Leveraging Data Preparation and Cleaning

When Analyzing Hospital Readmissions

Jason Willis

College of Information Technology, Western Governors University

Dr. Emelda Ntinglet

June 24, 2021

Table of Contents For Each Rubric

Hospital Readmission Problem

30-day hospital readmissions have some level of preventability, with estimates ranging between 15 – 45%, Jacqueline Pugh (*Evidence-Based Processes to Prevent Readmissions: More is Better, a Ten-Site Observational Study | BMC Health Services Research*, 2021). Care transmission processes reduce readmissions in clinical trials. Which combination of techniques provides the lowest readmissions for patients with certain patient risk factors?

Within the dataset I have prepared, there could be a potential variable of interest that may correlate positively and/or negatively with patient readmission outcomes. For instance, I'll be cleaning data describing a patients' level of complication risk. Additionally, I've also noticed the initial number of days a patient stayed in the hospital during the initial visit. Could these data fields intrinsically provide insight; are their synergistic combinations that may correlate to increased complications?

Research Question

Can patient complication risk levels help predict the potential for patient readmission? Could the length of stay provide any reduction of patient readmission within these marked risk levels?

Data Cleaning Plan

My first approach was to manually explore the raw medical data in its original comma-separated value (CSV) format. Within this view, I noticed series naming conventions seemed inconsistent, as some had special characters, some used Pascal Case, and others used underscores between words. Field variables were rewritten to improve consistency; updated to a hybrid of Pascal Case and Python snake_case format. Each first letter was capitalized, and spaces between words are represented with an underscore.

Required Variables

I created a data dictionary from the raw medical data to help describe the data, their types, null count, and additional comments. Below is a table describing some of the data:

Table 1 - Data Dictionary for medical_raw_data.csv

	Information	Non-Null Count	Data Type	Type of Data	Example	Comments
Unnamed: 0		10000	int64	Discrete	1	Redundant, CaseOrder also counts; delete
CaseOrder	A placeholder variable to preserve the original order of the raw data file	10000	int64	Discrete	1	Inconsistent var name
Customer_id	Unique Patient ID	10000	object	Categorical	C412403	
Interaction	Unique IDs related to patient transactions, procedures, and admissions	10000	object	Categorical	8cd49b13-f45a-4b47-a2bd-173ffa932c2f	
UID		10000	object	Categorical	3a83ddb66e2ae73798bdf1d705dc0932	
City	Patient's city of residence as listed on the billing statement	10000	object	Categorical	Eva	
State	Patient's state of residence as listed on the billing statement	10000	object	Categorical	AL	
County	Patient's county of residence as listed on the billing statement	10000	object	Categorical	Morgan	
Zip	Patient's zip of residence as listed on the billing statement	10000	int64	Categorical	35621	
Lat	GPS coordinates of patient residence as listed on the billing statement	10000	float64	Continuous	34.3496	
Lng		10000	float64	Continuous	-86.72508	
Population	Pop within a mile radius of patient, based on census data	10000	int64	Discrete	2951	
Area	Area type (rural, urban, suburban), based on unofficial census data	10000	object	Categorical	Suburban	
Timezone	Time zone of patient residence based on patient's sign-up information	10000	object	Categorical	America/Chicago	
Job	Job of the patient (or primary insurance holder) as reported in the admissions information	10000	object	Categorical	Psychologist, sport and exercise	
Children	Number of children in the patient's household as reported in the admissions information	7412	float64	Discrete	1	Missing Data. *Note: Only objects & floats can have nulls in pandas.*
Age	Age of the patient as reported in admissions information	7586	float64	Discrete	53	Missing Data. *Note: Only objects & floats can have nulls in pandas.*

