

Performance Assessment for MSDA D206

MSDA D206: Data Cleaning

Student: Eric Yarger

Student ID: 010396172

Part 1: Research Question

A. Research Question/Decision

Medical readmission plays an important, and costly, role in modern hospital administration. According to Kaiser Health News, Medicare is reducing payments to 47% of the nation's hospitals, by an average of .64% payment per patient (Rau, 2021). As a data analyst for our hospital chain, I have decided that a question with great potential to positively affect readmission rates is: **What variables that we've collected data for can be grouped together to provide a more cohesive and streamlined analysis on readmission rates at our facilities?**

To find the answer, I will clean the full dataset, address missing variables and outliers, and perform Principal Component Analysis (PCA). This process will find what variables can be grouped together while still maintaining the same predictive power. The goal is to help inform the company as to what statistics and survey responses are unique and hold the most potential for impacting readmission rates positively.

B. Required Variables

There are a wide variety of variables that have been gathered for each individual in the dataset. The variables broadly identify the user ID in the medical system, their geographic location, socioeconomic background, lifestyle, medical history, and responses to a survey given. The type of missing data can be broken down into three categories: Missing at Random (MAR), Missing Not at Random (MNAR), and Missing Completely at Random (MCAR). Strategic Data Scientist Zakarie Hashi's article breakdown of these missing definitions was used to inform the analyst of the best approach to labeling the missing data type (Hashi, 2018). For this analysis the missing data in each variable will be assumed to be MCAR.

Table B1 - Initial Data Variables.

Variable	Type	Qualitative / Quantitative	Example	Missing Values?
Unnamed	int64	Quantitative	1	No
CaseOrder	int64	Quantitative	2	No

Customer_id	object	Qualitative	Z919181	No
Interaction	object	Qualitative	a2057123-abf5-4a2c-abad-8ffe335122	No
UID	object	Qualitative	e19a0fa00aeda885b8a436757e889bc9	No
City	object	Qualitative	Sioux Falls	No
State	object	Qualitative	SD	No
County	object	Qualitative	Minnehaha	No
Zip	int64	Quantitative	57110	No
Lat	float64	Quantitative	43.54321	No
Lng	float64	Quantitative	96.63772	No
Population	int64	Quantitative	35054	No
Area	object	Qualitative	Suburban	No
Timezone	object	Qualitative	America/Chicago	No
Job	object	Qualitative	Chief Executive Officer	No
Children	float64	Quantitative	3	Yes
Age	float64	Quantitative	53	Yes
Education	object	Qualitative	Some College, Less than 1 Year	No
Employment	object	Qualitative	Retired	No
Income	float64	Quantitative	14370.14	Yes
Marital	object	Qualitative	Widowed	No
Gender	object	Qualitative	Female	No
ReAdmis	object	Qualitative	No	No
VitD_Levels	float64	Quantitative	17.4158887	No
Doc_visits	int64	Quantitative	4	No
Full_meals_eaten	int64	Quantitative	1	No
VitD_supp	int64	Quantitative	0	No
Soft_drink	object	Qualitative	No	Yes
Initial_admin	object	Qualitative	Elective Admission	No
HighBlood	object	Qualitative	Yes	No

Stroke	object	Qualitative	No	No
Complication_risk	object	Qualitative	Medium	No
Overweight	float64	Quantitative	1	Yes
Arthritis	object	Qualitative	No	No
Diabetes	object	Qualitative	Yes	No
Hyperlipidemia	object	Qualitative	No	No
BackPain	object	Qualitative	No	No
Anxiety	float64	Quantitative	0	Yes
Allergic_rhinitis	object	Qualitative	No	No
Reflux_esophagitis	object	Qualitative	No	No
Asthma	object	Qualitative	Yes	No
Services	object	Qualitative	Blood Work	No
Initial_days	float64	Quantitative	10.59	Yes
TotalCharge	float64	Quantitative	3191.586768	No
Additional_charges	float64	Quantitative	17505.19246	No
Item1	int64	Quantitative	2	No
Item2	int64	Quantitative	4	No
Item3	int64	Quantitative	4	No
Item4	int64	Quantitative	4	No
Item5	int64	Quantitative	3	No
Item6	int64	Quantitative	4	No
Item7	int64	Quantitative	3	No
Item8	int64	Quantitative	3	No

Table 1A informs the reader of all of the variables in the dataset, in their unaltered format. Not all of these variables will be used in our analysis, but all of them will be cleaned. Table 1B below lists the variables

used for this project's PCA. Note that variable type and example have possibly changed format from Table 1A. The methods and actions associated with this will be addressed in detail later in the report.

Table B2 - Variables required for Analysis

Variable	Qualitative or Quantitative	Type	Example
Doc_visits	Quantitative	int64	6
VitD_supp	Quantitative	int64	2
Initial_days	Quantitative	int64	15
Total_charge	Quantitative	int64	3191
Additional_charges	Quantitative	int64	17939
Timely_admission	Quantitative	int64	3
Timely_treatment	Quantitative	int64	4
Timely_visit	Quantitative	int64	2
Reliability	Quantitative	int64	2
Options	Quantitative	int64	3
Hours	Quantitative	int64	4
Courteous	Quantitative	int64	5
Active_listen	Quantitative	int64	3

Part 2: Data Cleaning Plan

C1. Plan to Find Anomalies

The plan is to clean, categorize, and/or quantify every variable in the dataset. The purpose of cleaning the entire dataset is to provide a cohesive and useful source for other analysts on the healthcare team to use for their analysis. To break the plan down into its steps:

Step 1. Address redundant variables in the dataset and add 'Index' variable.

The first (unnamed) and second (CaseOrder) variables in the dataset are identical. Column #1 (unnamed) will be removed from the dataset. A new variable 'Index', (datatype int64, example '2'), will be added to the dataset. This variable uniquely indexes each case entry starting with the number '0'. This step will be done with Python in a Jupyter Notebook.

Step 2. Rename necessary variables in accordance with the provided dataset documentation.

The last eight variables in the dataset, named Item1, Item2, Item3, Item4, Item5, Item6, Item7, and Item8 will be renamed to the appropriate variable name, provided in the PDF 'D206 Data Cleaning Medical Data Considerations and Dictionary' that was included in the file download for this PA. This will be done with Python using the .rename() function in a Jupyter Notebook.

Step 3. Change all applicable qualitative data to quantitative data.

There are two general categories that gathered data fall into, quantitative and qualitative. To quote from an article written by a group of researchers from Deloitte's Center for Integrated Research, "Quantitative responses can be tallied and analyzed quickly, efficiently, and accurately with the help of mathematical logic formulas, algorithms, and, now, machines. Analyzing qualitative data is typically trickier, and it largely remains the province of human analysts, given that it requires a high degree of contextual understanding and social intelligence." (Mahto et. al., 2020).

The plan is to take some of the 'trickiness' mentioned out of the data through the cleaning process. Quantitative data will be considered more useful for this dataset, for both my analysis as well as for future dataset use by the project sponsor. As part of the data cleaning process, categorical qualitative variables with 10 or less categories will have their values replaced with quantitative variables.

The variables 'Overweight', 'Soft_drink', and 'Anxiety' all have 'NA' values. Each of these variables missing values can be addressed by substituting a negative response for 'NA' values. To justify the action

taken, patients know if they had a soft drink, they know if they've been medically diagnosed with Anxiety, and they know if they've been medically diagnosed as overweight. Therefore, a 'NA' response equates with a negative response, which in our case is represented as a 0 for these variables.

Step 4. Detect and address variables with null values,.

Detection will be done by using Python's `isnull().any()` function. This creates a 2 column table showing which variables have null values. The variables with null values in the dataset are: Children, Age, Income, Soft_drink, Overweight, Anxiety, and Initial_days. Each of these variables will need to be addressed individually to determine the best method of imputation for null values. This will be done by plotting a histogram of every variable with missing (NA) values. The shape of the Variable will be deduced - for example, is the data positively skewed, uniform, bimodal, etc. After assessing the histogram, the proper form of imputation will be decided and performed to address missing values. This will be done using Python's `.fillna()` function, with the method being determined by deciding what fits the data the best, i.e. the median, mode, mean, backfill, or forward fill. An 'after' histogram will be plotted to ensure that the performed data imputation did not alter the foundational shape of the variable data. Development environment, language, and libraries used will be broken down further in section C3.

Step 5. Detecting Outliers

Outliers in the dataset were visualized and detected using boxplots and Z-Scores. Boxplots provide a visually appealing, effective, and fast method for identifying outliers on both the low and high end of possible data entries per variable. Determining the Z-Score for variables allows for precisely identifying how many variables lie X number of times outside the variable's standard deviation. This analysis calculated and pulled entries that were +3 and -3 deviations from the variable mean.

