D206 – Data Cleaning

WGU M.S. Data Analytics

Lyssa Kline

April 7, 2024

A, Research Question

My research question is, “What demographics and lifestyle habits influence hospital admissions? This research question is crucial to the business because it identifies hospital admissions and readmissions. By analyzing demographics and lifestyle habits, the company can better evaluate the needs and services required for these patients.

In the future, this question could be expanded to explore additional areas of opportunity, such as the number of doctor visits, strokes, readmissions, and additional services required.

B, Describe Variables

The medical dataset used in this report contains 52 columns, each containing variables relating to a patient’s medical history and the services required. The below summarization describes each field that is contained inside of the acquired dataset, determines the data type (quantitative vs. qualitative), and gives an example of the output. See summarization below.

- CaseOrder (quantitative, example: 1) – This field is a quantitative value consisting of unique numbers for each customer record inside of the database. The field functions as an index field.
- Customer_id (qualitative, example: C412403) – this field is a qualitative value consisting of a unique number for each customer stored inside of the table and is used to look up a single customer inside of the table.
- Interaction (qualitative, example: 8cd49b13-f45a-4b47-a2bd-173ffa932c2f) – this column is also an ID, which identifies unique information related to patient transactions, procedures, and admissions.
- UID (qualitative, example: 3a83ddb66e2ae73798bdf1d705dc0932) – this field is also an ID, which identifies unique information related to patient transactions, procedures, and admissions.
- City (qualitative, example: Eva) – This field shows the patient’s city as listed on the billing statement.
- State (qualitative, example: AL) – This field shows the patient’s state as listed on the billing statement.
- County (qualitative, example: Morgan) – This field shows the patient’s county as listed on the billing statement.
- Zip (qualitative, example: 35621) – This field shows the patient’s zip code as listed on the billing statement.
- Lat (quantitative, example: 34.3496) – This field shows the patient’s geographical latitude location as listed on the billing statement.
- Lng (quantitative, example: -86.72508) – This field shows the patient’s geographical longitude location as listed on the billing statement.
- Population (quantitative, example: 2951) – This field shows the population within a mile radius of the patient, based on public census data.
- Area (qualitative, example: Suburban) – This field shows the patient’s home residence type (i.e. suburban, urban, rural, etc.) based on unofficial census data.
- Timezone (qualitative, example: America/Chicago) – this field shows the patient’s local time zone based on patients’ admissions information.

- Job (qualitative, example: Psychologist, sport and exercise) – This field shows the Patient current job occupation as indicated in the admissions data.
- Children (quantitative, example: 1) – This field shows the number of children in the patient's household, reported in the admissions data.
- Age (quantitative, example: 53) – This field shows the patient's age as reported in the admissions data.
- Education (qualitative, example: Some College, Less than 1 Year) – This field shows the patient's highest education obtained as reported in the admissions data.
- Employment (qualitative, example: Full Time) – This field shows the patient's current employment type (i.e full-time, part-time, etc.) as reported in the admissions data.
- Income (quantitative, example: 86575.93) – This field shows the patient's current income as reported in the admissions data.
- Marital (qualitative, example: Divorced) – This field shows the patient's current marital status (i.e. married, single, divorced) as reported in the admissions data.
- Gender (qualitative, example: Male) – This field shows the customer's self-identification (i.e.male, female, and non-binary) as reported in the admissions data.
- ReAdmis (qualitative, example: No) – this field shows whether a patient has been readmitted within a month of release.
- VitD_levels (quantitative, example: 17.80233049) – This field shows the patient's vitamin D levels, this is taken and recorded by the doctor.
- Doc_visits (quantitative, example: 6) - This field shows the number of times the primary physician visited the patient during the initial hospitalization.
- Full_meals_eaten (quantitative, example: 0) - This field shows the number of meals the patient ate while in the hospital.
- VitD_supp (quantitative, example: 0) - This field shows the number of times that vitamin D supplements were administered.
- Soft_drink (qualitative, example: No) - This field shows whether a patient drinks three or more sodas in a day.
- Initial_admin (qualitative, example: Emergency Admission) - this field shows how the patient was admitted into the hospital initially.
- HighBlood (qualitative, example: Yes) - This field shows if the patient has high blood pressure.
- Stroke (qualitative, example: No) - This field shows if the patient has had a stroke.
- Complication_risk (qualitative, example: Medium) - This field shows the level of complication risk for the patient.
- Overweight (qualitative, example: 0) – This field shows whether the patient is considered overweight based on age, gender, and height.
- Arthritis (qualitative, example: Yes) - This field shows if the patient has arthritis.
- Diabetes (qualitative, example: Yes) - This field shows if the patient has diabetes.
- Hyperlipidemia (qualitative, example: Yes) - this field shows if the patient has hyperlipidemia.
- BackPain (qualitative, example: Yes) - This field shows if the patient has back pain.
- Anxiety (qualitative, example: 0) – This field shows if the patient has an anxiety disorder.
- Allergic_rhinitis (qualitative, example: Yes) - This field shows if the patient has allergic rhinitis.

- Reflux_esophagitis (qualitative, example: Yes) - This field shows if the patient has reflux esophagitis.
- Asthma (qualitative, example: Yes) - This field shows if the patient has asthma.
- Services (qualitative, example: Blood Work) - This field shows the primary service the patient received while hospitalized.
- Initial_days (quantitative, example: 10.58576971) – This field shows the number of days the patient stayed in the hospital during the initial visit.
- TotalCharge (quantitative, example: 3191.048774) - This field shows the amount charged to the patient daily. This includes the amount charged not including specialized treatments.
- Additional_charges (quantitative, example: 17939.40342) - This field shows the average amount charged to the patient for miscellaneous procedures and treatments.
- Item1 (qualitative, example: 3) – This column indicates the patient's response to question 1 in the survey, 8 being 'least important' and 1 being 'most important'. The question was related to timely admissions.
- Item2 (qualitative, example: 3) – This column indicates the patient's response to question 1 in the survey, 8 being 'least important' and 1 being 'most important'. The question was related to timely treatment.
- Item3 (qualitative, example: 3) – This column indicates the patient's response to question 1 in the survey, 8 being 'least important' and 1 being 'most important'. The question was related to timely visits.
- Item4 (qualitative, example: 3) - This column indicates the patient's response to question 1 in the survey, 8 being 'least important' and 1 being 'most important'. The question was related to reliability.
- Item5 (qualitative, example: 4) - This column indicates the patient's response to question 1 in the survey, 8 being 'least important' and 1 being 'most important'. The question was related to options.
- Item6 (qualitative, example: 3) - This column indicates the patient's response to question 1 in the survey, 8 being 'least important' and 1 being 'most important'. The question was related to hours of treatment.
- Item7 (qualitative, example: 3) - This column indicates the patient's response to question 1 in the survey, 8 being 'least important' and 1 being 'most important'. The question was related to courteous staff.
- Item8 (qualitative, example: 4) - This column indicates the patient's response to question 1 in the survey, 8 being 'least important' and 1 being 'most important'. The question was related to evidence of active listening from the doctor.

C, Data Cleaning Plan

Cleaning data for analysis is the process of identifying and correcting errors in a dataset. Cleaning data helps to ensure accuracy, completeness, and consistency. The plan that was used to clean the data for this analysis, was to first find all the data quality issues. This was done by first reviewing the entire dataset to find any duplicates, missing values, outliers, and categorical variables.

C1, Plan Proposal

To adequately clean the data for this analysis, the first step is to identify where all the inconsistencies and data issues are. To do so, importing the raw data file into the workflow will get the process started. After importing and reading the data into the code, the next step is to review the data dictionary provided in the data zip file to start to get an understanding of the data architecture and contents.

The medical dataset was chosen for this analysis. The first steps of this cleaning process consisted of identifying any duplicate values, missing values, outliers, or re-expression of categorical values. To find these values, there are a variety of commonly known Python expressions that can be used to get started. Most of the expressions used help give a general understanding of the consistency of the dataset and the contents inside of it. info() helps give a general overview of the structure and buildup of the data. head() helps you visualize what each column contains and the values within it. describe() helps identify any outliers by showing the statistics of each column, this function is only adequate when assessing quantitative variables. All qualitative, or categorical, variables need an additional assessment. value_counts(), manual inspection, and boxplot() were used for qualitative variables. Value_counts() helps to determine unique values and counts or occurrences of that particular variable. A boxplot() helps graph the variables and visualize any outliers. dtypes lists each column and the associated datatype. duplicated() shows you if there any any duplicate values in a specific field.

