found outliers based on the boxplot. However, because of the values being collected for all of these

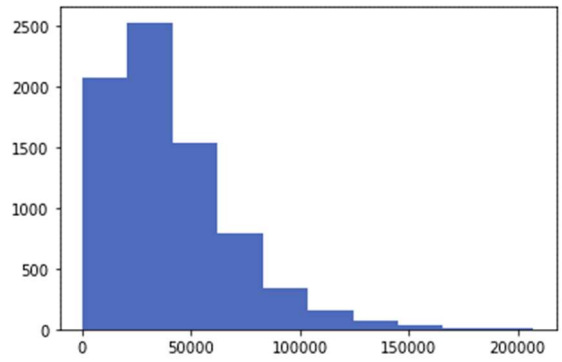
columns I chose not to consider them as outliers that were made in error and had to be dropped. They are all purposeful outliers and provide meaningful data.

D 2. I filled all missing values in using different methods. However, I tried many different ways to treat each column that had missing values and settled on the method the kept the distribution shape mostly the same. I used forward fills for the Children, Age, and Initial_days columns. In the Overweight and Anxiety columns I filled in missing values with a 0. That made sense as that column had either a 1 for yes or 0 for no. I assumed that an NA in this column was also a no. I did that same thing in the Soft_drink column, however this was a Yes/No column so I replaced all Na values with No. Finally, the Income column had all Na values replaced with the median value of that column. I've included a table with the original distributions and the distributions in those columns after I filled the missing data with the mentioned methods.

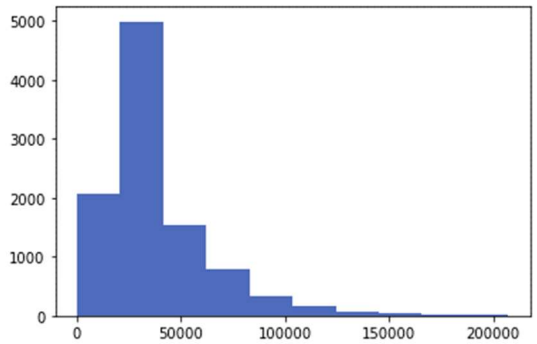
I decided on these methods based on what I learned in the data camp learning modules, and course materials (Larose & Larose, 2019). Those two sources gave me a conceptual and coding foundation to tackle these problems.



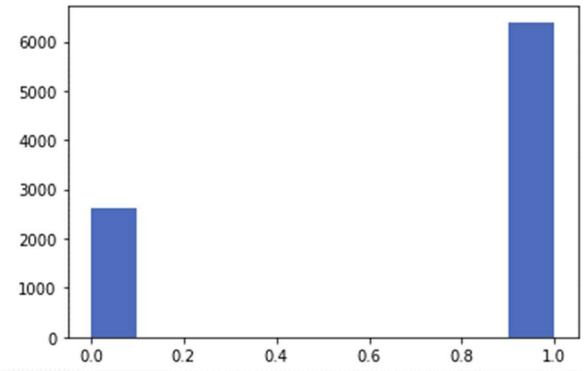
Income Original:



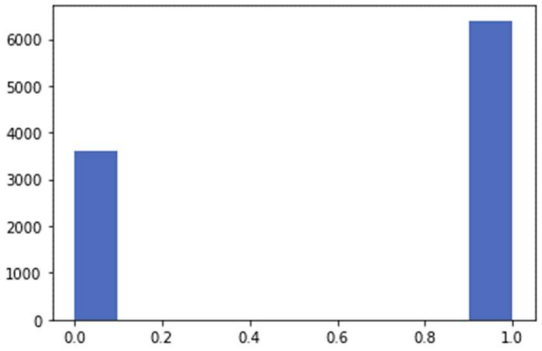
Income Modified:



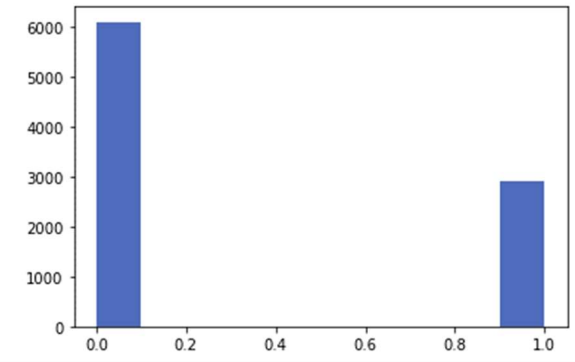
Overweight Original:



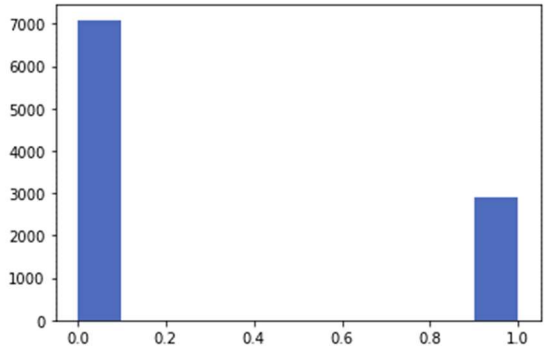
Overweight Modified:



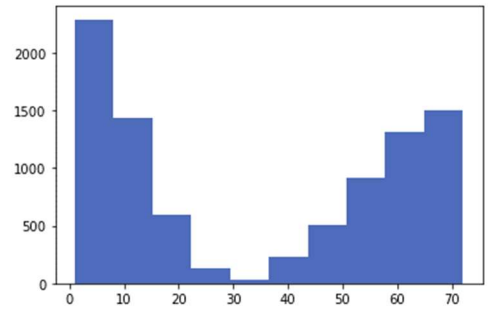
Anxiety Original:



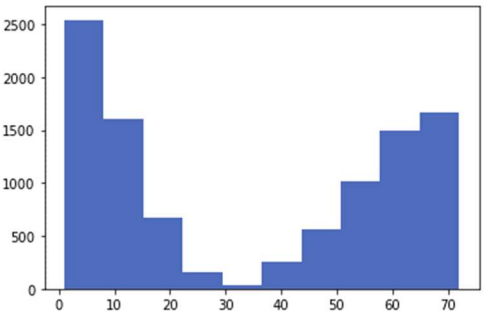
Anxiety Modified:



Initial_days Original:



Initial_days Modified:



As you can see the distributions after filling in missing data with the methods I chose have maintained the original distributions for the most part. This will help provide an accurate analysis in the later stages of the data life cycle.

D 3. After filling in all the missing data and verifying that the distributions maintained their integrity I again ran df.isnull().sum() to check for any null values in the data set. My query returned that the data set had no missing values as shown here:

```
Unnamed: 0            0
CaseOrder             0
Customer_id           0
Interaction           0
UID                   0
City                  0
State                 0
County                0
Zip                   0
Lat                   0
Lng                   0
Population            0
Area                  0
Timezone              0
Job                   0
Children              0
Age                   0
Education             0
Employment            0
Income                0
Marital               0
Gender                0
ReAdmis               0
VitD_levels           0
Doc_visits            0
Full_meals_eaten      0
VitD_supp             0
Soft_drink            0
Initial_admin         0
HighBlood             0
Stroke                0
Complication_risk     0
Overweight            0
Arthritis             0
Diabetes              0
Hyperlipidemia        0
BackPain              0
Anxiety               0
Allergic_rhinitis     0
Reflux_esophagitis    0
Asthma                0
Services              0
Initial_days          0
TotalCharge           0
Additional_charges    0
Item1                 0
Item2                 0
Item3                 0
Item4                 0
Item5                 0
Item6                 0
Item7                 0
Item8                 0
dtype: int64
```

Upon verifying the missing values being treated and the distributions remaining the same I could move onto re-expressing qualitative data into quantitative data. I used df.unique() to return all the unique values in the qualitative columns as shown here:

```
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']

Marital:  ['Divorced' 'Married' 'Widowed' 'Never Married' 'Separated']
Gender:  ['Male' 'Female' 'Prefer not to answer']
ReAdmis:  ['No' 'Yes']
Soft_drink:  ['No' 'Yes']
Initial_admin:  ['Emergency Admission' 'Elective Admission' 'Observation Admission']
HighBlood:  ['Yes' 'No']
Stroke:  ['No' 'Yes']
Complication_risk:  ['Medium' 'High' 'Low']
Arthritis:  ['Yes' 'No']
Diabetes:  ['Yes' 'No']
Hyperlipidemia:  ['No' 'Yes']
BackPain:  ['Yes' 'No']
Allergic_rhinitis:  ['Yes' 'No']
Reflux_esophagitis:  ['No' 'Yes']
Ashtma:  ['Yes' 'No']
Services:  ['Blood Work' 'Intravenous' 'CT Scan' 'MRI']
```