Using both the histogram and Z-scores all outliers were noted. My decision after finding the outliers was to leave them in the dataset for each variable. This was decided because even though the entries are outliers, they appear to be valid entries. In a field such as medicine, where outliers can correlate with a leading indication of illness or readmission, I feel that it is important to keep these variables in place.

Step 6. Perform PCA

From the dataset that is fully cleaned the variables will be pulled for PCA and saved in a separate file. These variables are: Doc_visits, VitD_supp, Initial_days, Total_charge, Additional_charges, Timely_admission, Timely_treatment, Timely_visits, Reliability, Options, Hours, Courteous, and Active_listen. This analysis ultimately yields four viable principal components, which will be discussed in detail in section E1.

C2. Justification of Approach

In order to justify the approach for cleaning each variable, included is a table listing the variables found necessary to clean, variable characteristics, method chosen to clean the data, and justification for this approach.

Table C2 - Variables to clean, what action was taken, why it was taken, and the function used to complete this task.

Variable	Action Taken	Justification	Python Function Used
Column_1 Unnamed variable	Column was dropped from the dataset.	It was redundant. Column #2, 'CaseOrder' has the same value and purpose.	.drop()
Index	Variable added to the dataset	0 Indexes case in dataset	.insert()
Children	Find data shape and null values, ensuring that data shape and integrity remain intact. Address missing values.	The variable is not useful for further analysis with missing data values.	.fillna() using median values to maintain positive skew shape of data.
Age	Find data shape and null values, ensuring that data shape and integrity remain intact. Address missing values.	The variable is not useful for further analysis with missing data values.	.fillna() using 'backfill' and 'ffill' to maintain uniform distribution of data.

Area	Rename Suburban = 1 Urban = 2 Rural = 3	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Education	Education level names will be replaced with numbers. 'No Schooling Completed' = 0 'Nursery School to 8th Grade' = 1 '9th Grade to 12th Grade, No Diploma' = 2 'GED or Alternative Credential' = 3 'Regular High School Diploma' = 4 'Some College, Less than 1 Year' = 5 'Some College, 1 or More Years No Degree' = 6 'Professional School Degree' = 7 'Associate's Degree' = 8 'Bachelor's Degree' = 9 'Master's Degree' = 10 'Doctorate's Degree' = 11	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Income	Find data shape and null values, ensuring that data shape and integrity remain intact. Address missing values.	The variable is not useful for further analysis with missing data values.	.fillna() using median to maintain positive skew shape of data.
Initial_days	Find data shape and null values, ensuring that data shape and integrity remain intact. Address missing values.	The variable is not useful for further analysis with missing data values.	.fillna() using backfill method to maintain bimodal shape of data.
CaseOrder	Changed variable name to 'Case_order'.	This was done to have consistent Variable naming syntax across the dataset.	.rename()
Area	Changed Values of: 'Suburban' to 1. 'Urban' to 2. 'Rural' to 3.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Employment	Changed Values of: 'Full Time' to 1.	This quantifies the variable cases, making it easier to use	.replace()

	'Retired' to 2. 'Unemployed' to 3. 'Student' to 4. 'Retired' to 5.	the data for analysis.	
Marital	Changed Values of: 'Married' to 1. 'Divorced' to 2. 'Widowed' to 3. 'Never Married' to 4. 'Separated' to 5.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Gender	Changed Values of: 'Female' to 1. 'Male' to 2. 'Prefer not to answer' to 3.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
ReAdmis	Changed Values of: 'No' to a 0. 'Yes' to a 1.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
High_blood	Changed Values of: 'No' to a 0. 'Yes' to a 1.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Soft_drink	Changed values of: 'NA' to 0.	My reasoning for choosing to value each 'NA' entry as 0 is that patients know if they had a soda or not. An answer of 'NA' is a logical case entry equitable to 0 in this instance.	.fillna() using 0 as replacement
Initial_admin	Chaged Values of: 'Emergency Admission' to 3. 'Elective Admission' to 2. 'Observation Admission' to 1	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Stroke	Changed Values of: 'No' to 0. 'Yes' to 1.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Complication_risk	Changed Values of: 'High' to 3. 'Medium' to 2. 'Low' to 1.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Overweight	Changed Values of:	My reasoning for choosing to	.fillna() using

	'NA' to a 0.	value each 'NA' entry as a 0 is that the individual evaluating the patient knows if they are overweight or not. An answer of 'NA' is a logical case entry equitable to 0 in this instance.	0 as replacement
Arthritis	Changed Values of: 'No' to 0. 'Yes' to 1.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Diabetes	Changed Values of: 'No' to 0. 'Yes' to 1.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Hyperlipidemia	Changed Values of: 'No' to 0. 'Yes' to 1.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
BackPain	Changed variable name to 'Back_pain'.	This was done to have consistent Variable naming syntax across the dataset.	.rename()
	Changed Values of: 'No' to 0. 'Yes' to 1.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Anxiety	Changed Values of: 'NA' to 0.	My reasoning for choosing to value each 'NA' entry as a 0 is that the patient knows if they have a medical diagnosis of Anxiety or not. An answer of 'NA' is a logical case entry equitable to 0 in this instance.	.fillna() using 0 as replacement
Allergic_rhinitis	Changed Values of: 'No' to 0. 'Yes' to 1.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Reflux_esophagitis	Changed Values of: 'No' to 0. 'Yes' to 1.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Asthma	Changed Values of: 'No' to 0. 'Yes' to 1.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()

Services	Changed Values of: 'Blood Work' to 1. 'Intravenous' to 2. 'CT Scan' to 3. 'MRI' to 4.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Item1	Changed variable name to 'Timely_admissions'.	The variable name was in the corresponding PDF that was downloaded with the dataset. Changing the name makes the data responses less ambiguous, and more useful for analysis.	.rename()
Item2	Changed variable name to 'Timely_treatment'.	The variable name was in the corresponding PDF that was downloaded with the dataset. Changing the name makes the data responses less ambiguous, and more useful for analysis.	.rename()
Item3	Changed variable name to 'Timely_visits'.	The variable name was in the corresponding PDF that was downloaded with the dataset. Changing the name makes the data responses less ambiguous, and more useful for analysis.	.rename()
Item4	Changed variable name to 'Reliability'.	The variable name was in the corresponding PDF that was downloaded with the dataset. Changing the name makes the data responses less ambiguous, and more useful for analysis.	.rename()
Item5	Changed variable name to 'Options'.	The variable name was in the corresponding PDF that was downloaded with the dataset. Changing the name makes the data responses less ambiguous, and more useful for analysis.	.rename()
Item6	Changed variable name to 'Hours_of_treatment'.	The variable name was in the corresponding PDF that was downloaded with the dataset. Changing the name makes the data responses less ambiguous, and more useful for analysis.	.rename()

Item7	Changed variable name to 'Courteous_staff'.	The variable name was in the corresponding PDF that was downloaded with the dataset. Changing the name makes the data responses less ambiguous, and more useful for analysis.	.rename()
Item8	Changed variable name to 'Doctor_listened'.	The variable name was in the corresponding PDF that was downloaded with the dataset. Changing the name makes the data responses less ambiguous, and more useful for analysis.	.rename()
Income, VitD_levels, Initial_days, Total_charge, Additional_charges	Round entries to nearest integer.	Each variable was analyzed and I decided the dataset would be easier to analyze with these variables rounded up to be integers.	.round() .astype('int64')
Income, VitD_levels, Initial_days, Total_charge, Additional_charges, Children, Age, Education, Readmis, Soft_drink, High_blood, Stroke, Overweight, Arthritis, Diabetes, Anxiety	Convert datatype to int64	After cleaning up these variables by numerical conversion and rounding, they are now integers. To clean the dataset further, and make it more accessible for analysis the decision was made to convert to integers.	.astype('int64')

C3. Justification of Tools

The table below lists the system, environments, tools, and libraries used to clean and analyze the dataset. Justification and example of usage is provided for each tool.