To adequately detect all data quality issues, each of these functions discussed was utilized. The entire dataset, including all quantitative variables, was analyzed using the discussed functions. This approach will ensure the detection of data quality issues.

C2, Plan Justification

This plan looks at the data cleaning process from both a broad scope looking at the entire table, as well as a smaller scope by reviewing individually identified variables for errors or inconsistencies. head(), describe() and info() are the best functions to help start to visualize the dataset, these functions allow you to review the data visually and to determine the statistics (i.e. mean, standard deviation, min, max, etc.). The describe function adequately assesses quantitative variables, variables that have numerical values can be assessed with this function to review statistical values. However, qualitative variables, or variables that are categorical and cannot be assessed with numerical statistics must be assessed in additional ways. This plan used a value_count() and boxplot() function to review the qualitative variables visually and objectively.

It was decided best to use the data dictionary that was provided as part of this PA, due to its assistance in understanding what each variable is and the data quality it contains. Using the dictionary alongside the duplicated() and dtypes functions, helped to identify the gaps, or if the null, incomplete, or duplicated values in the variables made sense.

Looking at both the table as a whole and the data dictionary helped to gain a general understanding of the dataset and any anomalies inside of it. The next steps were to start to assess the best way to move forward with cleaning the data for analysis.

C3, Programming Language Justification

Python was chosen for this analysis due to the vast libraries designed for data cleaning, the flexibility and expressiveness it offers, and integration with other tools, as well as current business needs in the workforce. The various features that Python offers make it a great choice for data-cleaning tasks.

To do the initial review and cleaning of the datasets, Pandas, and NumPy were both used vastly to clean the data in this analysis. NumPy is a package used for numerical computing, this is a numerical computing library that helped during this report when needed to build an array or reshape data. Pandas is a library built on top of NumPy that provides data structures and functions for data manipulation and analysis. Pandas is a key package used in data cleaning, exploration, and analysis. Pandas is a powerful library used for data manipulation; this library allowed me to use comprehensive functions used in this report for data cleaning. IPython.display is a module in python that provides enhances features to the python shell, this was imported to be able to visualize the whole dataset in Jupyter Notebook for outliers. For further analysis and cleaning in this workflow, PCA, StandardScaler, matplotlib, and seaborn were used for the visualization, graphing, and principal component analysis components of the data cleaning project. These packages assist in data visualization and reduction techniques needed to clean the data. PCA is a technique for dimensionality reduction and data visualization, this package was needed when inserting the various variables into the PCA data frame to be charted. StandardScaler is a preprocessing technique used for scaling numerical features, this was used to standardize the PCA variables and determine the mean and standard deviation in the PCA analysis. Matplotlib is helpful when creating visualizations in Python, this was used to chart the PCA analysis. Seaborn is a helpful statistical visualization tool; this was used to create a statistical plot in the PCA analysis.

C4, Quality Code

See the code attached 'D206 – C4 – Detection Code.ipynb' for the detection code used.

D, Data Cleaning Summarized

The data cleaning process consisted of identifying and cleaning duplicates, missing values, outliers, and the re-expression of categorical variables. To find all data quality issues and determine the next steps for mitigating these issues, the rubric and the data dictionary played a big part in this assessment. See the summary below for the description of the cleaning process.

D1, Data Quality Issue Findings

To find duplicate values, a review of the data dictionary helped to assess the fields that could be impacted by duplicates. All unique fields in this dataset should be unique and not contain any duplicates to not cause issues in the assessment. CaseOrder, Customer_ID, Interaction, and IUD were determined to be the unique 'index' type fields in the dataset. To determine if these values contained duplicates the function duplicated() was useful in assessing each unique value field.

The process to find null values starts with an initial isnull() function which reviewed the entire dataset for which fields contained null values. From this, I could see that there were several null values in a few of the fields; Children had 2588 nulls, Age had 2414 nulls, Income had 2464 nulls, Soft_Drink had 2467 nulls, Overweight had 982 nulls, Anxiety had 984 nulls, and Initial_Days had 1056 nulls. An additional check on Initial_Days was performed using the isnull to review if there was a need to change these values or not.

To find outliers in the dataset, a spot check of the entire dataset will help to identify where the outliers are. Using the function describe() shows the entire dataset's statistical values, this function shows you the mean, standard deviation, min, and max of each field. Based on the rubric, all quantitative variables must be reviewed for outliers. After reviewing these fields, which as indicated under topic B, were quantitative; it was observed that timezones and VitD_Levels both seemed high. After further review of the VitD_Levels field, it was noted that these values do correlate with each other and are not high. All qualitative variables were assessed using the value_counts() function and a boxplot(). Reviewing these functions helped to visualize each categorical variable for any outliers and occurrences of that variable. All variables seemed adequate after this review. Timezones, however, reflected that there were 26 saved time zones in the dataset, this number is high due to there being only 9 standard time zones in the U.S. The process of cleaning the time zones to adequately reflect 9 standard time zones could also be considered part of the re-expression of categorical variables, however, it will be placed in outliers since it was identified during these steps.

The last data quality assessment that was performed was to find re-expression of categorical variables. This process consisted of identifying and transforming categorical variables into a different representation that is more suitable for analysis. This consisted of using a few different functions to identify which variables could be re-expressed. The dtypes function was used to output all the data types for each of the columns in the table. 25/52 fields were identified to be incorrect with the data types needing to be changed based on the value contained inside of the field. Additionally, a further review was done on a few values to determine if they needed re-expression. The unique() function was performed on City, State, and County to ensure these fields were properly inputted and correlated with the other inputs. The assessment came back clean showing only cities that matched other records, there were not any non-matching variables. If there would have been variables that were non-matching this would have required some additional cleaning. Lastly, during this analysis, there were a few additional data naming issues found that although were not necessary to note and clean for this report, it is a good practice to do so and were decided best to follow. There were a few naming conventions that were hard to follow and not proper. The first that was noticed, were several field names that did not have the proper capitalization. For example, Customer_id could be Customer_ID and it would be cleaner inside the dataset. Additionally, the survey questions were named 'Item1','Item2', etc. This naming was identified as hard to follow and undescriptive. Lastly, one more observation was that the charge fields had 6 decimal places, this tends to be a little too much information and is unneeded when determining and viewing prices of services.

D2, Methods Justification

To treat the values referenced in D1 an in-depth assessment of the rubric and data dictionary were needed to draw conclusions that made sense for the data. For the duplicates, after review, it was determined that each one of these fields contained 10000 unique values and returned a 'false' for all records indicating that there were no duplicates. Therefore, no need to remove any data here.

To best treat the missing values, after reviewing the output against the definition of these variables in the data dictionary and the steps in the rubric, there were no steps taken to remove these null values at this time. It was determined that the null values just meant a customer did not input an answer to that question. It can be noted that changing these values, even to 0 or yes/no, could highly impact the results that come from the analysis. After reviewing, Initial_Days was the only field that, in my opinion, could be changed to 0 without skewing any results. The fillna(0, inplace=True) function was used to replace all null values with 0 for any patients who stayed in the hospital for less than a day. This field was also changed from a float to an integer due to it being a count of days, a fractional day value is not needed here. It was determined that it is best to leave the rest of these fields as null values and omit them from any further analysis. If these fields are deemed important, it should be advised to make this a required input field for the patient filling out the admissions form. This would not affect the initial assessment; however, it should be noted that null values are removed from the dataset at a later point to adequately perform the PCA.

To best treat the outliers, after an assessment of the quantitative variables that were deemed high, time zones were the only field that needed to be cleaned up. There were 26 unique time zones in the dataset, while there are 9 standard time zones in the US. The cleaning process used to treat the abundance of time zones listed was to first determine the unique values using unique(), next placing these in a replace({}) function in an organized manner to be able to view which time zones correlated with each other. I then used a simple Google search to find the standard time zones, including the states that do not follow daylight savings time, this came down to 9 standard time zones (Eastern, Central, Mountain Standard, Mountain, Pacific, Alaska, Hawaii, Adak/Aleutian, and Atlantic). After indicating these I used my judgment on naming conventions and placed each time zone in the correct bucket. It should be advised to auto-populate or use a drop-down option for this field to keep it clean moving forward, a large variety of time zones could affect an analysis done that uses the customer's location information.