Then I created new columns for all the quantitative columns for their new numeric values. For example, for the Education column I created an Education_numeric column that would be populated with numeric values that equated to the qualitative descriptors in the original data set. After doing so I ran df.info() again to verify that my new columns had been created and with the correct type of data as shown here:

```
53  Education_numeric           10000 non-null  int64
54  Marital_numeric             10000 non-null  int64
55  Gender_numeric              10000 non-null  int64
56  ReAdmis_numeric             10000 non-null  int64
57  Soft_drink_numeric          10000 non-null  int64
58  Initial_admin_numeric       10000 non-null  int64
59  HighBlood_numeric           10000 non-null  int64
60  Stroke_numeric              10000 non-null  int64
61  Complication_risk_numeric   10000 non-null  int64
62  Arthritis_numeric           10000 non-null  int64
63  Diabetes_numeric            10000 non-null  int64
64  Hyperlipidemia_numeric      10000 non-null  int64
65  BackPain_numeric            10000 non-null  int64
66  Allergic_rhinitis_numeric   10000 non-null  int64
67  Reflux_esophagitis_numeric  10000 non-null  int64
68  Asthma_numeric              10000 non-null  int64
69  Services_numeric            10000 non-null  int64
```

I also again ran a df.unique query on each column to verify that I coded it properly as shown here:

```
Education_numeric:  [ 5  6  3  4  9 10  1  2 11  7  8  0]
Marital_numeric:  [3 4 2 0 1]
Gender_numeric:  [1 2 0]
ReAdmis_numeric:  [0 1]
Soft_drink_numeric:  [0 1]
Initial_admin_numeric:  [0 1 2]
HighBlood_numeric:  [1 0]
Stroke_numeric:  [0 1]
Complication_risk_numeric:  [1 2 0]
Arthritis_numeric:  [1 0]
Diabetes_numeric:  [1 0]
Hyperlipidemia_numeric:  [0 1]
BackPain_numeric:  [1 0]
Allergic_rhinitis_numeric:  [1 0]
Reflux_esophagitis_numeric:  [0 1]
Ashtma_numeric:  [1 0]
Services_numeric:  [0 1 2 3]
```

D 4. See code/script attached and code pasted here:

```
#Visualizing distribution shapes

print('Children Original: ')

plt.hist(df['Children'])

plt.show()

print('Age Original: ')

plt.hist(df['Age'])

plt.show()

print('Income Original: ')

plt.hist(df['Income'])

plt.show()

print('Overweight Original: ')

plt.hist(df['Overweight'])

plt.show()
```

```
print('Anxiety Original: ')
plt.hist(df['Anxiety'])
plt.show()
print('Initial_days Original: ')
plt.hist(df['Initial_days'])
plt.show()

#Fixing missing data based on distribution shapes
df['Children'].fillna(method='ffill', inplace=True)
df['Age'].fillna(method='ffill', inplace=True)
df['Income'].fillna(df['Income'].median(), inplace=True)
df['Soft_drink'].fillna("No", inplace=True)
df['Overweight'].fillna(0, inplace=True)
df['Anxiety'].fillna(0, inplace=True)
df['Initial_days'].fillna(method='ffill', inplace=True)

#Checking to make sure all missing data has been replaced with selected values
print(df.isnull().sum())

#Visualizing distribution shapes after changes
print('Children Modified: ')
plt.hist(df['Children'])
plt.show()
print('Age Modified: ')
```

```
plt.hist(df['Age'])

plt.show()

print('Income Modified: ')

plt.hist(df['Income'])

plt.show()

print('Overweight Modified: ')

plt.hist(df['Overweight'])

plt.show()

print('Anxiety Modified: ')

plt.hist(df['Anxiety'])

plt.show()

print('Initial_days Modified: ')

plt.hist(df['Initial_days'])

plt.show()


#Check for unique values to re-express categorical variables

print('Education: ', df.Education.unique(), "\n")

print('Marital: ', df.Marital.unique())

print('Gender: ', df.Gender.unique())

print('ReAdmis: ', df.ReAdmis.unique())

print('Soft_drink: ', df.Soft_drink.unique())

print('Initial_admin: ', df.Initial_admin.unique())

print('HighBlood: ', df.HighBlood.unique())

print('Stroke: ', df.Stroke.unique())
```

```
print('Complication_risk: ', df.Complication_risk.unique())

print('Arthritis: ', df.Arthritis.unique())

print('Diabetes: ', df.Diabetes.unique())

print('Hyperlipidemia: ', df.Hyperlipidemia.unique())

print('BackPain: ', df.BackPain.unique())

print('Allergic_rhinitis: ', df.Allergic_rhinitis.unique())

print('Reflux_esophagitis: ', df.Reflux_esophagitis.unique())

print('Ashtma: ', df.Asthma.unique())

print('Services: ', df.Services.unique())



#Turn categorical values into quantitative data

df['Education_numeric'] = df['Education']

dict_edu = {"Education_numeric": {"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,"Associate's Degree": 7,"Professional School Degree": 8,"Bachelor's Degree": 9,"Master's Degree": 10,"Doctorate Degree": 11 }}

df.replace(dict_edu, inplace=True)



df['Marital_numeric'] = df['Marital']

dict_marital = {"Marital_numeric": {"Never Married": 0,"Separated": 1,"Widowed": 2,"Divorced": 3, "Married": 4}}

df.replace(dict_marital, inplace=True)



df['Gender_numeric'] = df['Gender']

dict_gender = {"Gender_numeric": {"Prefer not to answer": 0,"Male": 1,"Female": 2}}

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