Tool	Description and Benefits	Comparative Example
Jupyter Notebook	Provides an all-in-one, cell-based	Ideal in this situation when compared to a

	environment for executing Python. Useful for visualizing code and executing it block-by-block. Also great for debugging.	different Python environment, such as Spyder or Idle, due to the ability to edit code in cells rather than line-by-line. I found it easier to go back and fix errors due to how visible and re-executable code is in Jupyter Notebook. This made it ideal for this project - because I have had to go back over the code many times.
Python	Python proves to be an accessible language with a large collection of libraries to perform every required analytical task with speed and accuracy	I chose Python due to my familiarity with the language, as well as my belief that in industry there will be a wider group of potential users that can modify the dataset using the code I've written. For this project, that is a benefit because I am assumed to be an analyst on a team of analysts. R is most likely the other language of choice for this task. I chose Python over R for the reasons stated above.
Matplotlib.pyplot	This library is very useful for creating visualizations from data in Python. It was used extensively for plotting histograms for analyzing data shape across variables.	Mathplotlib.pyplot was used to the ease of use and how well it meshes with Pandas for analyzing this dataset. Other options in this scenario would have been D-tail or bamboolib, both of which provide charting capabilities but don't work as well with the other tools I used for this analysis.
Pandas	Pandas, or Python Data Analysis Library is a fast, powerful, and very well documented library that is highly beneficial for data analysis.	Most of the training in this course - DataCamp, the Webinars for the course, and the textbook used pandas. I would be foolish not to use it. Other available tools to use would be PySpark, SciPy, Apache Spark. All of these tools would likely serve a similar purpose, but for my analysis Pandas is ideal due to it's ease of use, widely accessible documentation of functions, and training familiarity.
NumPY	Python library widely beneficial for working with arrays. Stacks as an essential component with Matplotlib and Seaborn as a data analysis and visualization tool.	Used when calculating cov_matrix and eigenvalues during PCA analysis. Justified as being the go-to tool for this from training provided from course webinar and textbook. Alternative tools that could have been used are MATLAB, R, and SciPY. NumPy is beneficial over these alternatives due to being free for commercial and non-commercial use, and is ideal in this analysis because of how it works well with Pandas, Seaborn, and Mathplotlib for creating an optimum analysis environment.
Seaborn	Seaborn is a data visualization library based on matplotlib. It's easy to use, and provides a useful interface for drawing informative graphs.	Seaborn was chosen over alternative tools such as Apache Superset, redash, or plotly. This is due to its ease of use, my prior familiarity with the tool, and simple syntax for creating boxplots for analyzing outliers.

Scipy.stats	Python library used for statistical analysis. Used for calculating z-scores for detection of outliers.	Scipy.stats was chosen for statistical analysis for my training with the library's .zscore() function. Other available tools for this include jax.numpy - which was not ideal due to a more complicated execution of the library's z-score calculation function.
Sklearn PCA	Used for feature extraction, normalization and transforming input data into more usable formats.	Used for PCA. This is the tool taught in the course materials, and is what I'm familiar with. Alternatives would be tidyverse in R - which isn't ideal in a dataset analyzed with Python for the entirety of the project!
Google Sheets	Used importing and exporting .csv files into the coding environment. Also used for quick addition/removal of variables/columns	Alternatives are any cell-based data manipulation software - Excel is a good example of a comparable program. Google Sheets was used because that's where I've stored all of the files, making it ideal for housing everything in one spot.

C4. Provide the Detection Code

Line-by-line code with comments is provided below. Included, and highly recommended to look at, is a PDF attachment to this PA that has the entire execution of my code, with visualizations, graphs, and charts at each step. Detection and cleaning were performed in conjunction with each other as I evaluated each variable in the dataset.

```
# Libraries used

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

import seaborn

import missingno as msno

from scipy import stats

# Import datafile

data = pd.read_csv('C:/Users/ericy/Desktop/medical_raw_data.csv')

# Graph variables to visualize missing data
```

16

```
msno.matrix(data, labels=True)

# Find out shape, create 'Index' variable, Drop 'Unnamed:0'.

data.shape

data['Index'] = pd.Series(range(0, 10000))

data.drop(['Unnamed: 0'], axis=1, inplace=True)

# Move 'Index' to beginning of data.

column_to_move = data.pop('Index')

data.insert(0, 'Index', column_to_move)

data.head()

# Double check which variables have null values

data.isnull().any()

# Total null cases in each variable

data.isnull().sum()

# Variables to address with null values: Children, Age, Income, Soft_drink,
Overweight, Anxiety, Initial_days.

# Rename Item1-Item8 variables to names provided in data's supplemental PDF.

# Also rename 'CaseOrder', 'ReAdmis', 'HighBlood', 'BackPain', and
'TotalCharge' to have the same syntax in variable naming across the dataset

data.rename(columns={'CaseOrder':'Case_order','ReAdmis':'Readmis','HighBlood':'
High_blood','BackPain':'Back_pain','TotalCharge':'Total_charge','Item1':'Timely
_admission', 'Item2':'Timely_treatment', 'Item3':'Timely_visits',
'Item4':'Reliability', 'Item5':'Options', 'Item6':'Hours', 'Item7':'Courteous',
'Item8':'Active_listen'}, inplace=True)

# Check that variables were correctly renamed

data.info()

# Address missing values in 'Soft_drink'

print(data['Soft_drink'])

data['Soft_drink'].fillna(0, inplace=True)

data.isnull().sum()

# Address missing values in 'Anxiety'

plt.hist(data['Anxiety'])
```


17

```
data['Anxiety'].fillna(0, inplace=True)

plt.hist(data['Anxiety'])

# Address missing values in 'Overweight'

plt.hist(data['Overweight'])

data['Overweight'].fillna(0, inplace=True)

plt.hist(data['Overweight'])

# Check for null values

data.isnull().sum()

# Address null values in 'Children'.

plt.hist(data['Children'])

seaborn.boxplot(data['Children'])

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

plt.hist(data['Children'])

data.isnull().sum()

# Address null values in 'Age'.

plt.hist(data['Age'])

seaborn.boxplot(data['Age'])

data['Age'].fillna(method='backfill', inplace=True)

data.isnull().sum()

data['Age'].fillna(method='ffill', inplace=True)

data.isnull().sum()

plt.hist(data['Age'])

seaborn.boxplot(data['Age'])

# Address null values in 'Income'

plt.hist(data['Income'])

seaborn.boxplot(data['Income'])

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

plt.hist(data['Income'])
```

18

```
seaborn.boxplot(data['Income'])

# Begin analyzing 'Initial_days'
plt.hist(data['Initial_days'])
seaborn.boxplot(data['Initial_days'])
data['Initial_days'].fillna(method='backfill', inplace=True)
plt.hist(data['Initial_days'])
seaborn.boxplot(data['Initial_days'])

# Matrix showing no missing values
msno.matrix(data, labels=True)

data.isnull().sum()

data.info()

data.head()

plt.hist(data['Readmis'])

# Reexpression of 'Readmis' data as numeric
data['Readmis'] = data['Readmis'].astype(str)
data['Readmis'].replace(('Yes','No'), (1,0), inplace=True)
plt.hist(data['Readmis'])

# Reexpression of 'Soft_drink' data as numeric
data['Soft_drink'] = data['Soft_drink'].astype(str)
data['Soft_drink'].replace(('Yes','No'), (1,0), inplace=True)
plt.hist(data['Soft_drink'])

# Reexpression of 'High_blood' data as numeric
plt.hist(data['High_blood'])
data['High_blood'] = data['High_blood'].astype(str)
data['High_blood'].replace(('Yes','No'), (1,0), inplace=True)
plt.hist(data['High_blood'])

# Reexpression of 'Stroke' data as numeric
plt.hist(data['Stroke'])
```

19

```
data['Stroke'] = data['Stroke'].astype(str)

data['Stroke'].replace(('Yes','No'), (1,0), inplace=True)

# Reexpression of 'Arthritis' data as numeric

plt.hist(data['Arthritis'])

data['Arthritis'] = data['Arthritis'].astype(str)

data['Arthritis'].replace(('Yes','No'), (1,0), inplace=True)

plt.hist(data['Arthritis'])

# Reexpression of 'Diabetes' data as numeric

plt.hist(data['Diabetes'])

data['Diabetes'] = data['Diabetes'].astype(str)

data['Diabetes'].replace(('Yes','No'), (1,0), inplace=True)

plt.hist(data['Diabetes'])

# Reexpression of 'Hyperlipidemia' data as numeric

plt.hist(data['Hyperlipidemia'])

data['Hyperlipidemia'] = data['Hyperlipidemia'].astype(str)

data['Hyperlipidemia'].replace(('Yes','No'), (1,0), inplace=True)

# Reexpression of 'Back_pain' data as numeric

plt.hist(data['Back_pain'])

data['Back_pain'] = data['Back_pain'].astype(str)

data['Back_pain'].replace(('Yes','No'), (1,0), inplace=True)

plt.hist(data['Back_pain'])

# Reexpression of 'Allergic_rhinitis' as numeric

plt.hist(data['Allergic_rhinitis'])

data['Allergic_rhinitis'] = data['Allergic_rhinitis'].astype(str)

data['Allergic_rhinitis'].replace(('Yes','No'), (1,0), inplace=True)

plt.hist(data['Allergic_rhinitis'])

# Reexpression of 'Reflux_esophagitis' data as numeric

plt.hist(data['Reflux_esophagitis'])
```