For the re-expression of categorical variables, I used both the data dictionary as well as general understanding and workforce knowledge on how to best approach these variables. The first step in cleaning the dataset variables was the change the datatypes based on the values contained inside the field. Category fields have a limited number of unique values, Boolean fields contain yes or no fields, object fields are strings of text, integers are numbers, and float64 are integers with decimals. Using the astype() function, the below list of fields had data type changes.

- CaseOrder – is stored as an int64 but should be an object rather than an integer, this is an index field.
- Zip - is stored as an integer rather than an object, this removes the leading 0s and, therefore, should be an object.
- Area - should be stored as a category - this field has a limited number of unique values (e.g., rural, urban) and these values represent categories with a predefined hierarchy or ordering, storing it as a category data type is more suitable.
- Children - is stored as a float64 but should be stored as an integer, the need for a fractional child number is not needed here.
- Age - is stored as float64 but should be stored as an integer, the need for a fractional age number is not needed here.
- Education – is stored as an object but should be stored as a category due to the limited number of unique values.

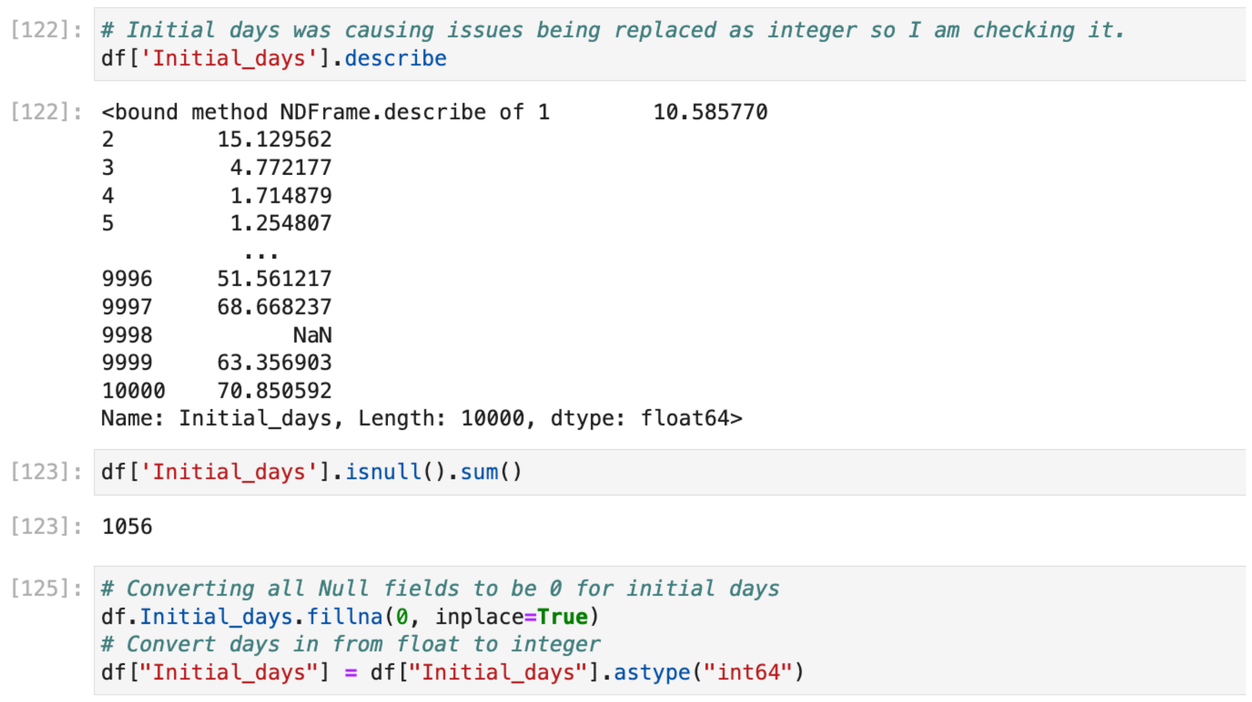
- Marital - is stored as an object but should be stored as a category due to the limited number of unique values.
- Gender - is stored as an object but should be stored as a category due to the limited number of unique values.
- ReAdmins – is stored as an object but should be stored as Boolean, due to it being a yes or no field.
- Soft drink - is stored as an object but should be stored as Boolean, due to it being a yes or no field.
- Initial_admin – is stored as an object but could be stored as a category due to a limited number of unique values.
- High blood – is stored as an object but could be stored as Boolean, due to it being a yes or no field.
- Stroke– is stored as an object but could be stored as Boolean, due to it being a yes or no field.
- Complication_risk – is stored as an object but could be stored as a category, due to the limited number of unique values.
- Overweight– is stored as a float64 could be stored as Boolean, due to it being a yes or no field.
- Arthritis – is stored as an object but could be stored as Boolean, due to it being a yes or no field.
- Diabetes – is stored as an object but could be stored as Boolean, due to it being a yes or no field.
- Hyperlipidemia – is stored as an object but could be stored as Boolean, due to it being a yes or no field.
- Backpain – is stored as an object but could be stored as Boolean, due to it being a yes or no field.
- Anxiety – is stored as a float64 but could be stored as Boolean, due to it being a yes or no field.
- Allergic_rhinitis – is stored as an object but could be stored as Boolean, due to it being a yes or no field.
- Reflux_esophagitis – is stored as an object but could be stored as Boolean, due to it being a yes or no field.
- Asthma – is stored as an object but could be stored as Boolean, due to it being a yes or no field.
- Services – is stored as an object but could be stored as a category, due to it being a yes or no field.
- Item1, item2, item3, item 4, item5, item6, item7, and item8 are all stored as int64, but should all be stored as category, due to the limited number of unique fields.

The last few data cleaning steps performed were to change TotalCharge and AdditionalCharge to two decimal places using the round(2) function. And lastly, to clean up the data set a bit more visually, the header titles could be changed to be more descriptive and cleaner. Using the rename(columns={}) functions, the following titles were changed.

- Customer_id : Customer_ID
- VitD_levels : VitD_Levels

- Doc_visits : Doc_Visits
- Full_meals_eaten : Full_Meals_Eaten
- VitD_supp : VitD_Supplement
- Soft_drink : Soft_Drink
- Initial_admin : Initial_Admin
- Complication_risk : Complication_Risk
- Allergic_rhinitis : Allergic_Rhinitis
- Reflux_esophagitis : Reflux_Esophagitis
- Initial_days : Initial_Days
- Additional_charges : Additional_Charges
- Item1 : Survey_Timely_Admission
- Item2 : Survey_Timely_Treatment
- Item3 : Survey_Timely_Visits
- Item4 : Survey_Reliability
- Item5 : Survey_Options
- Item6 : Survey_Hours_Treatment
- Item7 : Survey_Courteous
- Item8 : Survey_Active_Listening

See all outputted cleaning code below.



```
[122]: # Initial days was causing issues being replaced as integer so I am checking it.
       df['Initial_days'].describe

[122]: <bound method NDFrame.describe of 1        10.585770
       2        15.129562
       3         4.772177
       4         1.714879
       5         1.254807
                  ...
       9996     51.561217
       9997     68.668237
       9998           NaN
       9999     63.356903
       10000    70.850592
       Name: Initial_days, Length: 10000, dtype: float64>

[123]: df['Initial_days'].isnull().sum()

[123]: 1056

[125]: # Converting all Null fields to be 0 for initial days
       df.Initial_days.fillna(0, inplace=True)
       # Convert days in from float to integer
       df["Initial_days"] = df["Initial_days"].astype("int64")
```