```
df['ReAdmis_numeric'] = df['ReAdmis']

dict_ReAdmis = {"ReAdmis_numeric": {"No": 0, "Yes": 1}}

df.replace(dict_ReAdmis, inplace=True)



df['Soft_drink_numeric'] = df['Soft_drink']

dict_Soft_drink = {"Soft_drink_numeric": {"No": 0, "Yes": 1}}

df.replace(dict_Soft_drink, inplace=True)



df['Initial_admin_numeric'] = df['Initial_admin']

dict_Initial_admin = {"Initial_admin_numeric": {"Emergency Admission": 0, "Elective Admission":
1,"Observation Admission": 2}}

df.replace(dict_Initial_admin, inplace=True)



df['HighBlood_numeric'] = df['HighBlood']

dict_HighBlood = {"HighBlood_numeric": {"No": 0, "Yes": 1}}

df.replace(dict_HighBlood, inplace=True)



df['Stroke_numeric'] = df['Stroke']

dict_stroke = {"Stroke_numeric": {"No": 0, "Yes": 1}}

df.replace(dict_stroke, inplace=True)



df['Complication_risk_numeric'] = df['Complication_risk']

dict_complication = {"Complication_risk_numeric": {"Low": 0, "Medium": 1, "High": 2}}
```

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


df['Arthritis_numeric'] = df['Arthritis']
dict_arthritis = {"Arthritis_numeric": {"No": 0, "Yes": 1}}
df.replace(dict_arthritis, inplace=True)


df['Diabetes_numeric'] = df['Diabetes']
dict_diabetes = {"Diabetes_numeric": {"No": 0, "Yes": 1}}
df.replace(dict_diabetes, inplace=True)


df['Hyperlipidemia_numeric'] = df['Hyperlipidemia']
dict_hyperlipidemia = {"Hyperlipidemia_numeric": {"No": 0, "Yes": 1}}
df.replace(dict_hyperlipidemia, inplace=True)


df['BackPain_numeric'] = df['BackPain']
dict_backpain = {"BackPain_numeric": {"No": 0, "Yes": 1}}
df.replace(dict_backpain, inplace=True)


df['Allergic_rhinitis_numeric'] = df['Allergic_rhinitis']
dict_allergies = {"Allergic_rhinitis_numeric": {"No": 0, "Yes": 1}}
df.replace(dict_allergies, inplace=True)


df['Reflux_esophagitis_numeric'] = df['Reflux_esophagitis']
dict_reflux = {"Reflux_esophagitis_numeric": {"No": 0, "Yes": 1}}
```

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


df['Asthma_numeric'] = df['Asthma']
dict_asthma = {"Asthma_numeric": {"No": 0, "Yes": 1}}
df.replace(dict_asthma, inplace=True)


df['Services_numeric'] = df['Services']
dict_services = {"Services_numeric": {"Blood Work": 0, "Intravenous": 1, "CT Scan": 2, "MRI": 3}}
df.replace(dict_services, inplace=True)