20

```
data['Reflux_esophagitis'] = data['Reflux_esophagitis'].astype(str)

data['Reflux_esophagitis'].replace(('Yes','No'), (1,0), inplace=True)

plt.hist(data['Reflux_esophagitis'])

# Reexpression of 'Asthma' data as numeric.

plt.hist(data['Asthma'])

data['Asthma'] = data['Asthma'].astype(str)

data['Asthma'].replace(('Yes','No'), (1,0), inplace=True)

plt.hist(data['Asthma'])

# Reexpress 'Employment' data as numeric.

plt.hist(data['Employment'])

data['Employment'] = data['Employment'].astype(str)

data['Employment'].replace(('Full Time','Retired', 'Unemployed', 'Student',
'Part Time'), (1, 2, 3, 4, 5), inplace=True)

plt.hist(data['Employment'])

# Reexpress 'Marital' data as numeric

plt.hist(data['Marital'])

data['Marital'] = data['Marital'].astype(str)

data['Marital'].replace(('Divorced','Married', 'Widowed', 'Never Married',
'Separated'), (1, 2, 3, 4, 5), inplace=True)

# Reexpress 'Gender' data as numeric

plt.hist(data['Gender'])

data['Gender'] = data['Gender'].astype(str)

data['Gender'].replace(('Male','Female', 'Prefer not to answer'), (1, 2, 3),
inplace=True)

# Reexpress 'Initial_admin' as numeric data

plt.hist(data['Initial_admin'])

data['Initial_admin'] = data['Initial_admin'].astype(str)

data['Initial_admin'].replace(('Emergency Admission','Elective Admission',
'Observation Admission'), (1, 2, 3), inplace=True)

plt.hist(data['Initial_admin'])
```

21

```
# Reexpress 'Complication_risk' data as numeric

plt.hist(data['Complication_risk'])

data['Complication_risk'] = data['Complication_risk'].astype(str)

data['Complication_risk'].replace(('Low', 'Medium', 'High'), (1, 2, 3),
inplace=True)

plt.hist(data['Complication_risk'])

# Reexpress 'Services' data as numeric

plt.hist(data['Services'])

data['Services'] = data['Services'].astype(str)

data['Services'].replace(('Blood Work', 'Intravenous', 'CT Scan', 'MRI'), (1, 2,
3, 4), inplace=True)

plt.hist(data['Services'])

# Reexpress 'Area' data as numeric

plt.hist(data['Area'])

data['Area'] = data['Area'].astype(str)

data['Area'].replace(('Suburban', 'Urban', 'Rural'), (1, 2, 3), inplace=True)

plt.hist(data['Area'])

# Reexpress 'Education' data as numeric

plt.hist(data['Education'])

data['Education'] = data['Education'].astype(str)

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

plt.hist(data['Education'])

# Round 'Income' case entries

data['Income'].round()

data['Income'] = data['Income'].astype('int64')

# Round 'VitD_levels' case entries

data['VitD_levels'].round()
```

22

```
data['VitD_levels'] = data['VitD_levels'].astype('int64')

data['VitD_levels'].head()

# Round 'Initial_days' case entries

data['Initial_days'].round()

data['Initial_days'] = data['Initial_days'].astype('int64')

data['Initial_days'].head()

# Round 'Total_charge' case entries

data['Total_charge'].round()

data['Total_charge'] = data['Total_charge'].astype('int64')

data['Total_charge'].head()

# Round 'Additional_charges' case entries

data['Additional_charges'].round()

data['Additional_charges'] = data['Additional_charges'].astype('int64')

data['Additional_charges'].head()

# Convert 'Children', 'Age', 'Education', 'Readmis', 'Soft_drink',
'High_blood', 'Stroke',

# 'Overweight', 'Arthritis', 'Diabetes', and 'Anxiety' to int64 datatype

data['Children'] = data['Children'].astype('int64')

data['Age'] = data['Age'].astype('int64')

data['Education'] = data['Education'].astype('int64')

data['Readmis'] = data['Readmis'].astype('int64')

data['Soft_drink'] = data['Soft_drink'].astype('int64')

data['High_blood'] = data['High_blood'].astype('int64')

data['Stroke'] = data['Stroke'].astype('int64')

data['Overweight'] = data['Overweight'].astype('int64')

data['Arthritis'] = data['Arthritis'].astype('int64')

data['Diabetes'] = data['Diabetes'].astype('int64')

data['Anxiety'] = data['Anxiety'].astype('int64')

# export dataset that has been cleaned so far to CSV
```

23

```
data.to_csv('C:/Users/ericy/Desktop/partial_clean.csv')

# Begin outlier analysis by Z-Score

# z-score code format and syntax referenced from Scipy.org (Scipy.org, n.d.)

# Calculate and print Z-Scores for all quantitative variables

data['Age_z'] = stats.zscore(data['Age'])

Agez = data.query('Age_z > 3 | Age_z < -3')

Agez.head()

data['Children_z'] = stats.zscore(data['Children'])

Childrenz = data.query('Children_z > 3 | Children_z < -3')

Childrenz.sort_values(['Children_z'], ascending = False)

data['Income_z'] = stats.zscore(data['Income'])

Incomenz = data.query('Income_z > 3 | Income_z < -3')

Incomenz.sort_values(['Income_z'], ascending = False)

data['VitD_levels_z'] = stats.zscore(data['VitD_levels'])

VitD_levels_z = data.query('VitD_levels_z > 3 | VitD_levels_z < -3')

VitD_levels_z.sort_values(['VitD_levels_z'], ascending = False)

data['Doc_visits_z'] = stats.zscore(data['Doc_visits'])

Doc_visits_z = data.query('Doc_visits_z > 3 | Doc_visits_z < -3')

Doc_visits_z.sort_values(['Doc_visits_z'], ascending = False)

data['Full_mealz'] = stats.zscore(data['Full_meals_eaten'])

Full_mealz = data.query('Full_mealz > 3 | Full_mealz < -3')

Full_mealz.sort_values(['Full_mealz'], ascending = False)

data['VitD_suppz'] = stats.zscore(data['VitD_supp'])

VitD_suppz = data.query('VitD_suppz > 3 | VitD_suppz < -3')

VitD_suppz.sort_values(['VitD_suppz'], ascending = False)

data['Initial_days_z'] = stats.zscore(data['Initial_days'])

Initial_days_z = data.query('Initial_days_z > 3 | Initial_days_z < -3')

Initial_days_z.sort_values(['Initial_days_z'], ascending = False)
```

24

```
data['Total_charge_z'] = stats.zscore(data['Total_charge'])

Total_charge_z = data.query('Total_charge_z > 3 | Total_charge_z < -3')

Total_charge_z.sort_values(['Total_charge_z'], ascending = False)

data['Additional_charges_z'] = stats.zscore(data['Additional_charges'])

Additional_charges_z = data.query('Additional_charges_z > 3 |
Additional_charges_z < -3')

Additional_charges_z.sort_values(['Additional_charges_z'], ascending = False)

data['Population_z'] = stats.zscore(data['Population'])

Population_z = data.query('Population_z > 3 | Population_z < -3')

Population_z.sort_values(['Population_z'], ascending = False)

data['Zip_z'] = stats.zscore(data['Zip'])

Zip_z = data.query('Zip_z > 3 | Zip_z < -3')

Zip_z.sort_values(['Zip_z'], ascending = False)

data['Lat_z'] = stats.zscore(data['Lat'])

Lat_z = data.query('Lat_z > 3 | Lat_z < -3')

Lat_z.sort_values(['Lat_z'], ascending = False)

data['Lng_z'] = stats.zscore(data['Lng'])

Lng_z = data.query('Lng_z > 3 | Lng_z < -3')

Lng_z.sort_values(['Lng_z'], ascending = False)

data['Options_z'] = stats.zscore(data['Options'])

Options_z = data.query('Options_z > 3 | Options_z < -3')

Options_z.sort_values(['Options_z'], ascending = False)

data['Timely_admission_z'] = stats.zscore(data['Timely_admission'])

Timely_admission_z = data.query('Timely_admission_z > 3 | Options_z < -3')

Timely_admission_z.sort_values(['Timely_admission_z'], ascending = False)

data['Timely_treatment_z'] = stats.zscore(data['Timely_treatment'])

Timely_treatment_z = data.query('Timely_treatment_z > 3 | Options_z < -3')

Timely_treatment_z.sort_values(['Timely_treatment_z'], ascending = False)

data['Timely_visits_z'] = stats.zscore(data['Timely_visits'])
```