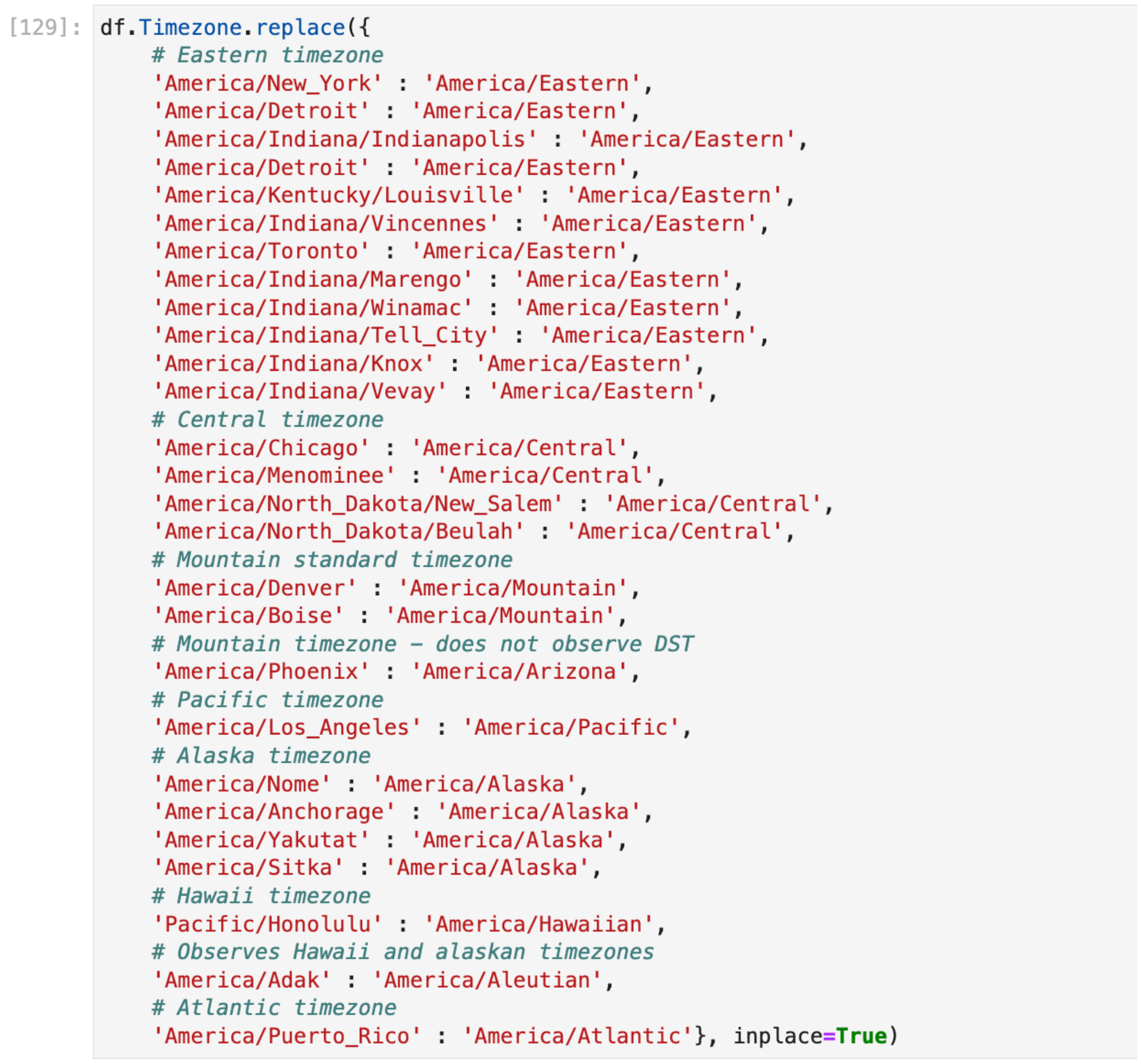
```
[126]: # Update columns to have better capitalization and naming conventions
df = df.rename(columns={'Customer_id': 'Customer_ID'})
df = df.rename(columns={'VitD_levels': 'VitD_Levels'})
df = df.rename(columns={'Doc_visits': 'Doc_Visits'})
df = df.rename(columns={'Full_meals_eaten': 'Full_Meals_Eaten'})
df = df.rename(columns={'VitD_supp': 'VitD_Supplement'})
df = df.rename(columns={'Soft_drink': 'Soft_Drink'})
df = df.rename(columns={'Initial_admin': 'Initial_Admin'})
df = df.rename(columns={'Complication_risk': 'Complication_Risk'})
df = df.rename(columns={'Allergic_rhinitis': 'Allergic_Rhinitis'})
df = df.rename(columns={'Reflux_esophagitis': 'Reflux_Esophagitis'})
df = df.rename(columns={'Initial_days': 'Initial_Days'})
df = df.rename(columns={'Additional_charges': 'Additional_Charges'})
df = df.rename(columns={'Item1': 'Survey_Timely_Admission'})
df = df.rename(columns={'Item2': 'Survey_Timely_Treatment'})
df = df.rename(columns={'Item3': 'Survey_Timely_Visits'})
df = df.rename(columns={'Item4': 'Survey_Reliability'})
df = df.rename(columns={'Item5': 'Survey_Options'})
df = df.rename(columns={'Item6': 'Survey_Hours_Treatment'})
df = df.rename(columns={'Item7': 'Survey_Courteous'})
df = df.rename(columns={'Item8': 'Survey_Active_Listening'})
```

```
[127]: # Re-examining time zones to determine best way to tackle this
df['Timezone'].describe()
```

```
[127]: count                10000
unique                  26
top       America/New_York
freq                  3889
Name: Timezone, dtype: object
```

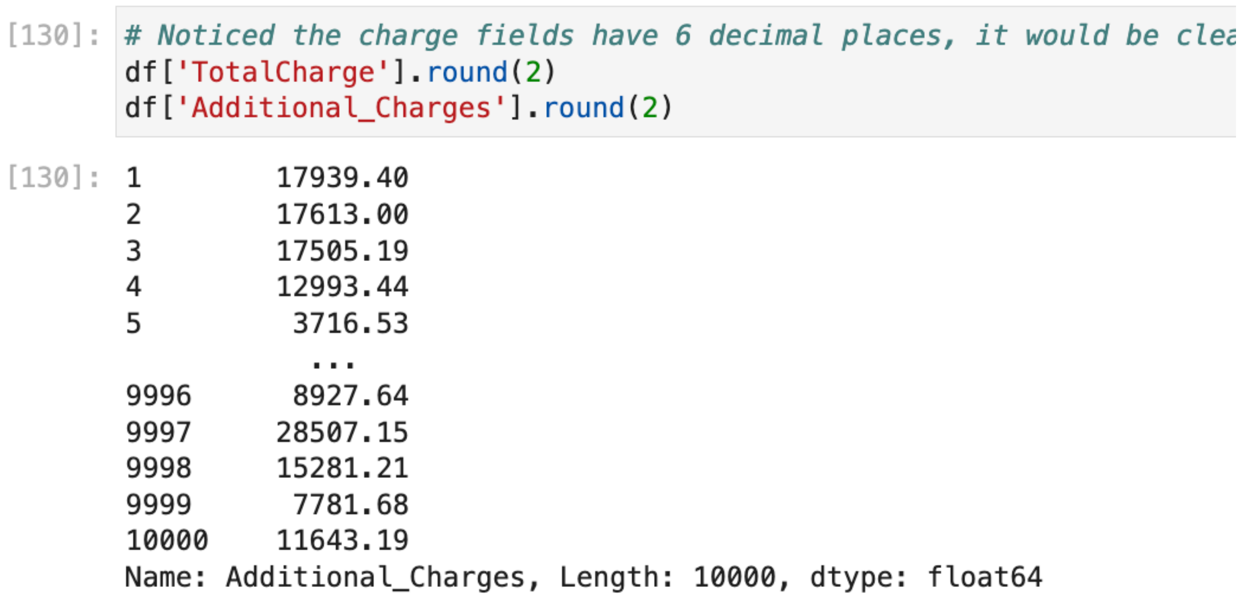
```
[128]: df['Timezone'].unique()
```

```
[128]: array(['America/Chicago', 'America/New_York', 'America/Los_Angeles',
       'America/Indiana/Indianapolis', 'America/Detroit',
       'America/Denver', 'America/Nome', 'America/Anchorage',
       'America/Phoenix', 'America/Boise', 'America/Puerto_Rico',
       'America/Yakutat', 'Pacific/Honolulu', 'America/Menominee',
       'America/Kentucky/Louisville', 'America/Indiana/Vincennes',
       'America/Toronto', 'America/Indiana/Marengo',
       'America/Indiana/Winamac', 'America/Indiana/Tell_City',
       'America/Sitka', 'America/Indiana/Knox',
       'America/North_Dakota/New_Salem', 'America/Indiana/Vevay',
       'America/Adak', 'America/North_Dakota/Beulah'], dtype=object)
```