	Information	Non-Null Count	Data Type	Type of Data	Example	Comments
Education	Highest earned degree of patient as reported in admissions information	10000	object	Categorical	Some College, Less than 1 Year	
Employment	Employment status of patient as reported in admissions information	10000	object	Categorical	Full Time	
Income	Annual income of the patient (or primary insurance holder) as reported at time of admission	7536	float64	Continuous	86575.93	Missing Data.
Marital	Marital status of the patient (or primary insurance holder) as reported on admission information	10000	object	Categorical	Divorced	
Gender	Customer self-identification as male, female, or nonbinary	10000	object	Categorical	Male	
ReAdmis	Whether the patient was readmitted within a month of release or not (yes, no)	10000	object	Categorical	No	Central to this study; *Note: Only objects & floats can have nulls in pandas.*
VitD_levels	The patient's vitamin D levels as measured in ng/mL	10000	float64	Continuous	17.80233049	
Doc_visits	Number of times the primary physician visited the patient during the initial hospitalization	10000	int64	Discrete	6	
Full_meals_eaten	Number of full meals the patient ate while hospitalized (partial meals count as 0, and some patients had more than three meals in a day if requested)	10000	int64	Discrete	0	
VitD_supp	The number of times that vitamin D supplements were administered to the patient	10000	int64	Discrete	0	
Soft_drink	Whether the patient habitually drinks three or more sodas in a day (yes, no)	7533	object	Categorical	NA	Missing values.
Initial_admin	The means by which the patient was admitted into the hospital initially (emergency admission, elective admission, observation)	10000	object	Categorical	Emergency Admission	
HighBlood	Whether the patient has high blood pressure (yes, no)	10000	object	Categorical	Yes	Inconsistent var name *Note: Only objects & floats can have nulls in pandas.*
Stroke	Whether the patient has had a stroke (yes, no)	10000	object	Categorical	No	*Note: Only objects & floats can have nulls in pandas.*
Complication_risk	Level of complication risk for the patient as assessed by primary patient assessment (high, med, low)	10000	object	Categorical	Medium	Data should help provide insight for my research question

	Information	Non-Null Count	Data Type	Type of Data	Example	Comments
Overweight	Whether the patient is considered overweight based on age, gender, and height (yes, no)	9018	object	Categorical	0	Missing Values. *Note: Only objects & floats can have nulls in pandas.*
Arthritis	Whether the patient has arthritis (yes, no)	10000	object	Categorical	Yes	*Note: Only objects & floats can have nulls in pandas.*
Diabetes	Whether the patient has diabetes (yes, no)	10000	object	Categorical	Yes	*Note: Only objects & floats can have nulls in pandas.*
Hyperlipidemia	Whether the patient has hyperlipidemia (yes, no)	10000	object	Categorical	No	*Note: Only objects & floats can have nulls in pandas.*
BackPain	Whether the patient has chronic back pain (yes, no)	10000	object	Categorical	Yes	Inconsistent var name. *Note: Only objects & floats can have nulls in pandas.*
Anxiety	Whether the patient has an anxiety disorder (yes, no)	9016	object	Categorical	1	Missing Values, *Note: Only objects & floats can have nulls in pandas.*
Allergic_ rhinitis	Whether the patient has allergic rhinitis (yes, no)	10000	object	Categorical	Yes	*Note: Only objects & floats can have nulls in pandas.*
Reflux_ esophagitis	Whether the patient has reflux esophagitis (yes, no)	10000	object	Categorical	No	*Note: Only objects & floats can have nulls in pandas.*
Asthma	Whether the patient has asthma (yes, no)	10000	object	Discrete	Yes	*Note: Only objects & floats can have nulls in pandas.*
Services	Primary service the patient received while hospitalized (blood work, intravenous, CT scan, MRI)	10000	object	Categorical	Blood Work	
Initial_days	**The number of days the patient stayed in the hospital during the initial visit**	8944	float64	Continuous	10.58576971	Missing Values. Data should help provide insight for my research question
TotalCharge	The amount charged to the patient daily. This value reflects an average per patient based on the total charge divided by the number of days hospitalized. This amount reflects the typical charges billed to patients, not including specialized treatments.	10000	float64	Continuous	3191.048774	Inconsistent var name
Additional_ charges	The average amount charged to the patient for miscellaneous procedures, treatments, medicines, anesthesiology, etc.	10000	float64	Continuous	17939.40342	
Items Below:	The following variables represent eight-question survey responses; the importance of various factors/surfaces on a scale of 1 to 8 (1 = most important, 8 = least important)					
Item1	Timely admission	10000	int64	Discrete	3	
Item2	Timely treatment	10000	int64	Discrete	3	
Item3	Timely visits	10000	int64	Discrete	2	
Item4	Reliability	10000	int64	Discrete	2	
Item5	Options	10000	int64	Discrete	4	
Item6	Hours of Treatment	10000	int64	Discrete	3	
Item7	Courteous staff	10000	int64	Discrete	3	
Item8	Evidence of active listening from Dr	10000	int64	Discrete	4	

Plan to Find Anomalies

In addition to manually exploring the data, I plan to leverage Python with various libraries to detect and inspect the data. I'll be looking for missing data, reviewing for possible removal, interpolation, or to leave in its original format. Next, I'll be scouring the data set for unwanted data, e.g., outliers and duplicate values, with some boxplots and python methods. Finally, I'll look at the data types to see if any field needs to be converted to a more appropriate format and setting the dataset index.