25

```
Timely_visits_z = data.query('Timely_visits_z > 3 | Timely_visits_z < -3')
Timely_visits_z.sort_values(['Timely_visits_z'], ascending = False)
data['Reliability_z'] = stats.zscore(data['Reliability'])
Reliability_z = data.query('Reliability_z > 3 | Reliability_z < -3')
Reliability_z.sort_values(['Reliability_z'], ascending = False)
data['Hours_z'] = stats.zscore(data['Hours'])
Hours_z = data.query('Hours_z > 3 | Hours_z < -3')
Hours_z.sort_values(['Hours_z'], ascending = False)
data['Courteous_z'] = stats.zscore(data['Courteous'])
Courteous_z = data.query('Courteous_z > 3 | Courteous_z < -3')
Courteous_z.sort_values(['Courteous_z'], ascending = False)
data['Active_listen_z'] = stats.zscore(data['Active_listen'])
Active_listen_z = data.query('Active_listen_z > 3 | Active_listen_z < -3')
Active_listen_z.sort_values(['Active_listen_z'], ascending = False)

# Beginning of PCA

# CSV with variables and z-scores removed in preparation for PCA
med = pd.read_csv('C:/Users/ericy/Desktop/pca_1.csv')
med.info()

# Define variables for PCA

# PCA code and process referenced from textbook (Larose, Larose. 2019. pages
179-182).

med =
med[['Doc_visits','VitD_supp','Initial_days','Total_charge','Additional_charges',
', 'Timely_admission','Timely_treatment','Timely_visits','Reliability','Options'
', 'Hours','Courteous','Active_listen']]

# Normalize data - scales data

med_normalized = (med-med.mean())/med.std()

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

pca.fit(med_normalized)

med_pca = pd.DataFrame(pca.transform(med_normalized),
```

```

columns=['PC1','PC2','PC3','PC4','PC5','PC6','PC7','PC8','PC9','PC10','PC11','P
C12','PC13'])

loadings = pd.DataFrame(pca.components_.T,

                        columns
=['PC1','PC2','PC3','PC4','PC5','PC6','PC7','PC8','PC9','PC10','PC11','P
C13'],

                        index=med.columns)

loadings

cov_matrix = np.dot(med_normalized.T, med_normalized) / med.shape[0]

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

plt.plot(eigenvalues)

plt.xlabel('number of components')

plt.ylabel('eigenvalue')

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

plt.show()

print(eigenvalues)

plt.plot(pca.explained_variance_ratio_)

plt.xlabel('number of components')

plt.ylabel('explained variance')

plt.show()

```

Part 3: Data Cleaning

D1. Cleaning Findings and D2 Mitigation Methods

The actions taken to clean the entire dataset for the `medical_raw_data.csv` file are outlined below.

Actions taken are broken down by variable, giving the reason for doing so, and the function used in Python.

Table D1-2a- Variables cleaned, what action was taken, why it was taken, and the function used to complete this.

Variable	Action Taken	Justification for Mitigation	Python Function Used
Column_1 Unnamed variable	Column was dropped from the dataset.	It was redundant. Column #2, 'CaseOrder' has the same value and purpose.	.drop()
Index	Variable added to the dataset	0 Indexes case in dataset	.insert()
Children	Find data shape and null values, ensuring that data shape and integrity remain intact. Address missing values.	The variable is not useful for further analysis with missing data values.	.fillna() using median values to maintain positive skew shape of data.
Age	Find data shape and null values, ensuring that data shape and integrity remain intact. Address missing values.	The variable is not useful for further analysis with missing data values.	.fillna() using 'backfill' and 'ffill' to maintain uniform distribution of data.
Area	Rename Suburban = 1 Urban = 2 Rural = 3	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Education	Education level names will be replaced with numbers. 'No Schooling Completed' = 0 'Nursery School to 8th Grade' = 1 '9th Grade to 12th Grade, No Diploma' = 2 'GED or Alternative Credential' = 3 'Regular High School Diploma' = 4 'Some College, Less than 1 Year' = 5 'Some College, 1 or More Years No Degree' = 6 'Professional School Degree' = 7 'Associate's Degree' = 8 'Bachelor's Degree' = 9 'Master's Degree' = 10	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()

	'Doctorate's Degree' =11		
Income	Find data shape and null values, ensuring that data shape and integrity remain intact. Address missing values.	The variable is not useful for further analysis with missing data values.	.fillna() using median to maintain positive skew shape of data.
Initial_days	Find data shape and null values, ensuring that data shape and integrity remain intact. Address missing values.	The variable is not useful for further analysis with missing data values.	.fillna() using backfill method to maintain bimodal shape of data.
CaseOrder	Changed variable name to 'Case_order'.	This was done to have consistent Variable naming syntax across the dataset.	.rename()
Area	Changed Values of: 'Suburban' to 1. 'Urban' to 2. 'Rural' to 3.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Employment	Changed Values of: 'Full Time' to 1. 'Retired' to 2. 'Unemployed' to 3. 'Student' to 4. 'Retired' to 5.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Marital	Changed Values of: 'Married' to 1. 'Divorced' to 2. 'Widowed' to 3. 'Never Married' to 4. 'Separated' to 5.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Gender	Changed Values of: 'Female' to 1. 'Male' to 2. 'Prefer not to answer' to 3.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
ReAdmis	Changed Values of: 'No' to a 0. 'Yes' to a 1.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
High_blood	Changed Values of: 'No' to a 0. 'Yes' to a 1.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()

Soft_drink	Changed values of: 'NA' to 0.	My reasoning for choosing to value each 'NA' entry as 0 is that patients know if they had a soda or not. An answer of 'NA' is a logical case entry equitable to 0 in this instance.	.fillna() using 0 as replacement
Initial_admin	Chaged Values of: 'Emergency Admission' to 3. 'Elective Admission' to 2. 'Observation Admission' to 1	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Stroke	Changed Values of: 'No' to 0. 'Yes' to 1.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Complication_risk	Changed Values of: 'High' to 3. 'Medium' to 2. 'Low' to 1.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Overweight	Changed Values of: 'NA' to a 0.	My reasoning for choosing to value each 'NA' entry as a 0 is that the individual evaluating the patient knows if they are overweight or not. An answer of 'NA' is a logical case entry equitable to 0 in this instance.	.fillna() using 0 as replacement
Arthritis	Changed Values of: 'No' to 0. 'Yes' to 1.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Diabetes	Changed Values of: 'No' to 0. 'Yes' to 1.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Hyperlipidemia	Changed Values of: 'No' to 0. 'Yes' to 1.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
BackPain	Changed variable name to 'Back_pain'. Changed Values of:	This was done to have consistent Variable naming syntax across the dataset. This quantifies the variable	.rename() .replace()

	'No' to 0. 'Yes' to 1.	cases, making it easier to use the data for analysis.	
Anxiety	Changed Values of: 'NA' to 0.	My reasoning for choosing to value each 'NA' entry as a 0 is that the patient knows if they have a medical diagnosis of Anxiety or not. An answer of 'NA' is a logical case entry equitable to 0 in this instance.	.fillna() using 0 as replacement
Allergic_rhinitis	Changed Values of: 'No' to 0. 'Yes' to 1.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Reflux_esophagitis	Changed Values of: 'No' to 0. 'Yes' to 1.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Asthma	Changed Values of: 'No' to 0. 'Yes' to 1.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Services	Changed Values of: 'Blood Work' to 1. 'Intravenous' to 2. 'CT Scan' to 3. 'MRI' to 4.	This quantifies the variable cases, making it easier to use the data for analysis.	.replace()
Item1	Changed variable name to 'Timely_admissions'.	The variable name was in the corresponding PDF that was downloaded with the dataset. Changing the name makes the data responses less ambiguous, and more useful for analysis.	.rename()
Item2	Changed variable name to 'Timely_treatment'.	The variable name was in the corresponding PDF that was downloaded with the dataset. Changing the name makes the data responses less ambiguous, and more useful for analysis.	.rename()
Item3	Changed variable name to 'Timely_visits'.	The variable name was in the corresponding PDF that was downloaded with the dataset. Changing the name makes the data responses less	.rename()

		ambiguous, and more useful for analysis.	
Item4	Changed variable name to 'Reliability'.	The variable name was in the corresponding PDF that was downloaded with the dataset. Changing the name makes the data responses less ambiguous, and more useful for analysis.	.rename()
Item5	Changed variable name to 'Options'.	The variable name was in the corresponding PDF that was downloaded with the dataset. Changing the name makes the data responses less ambiguous, and more useful for analysis.	.rename()
Item6	Changed variable name to 'Hours_of_treatment'.	The variable name was in the corresponding PDF that was downloaded with the dataset. Changing the name makes the data responses less ambiguous, and more useful for analysis.	.rename()
Item7	Changed variable name to 'Courteous_staff'.	The variable name was in the corresponding PDF that was downloaded with the dataset. Changing the name makes the data responses less ambiguous, and more useful for analysis.	.rename()
Item8	Changed variable name to 'Doctor_listened'.	The variable name was in the corresponding PDF that was downloaded with the dataset. Changing the name makes the data responses less ambiguous, and more useful for analysis.	.rename()
Income, VitD_levels, Initial_days, Total_charge, Additional_charges	Round entries to nearest integer.	Each variable was analyzed and I decided the dataset would be easier to analyze with these variables rounded up to be integers.	.round()
Income, VitD_levels, Initial_days, Total_charge,	Convert datatype to int64	After cleaning up these variables by numerical conversion and rounding, they are now integers. To	.astype('int64')