[129]:

```python
df.Timezone.replace({
    # Eastern timezone
    'America/New_York' : 'America/Eastern',
    'America/Detroit' : 'America/Eastern',
    'America/Indiana/Indianapolis' : 'America/Eastern',
    'America/Detroit' : 'America/Eastern',
    'America/Kentucky/Louisville' : 'America/Eastern',
    'America/Indiana/Vincennes' : 'America/Eastern',
    'America/Toronto' : 'America/Eastern',
    'America/Indiana/Marengo' : 'America/Eastern',
    'America/Indiana/Winamac' : 'America/Eastern',
    'America/Indiana/Tell_City' : 'America/Eastern',
    'America/Indiana/Knox' : 'America/Eastern',
    'America/Indiana/Vevay' : 'America/Eastern',
    # Central timezone
    'America/Chicago' : 'America/Central',
    'America/Menominee' : 'America/Central',
    'America/North_Dakota/New_Salem' : 'America/Central',
    'America/North_Dakota/Beulah' : 'America/Central',
    # Mountain standard timezone
    'America/Denver' : 'America/Mountain',
    'America/Boise' : 'America/Mountain',
    # Mountain timezone - does not observe DST
    'America/Phoenix' : 'America/Arizona',
    # Pacific timezone
    'America/Los_Angeles' : 'America/Pacific',
    # Alaska timezone
    'America/Nome' : 'America/Alaska',
    'America/Anchorage' : 'America/Alaska',
    'America/Yakutat' : 'America/Alaska',
    'America/Sitka' : 'America/Alaska',
    # Hawaii timezone
    'Pacific/Honolulu' : 'America/Hawaiian',
    # Observes Hawaii and alaskan timezones
    'America/Adak' : 'America/Aleutian',
    # Atlantic timezone
    'America/Puerto_Rico' : 'America/Atlantic'}, inplace=True)
```



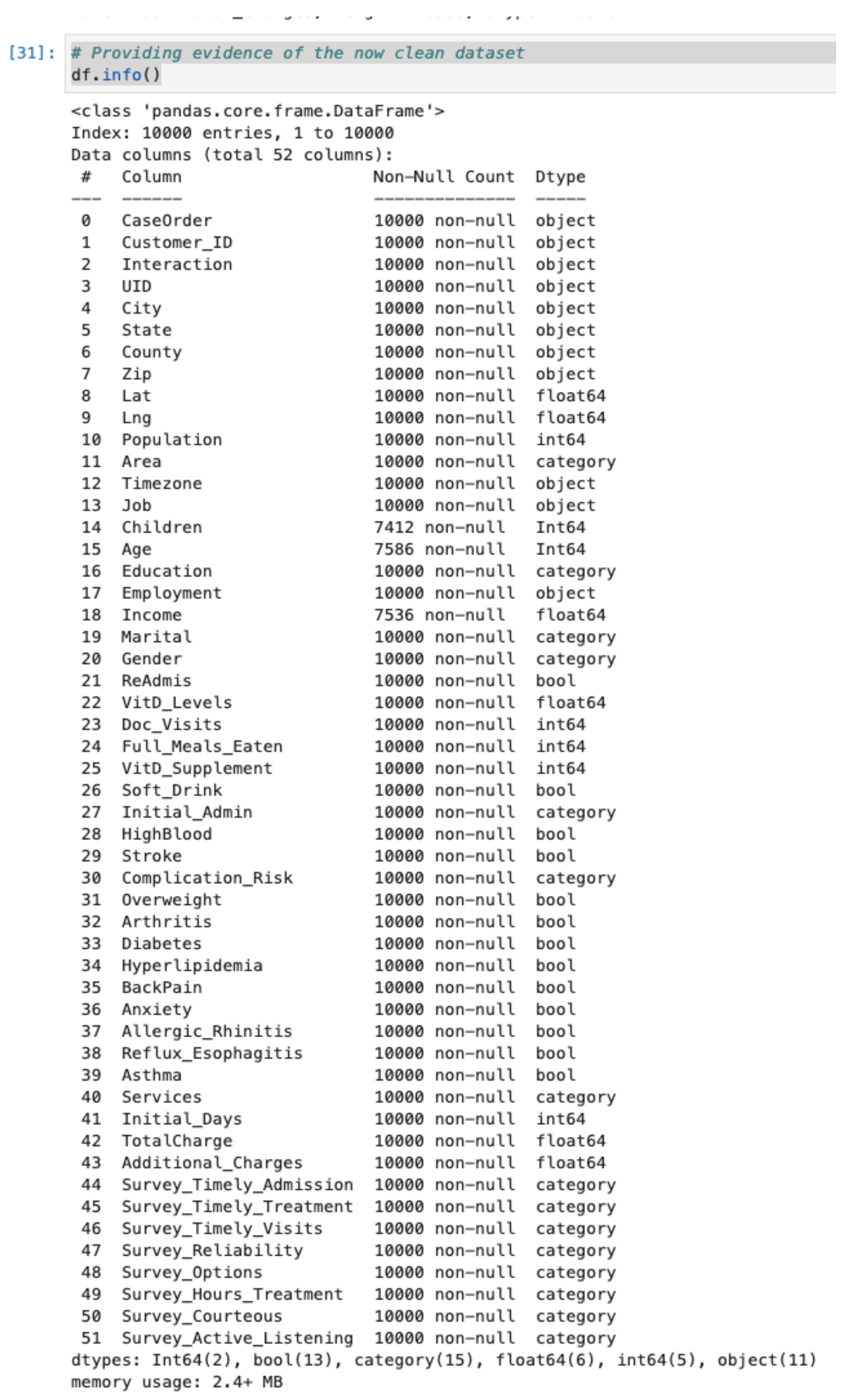
```
[130]: # Noticed the charge fields have 6 decimal places, it would be clea
       df['TotalCharge'].round(2)
       df['Additional_Charges'].round(2)

[130]: 1        17939.40
       2        17613.00
       3        17505.19
       4        12993.44
       5         3716.53
                   ...
       9996      8927.64
       9997     28507.15
       9998     15281.21
       9999      7781.68
       10000    11643.19
       Name: Additional_Charges, Length: 10000, dtype: float64
```

D3, Data Cleaning Summary

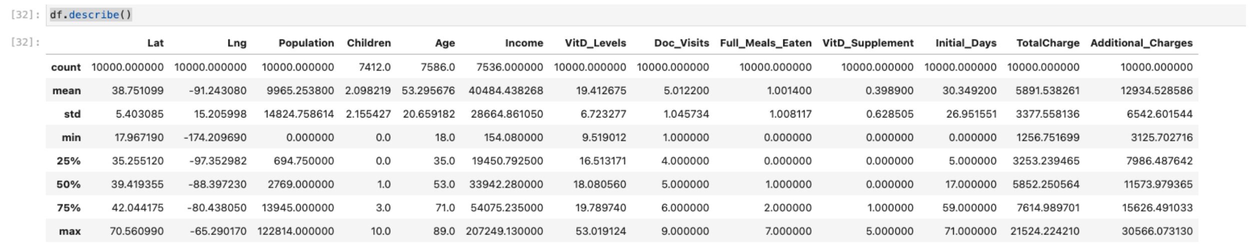
The process to clean the dataset for analysis first started with reviewing the data dictionary and rubric to help determine a plan for cleaning. The plan consisted of first identifying all duplicates, missing values, outliers, and the re-expression of categorical variables. Once the entire dataset had been reviewed and assessed, next came the steps to clean the various components found that needed cleaning. From the analysis, there were no duplicates found that needed to be corrected. For the variables that contained nulls, all except for initial days were determined that it was best for these to remain the same due to the chance that changing the values to 0 would directly impact any assessment done on the dataset. For the re-expression of variables, it should be noted that there is a level of subjectivity and bias needed when cleaning data. Specifically, when deciding to change the field names, personal judgment and experience are the best approaches to follow when trying to decide on what to change field names to. After all the functions to assess the variables in the dataset, as well as taking steps towards enhancing these, the dataset now is more visual. It contains only key fields and values that could be used in a variety of ways to improve the organization's processes.