#Check for new columns
df.info()


#Checking data in new columns to make sure replacements worked
print('Education_numeric: ', df.Education_numeric.unique())
print('Marital_numeric: ', df.Marital_numeric.unique())
print('Gender_numeric: ', df.Gender_numeric.unique())
print('ReAdmis_numeric: ', df.ReAdmis_numeric.unique())
print('Soft_drink_numeric: ', df.Soft_drink_numeric.unique())
print('Initial_admin_numeric: ', df.Initial_admin_numeric.unique())
print('HighBlood_numeric: ', df.HighBlood_numeric.unique())
print('Stroke_numeric: ', df.Stroke_numeric.unique())
print('Complication_risk_numeric: ', df.Complication_risk_numeric.unique())
print('Arthritis_numeric: ', df.Arthritis_numeric.unique())
```

```
print('Diabetes_numeric: ', df.Diabetes_numeric.unique())

print('Hyperlipidemia_numeric: ', df.Hyperlipidemia_numeric.unique())

print('BackPain_numeric: ', df.BackPain_numeric.unique())

print('Allergic_rhinitis_numeric: ', df.Allergic_rhinitis_numeric.unique())

print('Reflux_esophagitis_numeric: ', df.Reflux_esophagitis_numeric.unique())

print('Ashtma_numeric: ', df.Asthma_numeric.unique())