Additional_charges, Children, Age, Education, Readmis, Soft_drink, High_blood, Stroke, Overweight, Arthritis, Diabetes, Anxiety		clean the dataset further, and make it more accessible for analysis the decision was made to convert to integers.	
--	--	---	--

Z-scores were analyzed to determine the number of outliers in all quantitative variables. I determined that noting and keeping the outliers in the dataset for each variable will be the best option for this analysis. Some of the variables, such as Children and Income, had individual case reportings as high as 10 and over \$200,000, respectively. These numbers, while higher than average, do not seem to be the result of faulty information being recorded. To address VitD_levels entries, according to an article by Harvard Health Blog, optimal levels of Vitamin D in blood serum is estimated to be 40-60 ng/mL - with levels as low as 12.5 ng/mL being an appropriate cutoff marking Vitamin D deficiency (Tello, 2020). These cited reputable figures make our outliers look less like outliers, and more like normal variation from person to person. As such I decided to keep the outliers in the dataset. For all other outliers, please refer to the given reason under the column 'Why' in the table below.

Table D1-2b - Z score results, action, and reasoning.

Variable	# of outliers	What Action Is Taken	Justification
Age	0	None	No values to address
Children	303	Outliers are left in the dataset.	With a high z-score of 4.27, or 10 children, outliers are assumed to be valid data points.
Income	180	Outliers are left in the dataset.	The dataset shows a wide distribution of incomes. Data outliers are assumed to be valid.
VitD_Levels	500	Outliers are left in the dataset.	The dataset shows a wide distribution of VitD_levels.

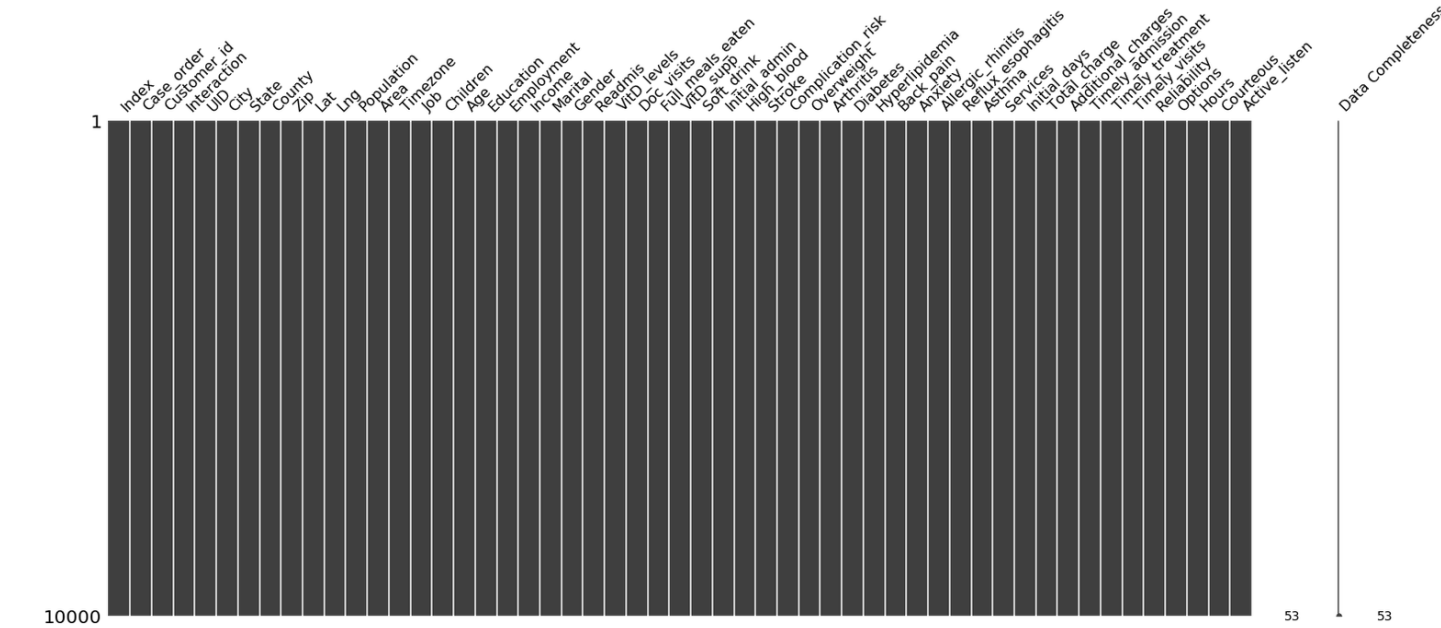
			Data outliers are assumed to be valid.
Doc_visits	8	Outliers are left in the dataset.	There are few outliers, they are assumed to be valid.
Full_meals_eaten	33	Outliers are left in the dataset.	There are few outliers, they are assumed to be valid.
VitD_suppl	77	Outliers are left in the dataset.	There are few outliers, they are assumed to be valid.
Initial_days	0	None	No values to address
Total_charge	276	Outliers are left in the dataset.	Medical charges vary broadly. Data is assumed to be valid for this analysis.
Population	218	Outliers are left in the dataset.	This is a variable that isn't going to be used in the analysis - I would drop it if the PA didn't require analyzing every variable. Population varies broadly - and values appear to not be valid. They would need to be synced with lat., lng, city, and zip to make more sense for analysis. In a professional environment the analysis cost would not justify the results in this particular scenario. But for this case, outliers are left in the dataset.
Zip	0	None	No values to address - but there are major inconsistencies in the variable - such as not enough digits to make a valid zip code. As with population - this variable would be dropped, but for completion purposes is analyzed.
Lat	144	Outliers are left in the dataset.	There are few outliers, they are assumed to be valid.
Lng	98	Outliers are left in the dataset.	There are few outliers, they are assumed to be valid.
Options	13	Outliers are left in the	There are few outliers, they

		dataset.	are assumed to be valid.
Timely_admission	11	Outliers are left in the dataset.	There are few outliers, they are assumed to be valid.
Timely_treatment	12	Outliers are left in the dataset.	There are few outliers, they are assumed to be valid.
Timely_visit	12	Outliers are left in the dataset.	There are few outliers, they are assumed to be valid.
Reliability	12	Outliers are left in the dataset.	There are few outliers, they are assumed to be valid.
Hours	10	Outliers are left in the dataset.	There are few outliers, they are assumed to be valid.
Courteous	11	Outliers are left in the dataset.	There are few outliers, they are assumed to be valid.
Active_listen	12	Outliers are left in the dataset.	There are few outliers, they are assumed to be valid.

D3. Summary of the Outcomes

The summary for each step outlined in part C1 are as follows:

Step 1: Redundant variables have been addressed and variable 'Index' has been added to the dataset as shown in the screenshot below..



Step 2: All necessary variables have been renamed.