The now cleaned dataset contains the correct data types needed for that specific field, unique names related to the values of the field, and appears to be cleaned efficiently. An output of the cleaned dataset is provided in screenshots below as well as linked in the sections below.



```
[31]: # Providing evidence of the now clean dataset
      df.info()

      <class 'pandas.core.frame.DataFrame'>
      Index: 10000 entries, 1 to 10000
      Data columns (total 52 columns):
       #   Column                   Non-Null Count  Dtype
      ---  ------                   --------------  -----
       0   CaseOrder                10000 non-null  object
       1   Customer_ID              10000 non-null  object
       2   Interaction              10000 non-null  object
       3   UID                      10000 non-null  object
       4   City                     10000 non-null  object
       5   State                    10000 non-null  object
       6   County                   10000 non-null  object
       7   Zip                      10000 non-null  object
       8   Lat                      10000 non-null  float64
       9   Lng                      10000 non-null  float64
       10  Population               10000 non-null  int64
       11  Area                     10000 non-null  category
       12  Timezone                 10000 non-null  object
       13  Job                      10000 non-null  object
       14  Children                 7412 non-null   Int64
       15  Age                      7586 non-null   Int64
       16  Education                10000 non-null  category
       17  Employment               10000 non-null  object
       18  Income                   7536 non-null   float64
       19  Marital                  10000 non-null  category
       20  Gender                   10000 non-null  category
       21  ReAdmis                  10000 non-null  bool
       22  VitD_Levels              10000 non-null  float64
       23  Doc_Visits               10000 non-null  int64
       24  Full_Meals_Eaten         10000 non-null  int64
       25  VitD_Supplement          10000 non-null  int64
       26  Soft_Drink               10000 non-null  bool
       27  Initial_Admin            10000 non-null  category
       28  HighBlood                10000 non-null  bool
       29  Stroke                   10000 non-null  bool
       30  Complication_Risk        10000 non-null  category
       31  Overweight               10000 non-null  bool
       32  Arthritis                10000 non-null  bool
       33  Diabetes                 10000 non-null  bool
       34  Hyperlipidemia           10000 non-null  bool
       35  BackPain                 10000 non-null  bool
       36  Anxiety                  10000 non-null  bool
       37  Allergic_Rhinitis        10000 non-null  bool
       38  Reflux_Esophagitis       10000 non-null  bool
       39  Asthma                   10000 non-null  bool
       40  Services                 10000 non-null  category
       41  Initial_Days             10000 non-null  int64
       42  TotalCharge              10000 non-null  float64
       43  Additional_Charges       10000 non-null  float64
       44  Survey_Timely_Admission  10000 non-null  category
       45  Survey_Timely_Treatment  10000 non-null  category
       46  Survey_Timely_Visits     10000 non-null  category
       47  Survey_Reliability       10000 non-null  category
       48  Survey_Options           10000 non-null  category
       49  Survey_Hours_Treatment   10000 non-null  category
       50  Survey_Courteous         10000 non-null  category
       51  Survey_Active_Listening  10000 non-null  category
      dtypes: Int64(2), bool(13), category(15), float64(6), int64(5), object(11)
      memory usage: 2.4+ MB
```



```
[32]: df.describe()
```

[32]:

	Lat	Lng	Population	Children	Age	Income	VitD_Levels	Doc_Visits	Full_Meals_Eaten	VitD_Supplement	Initial_Days	TotalCharge	Additional_Charges
count	10000.000000	10000.000000	10000.000000	7412.0	7586.0	7536.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	38.751099	-91.243080	9965.253800	2.098219	53.295676	40484.438268	19.412675	5.012200	1.001400	0.398900	30.349200	5891.538261	12934.528586
std	5.403085	15.205998	14824.758614	2.155427	20.659182	28664.861050	6.723277	1.045734	1.008117	0.628505	26.951551	3377.558136	6542.601544
min	17.967190	-174.209690	0.000000	0.0	18.0	154.080000	9.519012	1.000000	0.000000	0.000000	0.000000	1256.751699	3125.702716
25%	35.255120	-97.352982	694.750000	0.0	35.0	19450.792500	16.513171	4.000000	0.000000	0.000000	5.000000	3253.239465	7986.487642
50%	39.419355	-88.397230	2769.000000	1.0	53.0	33942.280000	18.080560	5.000000	1.000000	0.000000	17.000000	5852.250564	11573.979365
75%	42.044175	-80.438050	13945.000000	3.0	71.0	54075.235000	19.789740	6.000000	2.000000	1.000000	59.000000	7614.989701	15626.491033
max	70.560990	-65.290170	122814.000000	10.0	89.0	207249.130000	53.019124	9.000000	7.000000	5.000000	71.000000	21524.224210	30566.073130

D4, Data Anomalies and Issues

See the code attached 'D206 – D2 – Cleaning Code.ipynb' for the detection code used.

D5, CSV File

See the cleaned data file attached 'D206 – Medical Dataset - Clean.ipynb'.

D6, Data Cleaning Limitations

Some limitations commonly found when detecting data for cleaning are that Python libraries offer methods for handling missing data, but sometimes these methods might not be suitable. Certain techniques such as mean, or median detection could introduce bias. Additionally, data often contains errors, inconsistencies, or inaccuracies that are not easily detected or corrected. There is a level of manual effort needed to be done to review the dataset, this manual effort depending on the size of the dataset is very time-consuming.

Some limitations found when treating the dirty data discussed in D2 is that the process of removing null records, making manual adjustments, as well as relabeling a record; could have led to the loss of valuable information. A key example is when adjusting all null values in initial days to be 0, even though I exercised personal judgment as well as a review of the data dictionary which I do believe to be correct, an error in this assessment could fundamentally skew results. Additionally, using bias or personal opinions when deciding on how to clean data, could lead to bad assumptions about the dataset and nature of the goals of the analysis. Thus, is a disadvantage of the methods used to clean data.

D7, Limitations Summarized

Challenges that a data analyst may encounter after using the now-cleaned data for analysis would mostly fall in the areas that were updated based on personal judgment. Due to a lack of specialty knowledge and using best judgment, the process to rename the various column headers, and adjust the total charge and additional charge columns to be 2 decimals, was done from personal best judgment. This assumption was applied without knowing how this data flows through the business or if it will affect how the data comes into this dataset. This could cause a great deal of additional work for the analysts when attempting to use this data or ensure continued support and enhancements made to it.

Another assumption made on the dataset was that the initial day's null values could be set to 0, this was based on a review of the data dictionary and look at the table. However, a lack of domain knowledge could lead this to cause issues for an analyst. Regarding the datatype changes made, if any of these values were incorrectly updated or incorrectly assessed based on limited facts, this could greatly mess with any analysis and prevent the input of desired fields.

In summary, any objective or biased changes made to data could affect the way another analyst views the data. Additionally, any incorrect assumptions made initially that lead to the reduction or treatment of certain values could greatly skew the results of a future analysis before by a different analyst.