print('Services_numeric: ', df.Services_numeric.unique(), '\n')
```

D 5. CSV file uploaded with submission

D 6. I cleaned the data in a couple of ways. The first being that I filled in missing values. By filling in those missing values I maintained the integrity of the original distributions. However, that's not necessarily accurate. If those missing values had been filled in with accurate information the distributions probably would have been different to some degree. Because I chose to maintain the distributions any analysis done on this data set is based on data that is largely incomplete and might not provide the level of insight that a fully complete data set could.

I also converted qualitative data to quantitative data. While most columns were Yes/No columns and easily converted from 0 to 1, the Education column had 12 variables numbered 0-11. I tried my best to rank them however I might not have ranked them appropriately as I wasn't sure what some of the data was referring to. I am not sure what professional school is so I wasn't sure if that was higher or lower than some other options provided. Because of this any analysis run on level of education will not be entirely accurate.

D 7. I am not entirely sure what challenges a data analyst would run into when using my cleaned data set that I didn't mention in the previous response. There are a couple issues with the data set relating to the missing data in the original copy and how I went about filling in that data. Because the data may not be entirely accurate any analysis done on it can lead to faulty conclusions.

Part IV: PCA

E1. In my PCA I chose 20 variables. As stated in the text, "if there are fewer than 20 predictors, the eigenvalue criterion tends to recommend extracting too few components." (Larose & Larose 2019) Based on this I want to run a PCA with at least 20 variables to produce an output that would allow us to extract enough components.

Included below is a screenshot of the PCA loadings matrix which lists the individual variables chosen.

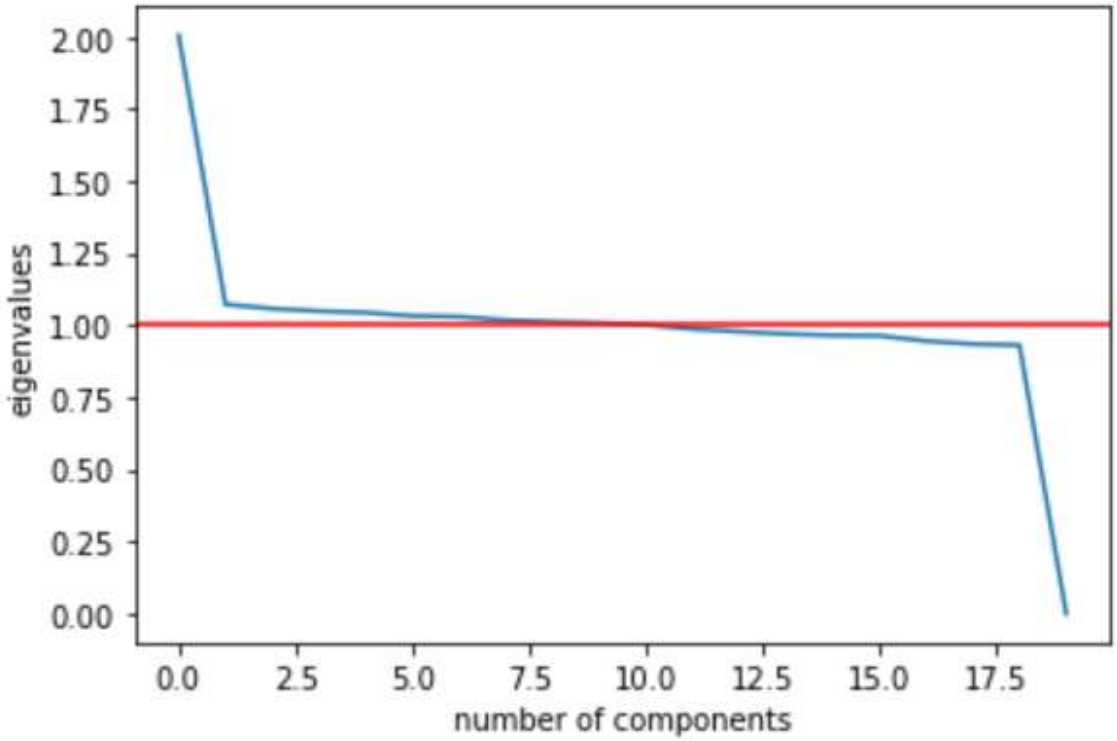
```
PCA(n_components=20)
                                PC1       PC2       PC3       PC4       PC5  \
ReAdmis_numeric              0.019851  0.127795  0.175456 -0.353865 -0.032540
Services_numeric             0.705512  0.009579 -0.013558 -0.004255 -0.010995
Age                          0.013203 -0.229001 -0.143450  0.074654  0.354792
Gender_numeric               0.036482 -0.036906  0.062988  0.395736 -0.274561
Doc_visits                  -0.001959 -0.261453  0.133387  0.008292 -0.242440
Soft_drink_numeric           0.011052  0.114656  0.543563  0.084041  0.164635
Initial_admin_numeric       -0.021101 -0.112728 -0.255054  0.400224 -0.167758
HighBlood_numeric            0.007924 -0.394309 -0.055899 -0.298067 -0.065738
Stroke_numeric              -0.003465 -0.196086  0.016540  0.251773 -0.072744
Complication_risk_numeric    0.004593 -0.140210 -0.025062 -0.192025 -0.258244
Arthritis_numeric           -0.004227  0.047176 -0.436380 -0.255660  0.114494
Diabetes_numeric             0.034947  0.094009 -0.086680  0.241931  0.356749
Hyperlipidemia_numeric      -0.004628  0.427117  0.150956 -0.177230  0.298589
BackPain_numeric             0.008511 -0.236902  0.538287  0.062751  0.062242
Allergic_rhinitis_numeric    0.006058 -0.242265 -0.058299 -0.164063  0.247245
Reflux_esophagitis_numeric   0.004447  0.157243  0.136674  0.022118 -0.331555
Asthma_numeric               0.004309 -0.159516  0.043155  0.303615  0.411165
Services_numeric             0.705512  0.009579 -0.013558 -0.004255 -0.010995
Education_numeric           -0.019142  0.238284  0.026542 -0.072749 -0.164102
Marital_numeric              0.012822 -0.454559  0.163839 -0.268373  0.081588