Step 3: All variables with applicable qualitative data have been transformed to quantitative data. Applicable data types changed to int64. Shown in the screenshot below.

```

Data columns (total 53 columns):
 #   Column                Non-Null Count  Dtype
---  -
 0   Index                 10000 non-null   int64
 1   Case_order            10000 non-null   int64
 2   Customer_id           10000 non-null   object
 3   Interaction            10000 non-null   object
 4   UID                   10000 non-null   object
 5   City                  10000 non-null   object
 6   State                 10000 non-null   object
 7   County                10000 non-null   object
 8   Zip                   10000 non-null   int64
 9   Lat                   10000 non-null   float64
10   Lng                   10000 non-null   float64
11   Population            10000 non-null   int64
12   Area                  10000 non-null   int64
13   Timezone              10000 non-null   object
14   Job                   10000 non-null   object
15   Children              10000 non-null   int64
16   Age                   10000 non-null   int64
17   Education              10000 non-null   int64
18   Employment            10000 non-null   int64
19   Income                10000 non-null   int64
20   Marital               10000 non-null   int64
21   Gender                10000 non-null   int64
22   Readmis               10000 non-null   int64
23   VitD_levels           10000 non-null   int64
24   Doc_visits            10000 non-null   int64
25   Full_meals_eaten      10000 non-null   int64
26   VitD_supp             10000 non-null   int64
27   Soft_drink            10000 non-null   int64
28   Initial_admin         10000 non-null   int64
29   High_blood            10000 non-null   int64
30   Stroke                10000 non-null   int64
31   Complication_risk     10000 non-null   int64
32   Overweight            10000 non-null   int64
33   Arthritis             10000 non-null   int64
34   Diabetes              10000 non-null   int64
35   Hyperlipidemia        10000 non-null   int64
36   Back_pain             10000 non-null   int64
37   Anxiety               10000 non-null   int64
38   Allergic_rhinitis     10000 non-null   int64
39   Reflux_esophagitis    10000 non-null   int64
40   Asthma                10000 non-null   int64
41   Services              10000 non-null   int64
42   Initial_days          10000 non-null   int64
43   Total_charge          10000 non-null   int64
44   Additional_charges    10000 non-null   int64
45   Timely_admission      10000 non-null   int64
46   Timely_treatment       10000 non-null   int64
47   Timely_visits         10000 non-null   int64
48   Reliability           10000 non-null   int64
49   Options               10000 non-null   int64
50   Hours                 10000 non-null   int64
51   Courteous             10000 non-null   int64
52   Active_listen         10000 non-null   int64
dtypes: float64(2), int64(43), object(8)
memory usage: 4.0+ MB

```

Step 4: All variables with null values were detected and addressed with the proper method. Justification and details were given for explaining the reasoning behind choosing the methods for imputation for each variable.

Step 5: Z-scores were calculated for all quantitative variables. Number of outliers per variable is noted in section D1, and the decision to keep outliers in the dataset was detailed.

Step 6: Required variables for PCA were given, and the code to run PCA was detailed. More information will be detailed regarding this step in section E1.

D4. Mitigation Code

I performed the analysis and cleaning on each variable in order, while going through the dataset. Because of this process, mitigation code is mixed in with the detection code. The code for mitigating anomalies can be referenced under section C4 - Provide the Code of this document. Also, please refer to the attached PDF containing all executable code included in this submission detailing and visualizing data anomalies and their performed cleaning.

D5. Clean Data

Included as an attachment in this submission is a .csv file containing the cleaned data set created from the raw data.

D6 and D7. Limitations and the Impact of the Limitations

Limitations of the data-cleaning process implemented are as follows. The conversion process of qualitative to categorical creates variables that can be manipulated more easily by humans and robots. They do, however, lose meaning through loss of context. A good example of this is the variable 'Services'. Before this process the responses for services included 'Blood Work', 'Intravenous', etc. and were converted to '1', '2', etc., which makes the variable meaningless without context. Analysts and researchers using this data will find it easier to use this quantitative data, at the limitation of having to look up what each number stands for.

Another limitation is the outcome from decisions made with outliers. The decision to delete, impute, or leave outliers in all have very different outcomes on the size and potential quality of the dataset as a whole. In this dataset, all outliers were noted but kept in. This comes with the limitation that outliers could potentially affect PCA results, or could potentially paint an inaccurate picture of the variable.

The method of imputation chosen for each variable also has an effect on the dataset. Histograms were plotted before and after each variable had its null values imputed to ensure the shape of the variables' data

remained the same, but there are many ways to fill these missing values. From choosing to impute with something considered simple such as the median or mode to using an advanced method like KNN or MICE - the variable fit will vary from tool used to tool used. All efforts were taken to make sure the data retained its basic before shape after filling in missing values - but there isn't one perfect way of doing this that doesn't come with limitations.

Another limitation in the cleaning process falls onto the category of variables that could be described as geographic/demographic. This includes variables Lat, Lng, Population, County, City, and Timezone, There are inconsistencies that don't show up as outliers in Zip, such as entries with only 4 digits. The variable 'Population' is rife with non-null inaccuracies, with many cities showing a population of 0. Or large metros such as Dallas, having a multitude of listed populations - none of them accurate. Lat and Lng could be off on a case-by-case basis for the zip and city listed for the case. These variables would need an inordinate amount of time spent on them to be turned into calibrated, high-quality, useful data points. The scope of this task in comparison to the return provided to the project would not be viable - and as such is a noted limitation in this dataset. These variables are, however, useful for providing a basic starting point for where these entries came from geographically. This is why I have left them in the dataset.

E1. Principal Components

The screenshot below shows all principal components and their variable weight and composition..

```
[8]: med_pca = pd.DataFrame(pca.transform(med_normalized),
                           columns=['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10', 'PC11', 'PC12', 'PC13'])

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

loadings
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13
Doc_visits	0.007069	-0.002075	-0.013897	0.551134	-0.750294	0.362101	-0.019941	-0.026724	-0.025656	0.007265	-0.010368	-0.003168	-0.002215
VitD_supp	-0.004949	0.019578	0.034108	0.545353	0.650617	0.524840	0.030928	0.030766	0.013852	-0.003816	0.010258	-0.006602	-0.001276
Initial_days	-0.016866	0.425164	0.562897	-0.043657	-0.030162	0.021512	-0.011771	-0.006577	0.000766	-0.007445	0.031354	-0.696589	-0.111189
Total_charge	-0.014241	0.440002	0.552356	-0.010146	-0.022778	-0.003120	-0.000712	-0.014128	0.000295	0.011957	-0.028224	0.699144	0.102588
Additional_charges	0.003986	0.034518	0.020067	0.629137	0.092341	-0.768731	0.004383	-0.041244	0.005803	0.014262	-0.015063	-0.030291	-0.001239
Timely_admission	0.454784	-0.232816	0.184023	0.002111	0.007337	-0.003075	-0.095714	-0.076403	-0.010802	0.086216	0.181731	0.131544	-0.795083
Timely_treatment	0.428496	-0.226595	0.186167	0.004032	0.004817	-0.002444	-0.146858	-0.134481	-0.062202	0.102062	0.625524	-0.052843	0.531349
Timely_visits	0.395301	-0.228728	0.188430	-0.004630	0.027615	0.010274	-0.204619	-0.212429	-0.238900	-0.433423	-0.620798	-0.045733	0.188733
Reliability	0.152243	0.437651	-0.346530	-0.019568	0.047671	0.027333	-0.365196	-0.361566	-0.387968	0.483537	-0.113822	-0.015611	-0.013180
Options	-0.190134	-0.463555	0.355510	-0.004262	-0.000015	0.000493	0.124501	0.058344	-0.132365	0.694576	-0.307619	-0.029299	0.090615
Hours	0.410398	0.134827	-0.093755	-0.005404	-0.014136	0.013554	-0.050728	0.061982	0.796740	0.266844	-0.274555	-0.035727	0.122965
Courteous	0.356642	0.150254	-0.089585	0.013667	-0.017869	-0.029628	0.035179	0.846287	-0.335176	0.068621	-0.060967	-0.010726	0.049978
Active_listen	0.312688	0.137892	-0.094741	-0.018968	-0.013051	0.008766	0.879324	-0.270498	-0.151259	0.040836	-0.037450	-0.017563	0.031502

```
[10]: cov_matrix = np.dot(med_normalized.T, med_normalized) / med.shape[0]