E, Principal Component Analysis

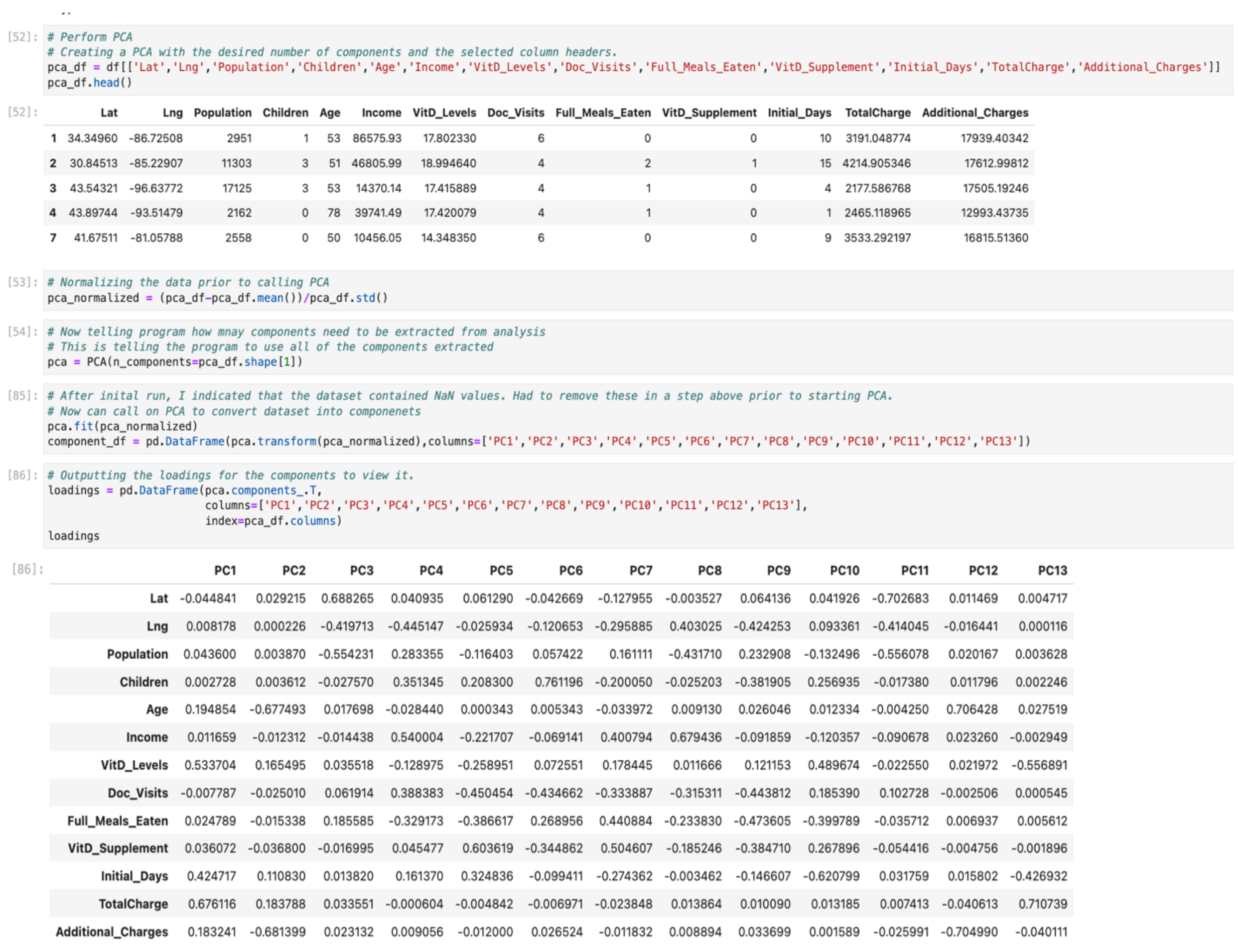
Principal Component Analysis (PCA) is a dimensionality reduction technique commonly used in machine learning and data analysis to reduce the number of features in a dataset while preserving the most important information. In Python, you can perform PCA using the PCA class from the sklearn.decomposition module in scikit-learn. See the summary below for the PCA method and functions used in this analysis.

E1, PCA Details

PCA requires that all input variables are quantifiable numeric values that do not contain null values, thus, all quantitative variables were chosen for the PCA, and all null values were removed (using the function dropna(inplace=True)) before inputting the variables into the matrix. The dataset was reduced in size by about 58% after removing all the null values. This code is contained in this section rather than under the data cleaning section above due to it being specific to the PCA performed. Without the PCA these variables most likely would not have been removed as part of the normal data cleaning process.

The following 13 variables were used in the analysis. A screenshot of the PCA loadings matrix is provided below the list of variables.

- Lat
- Lng
- Population
- Children
- Age
- Income
- VitD_Levels
- Doc_Visits
- Full_Meals_Eaten
- VitD_Supplement
- Initial_Days
- TotalCharge
- Additional_Charges.



```
[52]: # Perform PCA
      # Creating a PCA with the desired number of components and the selected column headers.
      pca_df = df[['Lat','Lng','Population','Children','Age','Income','VitD_Levels','Doc_Visits','Full_Meals_Eaten','VitD_Supplement','Initial_Days','TotalCharge','Additional_Charges']]
      pca_df.head()
```

[52]:

	Lat	Lng	Population	Children	Age	Income	VitD_Levels	Doc_Visits	Full_Meals_Eaten	VitD_Supplement	Initial_Days	TotalCharge	Additional_Charges
1	34.34960	-86.72508	2951	1	53	86575.93	17.802330	6	0	0	10	3191.048774	17939.40342
2	30.84513	-85.22907	11303	3	51	46805.99	18.994640	4	2	1	15	4214.905346	17612.99812
3	43.54321	-96.63772	17125	3	53	14370.14	17.415889	4	1	0	4	2177.586768	17505.19246
4	43.89744	-93.51479	2162	0	78	39741.49	17.420079	4	1	0	1	2465.118965	12993.43735
7	41.67511	-81.05788	2558	0	50	10456.05	14.348350	6	0	0	9	3533.292197	16815.51360

```
[53]: # Normalizing the data prior to calling PCA
      pca_normalized = (pca_df-pca_df.mean())/pca_df.std()
```

```
[54]: # Now telling program how mnay components need to be extracted from analysis
      # This is telling the program to use all of the components extracted
      pca = PCA(n_components=pca_df.shape[1])
```

```
[85]: # After inital run, I indicated that the dataset contained NaN values. Had to remove these in a step above prior to starting PCA.
      # Now can call on PCA to convert dataset into componenets
      pca.fit(pca_normalized)
      component_df = pd.DataFrame(pca.transform(pca_normalized),columns=['PC1','PC2','PC3','PC4','PC5','PC6','PC7','PC8','PC9','PC10','PC11','PC12','PC13'])
```

```
[86]: # Outputting the loadings for the components to view it.
      loadings = pd.DataFrame(pca.components_.T,
                              columns=['PC1','PC2','PC3','PC4','PC5','PC6','PC7','PC8','PC9','PC10','PC11','PC12','PC13'],
                              index=pca_df.columns)
      loadings
```

[86]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13
Lat	-0.044841	0.029215	0.688265	0.040935	0.061290	-0.042669	-0.127955	-0.003527	0.064136	0.041926	-0.702683	0.011469	0.004717
Lng	0.008178	0.000226	-0.419713	-0.445147	-0.025934	-0.120653	-0.295885	0.403025	-0.424253	0.093361	-0.414045	-0.016441	0.000116
Population	0.043600	0.003870	-0.554231	0.283355	-0.116403	0.057422	0.161111	-0.431710	0.232908	-0.132496	-0.556078	0.020167	0.003628
Children	0.002728	0.003612	-0.027570	0.351345	0.208300	0.761196	-0.200050	-0.025203	-0.381905	0.256935	-0.017380	0.011796	0.002246
Age	0.194854	-0.677493	0.017698	-0.028440	0.000343	0.005343	-0.033972	0.009130	0.026046	0.012334	-0.004250	0.706428	0.027519
Income	0.011659	-0.012312	-0.014438	0.540004	-0.221707	-0.069141	0.400794	0.679436	-0.091859	-0.120357	-0.090678	0.023260	-0.002949
VitD_Levels	0.533704	0.165495	0.035518	-0.128975	-0.258951	0.072551	0.178445	0.011666	0.121153	0.489674	-0.022550	0.021972	-0.556891
Doc_Visits	-0.007787	-0.025010	0.061914	0.388383	-0.450454	-0.434662	-0.333887	-0.315311	-0.443812	0.185390	0.102728	-0.002506	0.000545
Full_Meals_Eaten	0.024789	-0.015338	0.185585	-0.329173	-0.386617	0.268956	0.440884	-0.233830	-0.473605	-0.399789	-0.035712	0.006937	0.005612
VitD_Supplement	0.036072	-0.036800	-0.016995	0.045477	0.603619	-0.344862	0.504607	-0.185246	-0.384710	0.267896	-0.054416	-0.004756	-0.001896
Initial_Days	0.424717	0.110830	0.013820	0.161370	0.324836	-0.099411	-0.274362	-0.003462	-0.146607	-0.620799	0.031759	0.015802	-0.426932
TotalCharge	0.676116	0.183788	0.033551	-0.000604	-0.004842	-0.006971	-0.023848	0.013864	0.010090	0.013185	0.007413	-0.040613	0.710739
Additional_Charges	0.183241	-0.681399	0.023132	0.009056	-0.012000	0.026524	-0.011832	0.008894	0.033699	0.001589	-0.025991	-0.704990	-0.040111