                                PC6       PC7       PC8       PC9      PC10  \
ReAdmis_numeric             -0.143015 -0.119035  0.181851  0.522127  0.108240
Services_numeric             0.004155  0.017485  0.019507 -0.009729 -0.010693
Age                         -0.062761 -0.149631 -0.112612  0.494622  0.017862
Gender_numeric              -0.219364 -0.189680 -0.461167  0.033749 -0.077140
Doc_visits                   0.246096 -0.582819  0.107615 -0.035123 -0.031306
Soft_drink_numeric           0.196500 -0.273601 -0.004180 -0.054392  0.139432
Initial_admin_numeric        0.020542  0.095840  0.392278  0.006889  0.073086
HighBlood_numeric            0.095530  0.093574  0.124019 -0.150690  0.437013
Stroke_numeric               0.110094  0.149226  0.225268  0.426483  0.119244
Complication_risk_numeric    0.494101  0.105350 -0.316720  0.033632  0.347314
Arthritis_numeric           -0.369964 -0.313207  0.087425  0.013103  0.228266
Diabetes_numeric             0.222501 -0.425421  0.146032 -0.072290  0.276184
Hyperlipidemia_numeric       0.159638  0.191373 -0.063646  0.029092  0.110516
BackPain_numeric            -0.338007  0.069437  0.161794  0.074169  0.033480
Allergic_rhinitis_numeric   -0.144564 -0.223779 -0.321020 -0.215393 -0.121809
Reflux_esophagitis_numeric  -0.441239 -0.036442 -0.120240 -0.084374  0.545925
Asthma_numeric              -0.102526  0.238795  0.013412 -0.289638  0.330642
Services_numeric             0.004155  0.017485  0.019507 -0.009729 -0.010693
Education_numeric           -0.035461 -0.156908  0.460780 -0.308875 -0.114648
Marital_numeric             -0.094869  0.086213  0.140313 -0.163975 -0.230519
```

	PC16	PC17	PC18	PC19	\
ReAdmis_numeric	-0.050733	-0.058256	-0.108383	0.244485	
Services_numeric	-0.002500	0.008656	-0.000445	0.003786	
Age	0.062460	0.164435	-0.238058	-0.071583	
Gender_numeric	0.275468	-0.028774	0.269179	0.252051	
Doc_visits	-0.116821	0.541442	-0.035362	0.201534	
Soft_drink_numeric	0.376105	-0.450375	-0.065626	0.189284	
Initial_admin_numeric	-0.047727	-0.267675	-0.149689	0.299899	
HighBlood_numeric	0.604621	0.125542	-0.007621	-0.176800	
Stroke_numeric	-0.028099	0.008687	0.177146	0.171509	
Complication_risk_numeric	-0.401106	-0.207756	0.276240	0.074581	
Arthritis_numeric	0.045635	-0.178694	0.480117	0.200406	
Diabetes_numeric	-0.238413	-0.191279	-0.025169	-0.327723	
Hyperlipidemia_numeric	-0.008969	0.305429	0.105164	0.367868	
BackPain_numeric	-0.225011	0.043870	0.458607	-0.344312	
Allergic_rhinitis_numeric	-0.177501	-0.073691	-0.092135	0.137754	
Reflux_esophagitis_numeric	-0.224839	0.037990	-0.449507	-0.028934	
Asthma_numeric	-0.085380	0.278721	0.054463	0.338494	
Services_numeric	-0.002500	0.008656	-0.000445	0.003786	
Education_numeric	-0.021570	0.108999	0.140777	0.073220	
Marital_numeric	-0.183192	-0.285961	-0.193797	0.312618	

	PC20
ReAdmis_numeric	2.696760e-16
Services_numeric	-7.071068e-01
Age	1.734723e-17
Gender_numeric	-1.665335e-16
Doc_visits	4.380177e-17
Soft_drink_numeric	1.526557e-16
Initial_admin_numeric	3.053113e-16
HighBlood_numeric	-2.151057e-16
Stroke_numeric	-7.112366e-17
Complication_risk_numeric	-1.804112e-16
Arthritis_numeric	1.387779e-16
Diabetes_numeric	5.551115e-17
Hyperlipidemia_numeric	2.081668e-17
BackPain_numeric	5.551115e-17
Allergic_rhinitis_numeric	4.857226e-17
Reflux_esophagitis_numeric	-1.249001e-16
Asthma_numeric	1.665335e-16
Services_numeric	7.071068e-01
Education_numeric	-7.979728e-17
Marital_numeric	-2.636780e-16

E2. Using the Scree Plot below I'd probably have to drop all PCs after PC5. The text states that we want to drop PCs with an eigenvalue below 1. PC5 seems to be the last PC that has a value over 1.



E3. PCA allows us to see the correlation coefficients that variables have within different principal components. When look at data this way we are able to quickly see the relationships between variables when compared between different principal components. The closer the numbers in each principal component are to 1 or -1 the more strongly they are related in a positive or negative manner. Running this analysis allows us to group together the variables that strongly correlate to each other. In this case we'd look at the variables that correlate to patient readmissions and seeing what variables the hospital system can control to hopefully prevent future readmissions.

F. Panopto Recording Link provided in submission.

G. No in text citations made for third-party code references.

H.

Larose, C. D., & Larose, D. T. (2019). Data science using Python and R.