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

[12]: plt.plot(eigenvalues)
plt.xlabel('number of components')
plt.ylabel('eigenvalue')
plt.axhline(y=1, color='orange')
plt.show()
```

The four viable principal components from this table are listed, with their eigenvalue and top positively contributing variables listed.

PC Name	Eigenvalue	Top Variables
PC1	2.95	Timely_admission, Timely_treatment, Hours, Courteous
PC2	1.66	Initial_days, Total_charge, Reliability
PC3	1.62	Initial_days, Total_charge, Options
PC4	1.01	Doc_visits, VitD_supp, Additional_charges

E2. Criteria Used

This method was drawn from the course's textbook, *Data Science Using Python and R* (Larose, Larose, 2019, p 179-182). The criteria used to identify the principal components began with looking at the question of

40

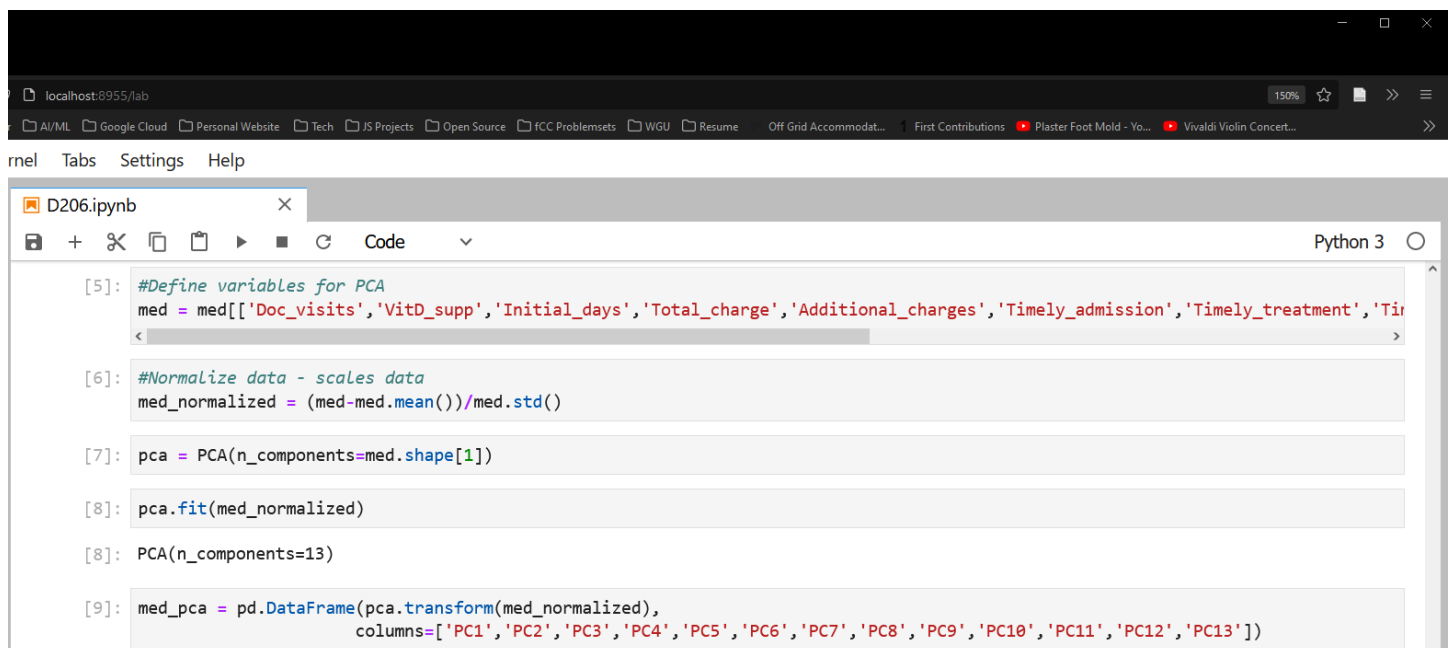
which variables were most likely to be similarly predictive in relation to readmission. The survey responses, variables associated with the patient's medical history and visit length, and the variables associated with cost all seemed to be great candidates. The screenshot below lists all used variables for PCA.

```
[2]: med = pd.read_csv('C:/Users/ericy/Desktop/PCA_1.csv')
```

```
[3]: med.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 13 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Doc_visits             10000 non-null  int64
1   VitD_supp              10000 non-null  int64
2   Initial_days           10000 non-null  int64
3   Total_charge            10000 non-null  int64
4   Additional_charges      10000 non-null  int64
5   Timely_admission        10000 non-null  int64
6   Timely_treatment        10000 non-null  int64
7   Timely_visits           10000 non-null  int64
8   Reliability             10000 non-null  int64
9   Options                10000 non-null  int64
10  Hours                  10000 non-null  int64
11  Courteous               10000 non-null  int64
12  Active_listen           10000 non-null  int64
dtypes: int64(13)
memory usage: 1015.8 KB
```

These variables were then normalized and had PCA performed as shown in the screenshot below.

A screenshot of a Jupyter Notebook interface. The top bar shows the browser address as localhost:8955/lab. Below the browser tabs, the Jupyter Notebook tabs show 'D206.ipynb' is active. The code editor displays the following Python code:

```
[5]: #Define variables for PCA
med = med[['Doc_visits', 'VitD_supp', 'Initial_days', 'Total_charge', 'Additional_charges', 'Timely_admission', 'Timely_treatment', 'Timely_visits', 'Reliability', 'Options', 'Hours', 'Courteous', 'Active_listen']]

[6]: #Normalize data - scales data
med_normalized = (med-med.mean())/med.std()

[7]: pca = PCA(n_components=med.shape[1])

[8]: pca.fit(med_normalized)

[8]: PCA(n_components=13)

[9]: med_pca = pd.DataFrame(pca.transform(med_normalized),
                           columns=['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10', 'PC11', 'PC12', 'PC13'])
```

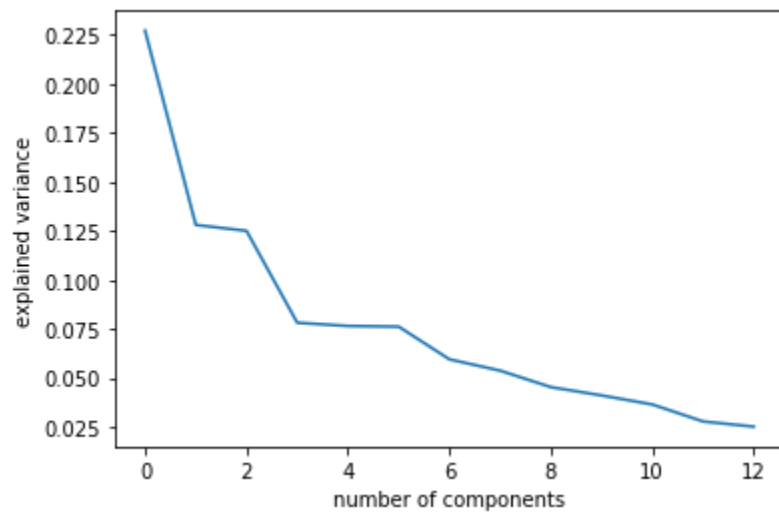
The loadings were visualized, showing the relationship, correlation value, and contribution for each variable in the PC as shown in section E1. The viable principal components were identified by determining their

41

eigenvalue and explained variance.. Below is a screenshot of the Scree plot visualizing this process, the printout of each PC's eigenvalue, and the Scree plot visualizing each PC's Explained Variance.



Scree plot, Explained Variance for each principal component



PC's with eigenvalues > 1 and significant explained variance were determined to be viable. This was determined to be PC1, PC2, PC3, and PC4.

E3. Benefits

The benefit of our PCA lies in the reduction of variables necessary for predicting readmission rates. In the healthcare administration industry this equates into more accessible information by decreasing dataset size and complexity. The dataset also helps the customer experience, which in turn can reduce readmissions.

For example survey questions can be added and removed to better predict how well the customer really feels their experience was. Customers with responses predicting readmission can be reached out to preemptively to address concerns, possibly negating readmission. Additionally, data points such as Initial_days and Total_charge can be used for predictive purposes in identifying customers who are more likely to be candidates for readmission. This provides healthcare providers with an additional tool in assessing patient's health outcomes. The PCA results can help consolidate these variables into predictive and actionable insights. PCA answers our question in section A and shows which variables can be grouped together to predict readmission.

Part 4: Supporting Documents

F. Video

Attached to the PA submission is a Panopto recording of my Python code used to analyze and clean the dataset.

G. Sources of Third-Party Code

Scipy.org. (No Date). *Scipy.stats.zscore*. [scipy.stats.zscore(a, axis=0, ddof=0, nan_policy='propagate')].
<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.zscore.html>

Larose, Chantal D., Larose, Daniel T.. 2019. *Data Science Using Python and R (1st Edition)* (pp. 179-182). John Wiley & Sons.

H. Sources

Rau, Jordan. (2021, October 28). *Medicare Punishes 2,499 Hospitals for High Readmissions*.

<https://khn.org/news/article/hospital-readmission-rates-medicare-penalties/>

Mahto, M., Hogan, S.K., Hatfield, S., Coppola, M., Kulkarni, A. (2020, February 6) *The role of machine learning in qualitative data analysis*.

<https://www2.deloitte.com/us/en/insights/focus/technology-and-the-future-of-work/machine-learning-qualitative-data.html>

Hashi, Zakarie. (2018, July 28). *Difference between MAR, MCAR, and MNAR missing data*.

<https://www.linkedin.com/pulse/difference-between-mar-mcar-mnar-missing-data-zakarie-a-hash/>

Tello, Monique. (2020, April 16). *Vitamin D: What's the "right" level?*

<https://www.health.harvard.edu/blog/vitamin-d-whats-right-level-2016121910893>

Larose, Chantal D., Larose, Daniel T.. 2019. *Data Science Using Python and R (1st Edition)* (pp. 179-182).

John Wiley & Sons.