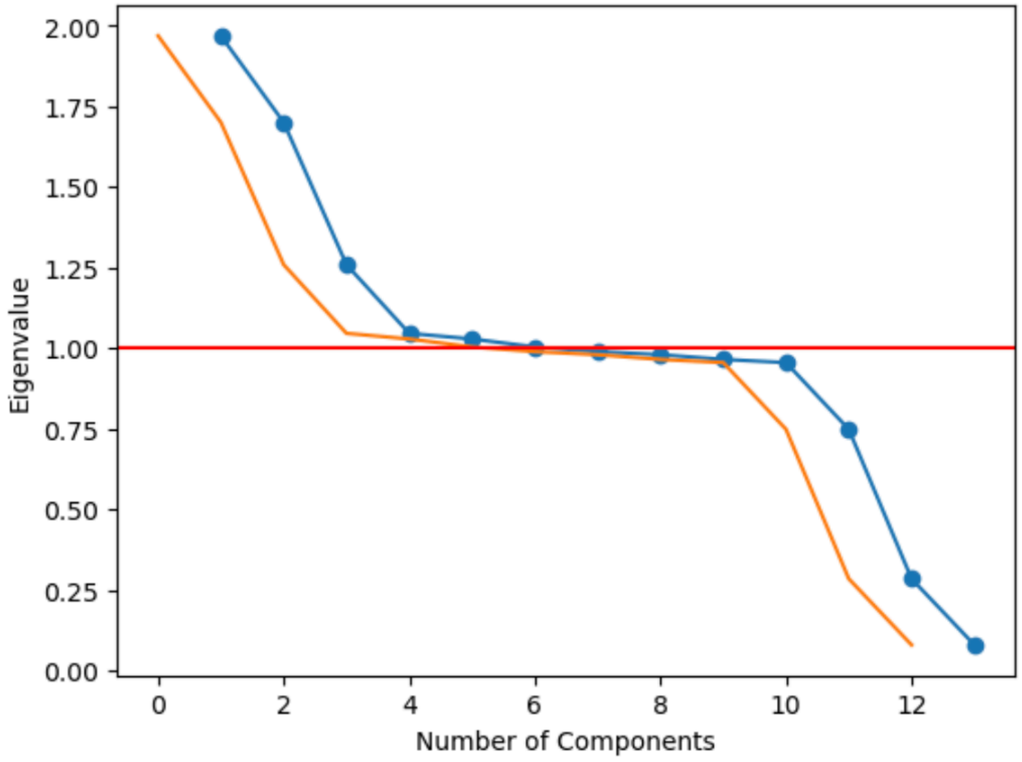
E2, PCA Plotted

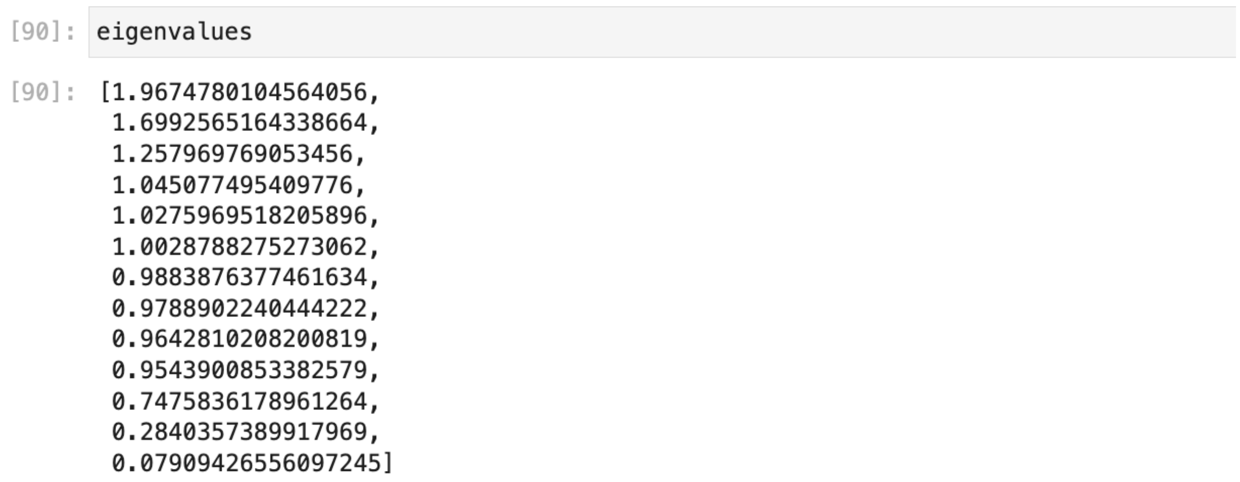
In PCA, eigenvalues play an important role in determining the principal components and their importance. In Python, the NumPy and scikit-learn libraries are utilized to calculate eigenvalues as part of the PCA process. To determine the values that should be retained, the Kaiser rule can be applied using eigenvalues. The Kaiser rule states that you want to keep all principal components with an eigenvalue equal to or greater than 1. Eigenvalues represent the amount of variance explained by each principal component. Each principal component with an eigenvalue equal to or greater than 1 shows the magnitude of the variance captured by that component. Higher values correspond to components that have more variance in the data. Total variance in the data is the sum of all eigenvalues, thus, this approach helps to provide a measure of overall variance in the dataset.

The steps taken to determine the variance that should be retained, started with determining the quantitative variables and creating the PCA matrix. Next, you can extract the eigenvalues from the PCA using specific code built into the Python libraries and output visuals to get a representation of the values. As noted in the rubric for this course, the PCs represent the combination of variables that are related, based on the PCA loadings matrix. See eigenvalues and scree plot displayed below. Note that the axis on the plat was adjusted so that the axis starts at 1 rather than the auto-selected 0.

```
[88]: # Extracting the eigenvalues
      matrix = np.dot(pca_normalized.T, pca_normalized) / pca_df.shape[0]
      eigenvalues = [np.dot(eigenvector.T, np.dot(matrix, eigenvector)) for
                     eigenvector in pca.components_]
```

```
[89]: # Running code to view eigenvalues
      # Adjusting the plot so that the axis starts at 1 rather then auto selected 0.
      plt.plot(np.arange(1,len(eigenvalues)+1),eigenvalues, marker='o')
      plt.plot(eigenvalues)
      plt.xlabel('Number of Components')
      plt.ylabel('Eigenvalue')
      plt.axhline(y=1, color='red')
      plt.show()
```





```
[90]: eigenvalues

[90]: [1.9674780104564056,
       1.6992565164338664,
       1.257969769053456,
       1.045077495409776,
       1.0275969518205896,
       1.0028788275273062,
       0.9883876377461634,
       0.9788902240444222,
       0.9642810208200819,
       0.9543900853382579,
       0.7475836178961264,
       0.2840357389917969,
       0.07909426556097245]
```

The scree plot shown above clearly shows that the first 6 principal components have eigenvalues greater than 1, you can see the number of variables all plotted above the red line. Additionally, printing out the list of eigenvalues shows you the clear value associated with each one. These are the variables we will be retaining. As indicated, the above information tells us that the first 6 principal components have an eigenvalue greater than 1, thus, these should all be retained. Components 7 through 13 all have eigenvalues below 1, so they will be removed.

E3, PCA Benefits

PCA has a variety of benefits associated with it. PCA is a dimensionality reduction framework, in this analysis PCA assisted in breaking down the data to a granular level and identifying the key quantitative dimensions. The PCA framework is often used by data scientists inside of an organization due to the dimensionality of the organization's data being worked with. When using PCA in an analysis a data scientist can produce improved details about the data inside of an organization. PCAs help organizations identify important variables, enable the use of visualizations, reduce data redundancy, assist in feature engineering projects, improve model performance, and help compress data. All of these benefits help organizations gain insights into their data, as well as improve efficiency and build better models for decision-making or problem-solving tasks.

In this dataset, the PCAs were determined using the techniques discussed in the above sections. PCA combines the variables that have relationships with each other. This process could be very important and useful for an organization looking to identify trends and correlations in the data. PCA in this case takes the "like" quantitative variables and groups them to determine the relationship. In the loadings output printed above, when you review the loadings of each variable on these principal components, variables with high loadings (absolute values) on a particular PC are strongly associated with that component. For example, under PC1 totalcharge and initial_days are a few of the highest values associated with the PC indicating that the days a patient was in the hospital are highly correlated with the amount that a patient was charged. This type of information can greatly help an organization make decisions about various parts of the business. In this case, the organization can infer that patients will tend to pay more as they stay longer and the longer they stay is based on how healthy or old they are at admission. In this type of analysis, organizations can test much fewer variables against each other rather than every single known variable.

F, Panopto Video Recording Executions

The video recording for this assignment includes a vocalized demonstration of all the code, the code being executed, and the results of the code being represented inside this report.

The video recording for this project can be found inside the Panopto drop box titled “

Panopto video link: https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=cae660d7-0974-4629-950a-b1410161d424

G, Web Sources

No sources, or segments of third-party code, were used to acquire data or to support the report.

H, Acknowledge Sources

I acknowledge that no sources, or segments of third-party sources, were stated or copied from the web into this report.