Justification of Approach

I felt the workflow to use Python (to include Pandas, NumPy, Matplotlib, and SciPy libraries) inside JupyterLab was more intuitive than my experience with R in the course. Most likely, this is most likely due to my preference of comfort when choosing code over what felt like a very statistical syntax in R. Additionally, when following the instruction of Michele Vallisneri (*Python Statistics Essential Training: Data Cleaning | LinkedIn Learning*, 2018), Reindert-Jan Ekker (*Pandas Playbook: Manipulating Data, Pluralsight*, 2021*), and Larose et al. (Data Science using Python and R,* 2020), I found that combining their research and cleaning approaches seemed to feel more natural to me.

Justification of Tools

While I want to learn both Python and R, I chose Python for this assessment. As a general-purpose programming language, Python's should be syntactically similar to other languages I've programmed in and overall is more popular than R, which should help when I'm searching for solutions. According to Reindert-Jan Ekker (*Pandas Playbook: Manipulating Data | Pluralsight*, 2021), Python, and especially the Pandas library, is "the most popular Python framework for doing data science and analysis…you simply cannot go without Pandas anymore".

Provide the Code

Saved within the provided folder is my 'JWillis_D206DataCleaning.ipynb' file, demonstrating the steps taken to clean the medical_raw_data.csv file. Once you have either Jupyter Notebook or JupyterLab running, open JWillis_D206DataCleaning.ipynb. The first step is to import libraries and read the raw data from the CSV. Once the libraries are imported, it's time to read the data. Ensure to either save the medical_raw_data.csv file in the same folder (shown below in Figure 2) or provide the correct path when reading in the data.

Figure 1 Import libraries and read in raw medical data.

Once the data is read into JunyperLab, I started to explore within the data to understand the shape (10,000 rows and 53 columns) and information, i.e., data type per column, nulls, the type of index, and memory usage to keep the data in RAM. To do this, I used the shape property and .info() method, seen in Figure 3.

Exploration

```
[24]: med_df.shape
```

```
[24]: (10000, 53)
```

```
[25]: med_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 53 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Unnamed: 0        10000 non-null  int64
 1   CaseOrder         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  object
 13  Timezone          10000 non-null  object
 14  Job               10000 non-null  object
 15  Children          7412 non-null   float64
 16  Age               7586 non-null   float64
 17  Education         10000 non-null  object
 18  Employment        10000 non-null  object
 19  Income            7536 non-null   float64
 20  Marital           10000 non-null  object
```

Figure 2 - Initial data exploration using the shape property and info() method.

As seen in Figure 3, some "Non-Null" columns contain incomplete series, which are also listed in the data dictionary for convenient reference. Digging into nulls further, I confirmed the following series: Children, Age, Income, Soft_drink, Overweight, Anxiety, and Initial_days all have at least one null value. By running the .isnull().any() method on the data frame, a value of "True" was given on any column containing a null value.

List Only Columns Containing At Least One Null

```
med_df.loc[:, med_df.isnull().any()]
```

	Children	Age	Income	Soft_drink	Overweight	Anxiety	Initial_days
0	1.0	53.0	86575.93	NaN	0.0	1.0	10.585770
1	3.0	51.0	46805.99	No	1.0	NaN	15.129562
2	3.0	53.0	14370.14	No	1.0	NaN	4.772177
3	0.0	78.0	39741.49	No	0.0	NaN	1.714879
4	NaN	22.0	1209.56	Yes	0.0	0.0	1.254807
...	...	...	...	...	...	...	...
9995	NaN	25.0	45967.61	No	NaN	1.0	51.561217
9996	4.0	87.0	14983.02	No	1.0	0.0	68.668237
9997	3.0	NaN	65917.81	Yes	1.0	1.0	NaN
9998	3.0	43.0	29702.32	No	1.0	0.0	63.356903
9999	8.0	NaN	62682.63	No	1.0	0.0	70.850592

10000 rows × 7 columns

Figure 3 - Listing Columns Containing At Least One Null Value

Additionally, to figure out if any row has all nulls (which would be a good candidate to delete) med_df.isnull().all(axis=1).any() was run, where "axis=1' is used to denote the x-axis or rows. Next, using the .head() and .tail() methods (Figure 4), I'm able to peek into the data and briefly verify the first and last few tuples within the data set.

```
[26]: med_df.head(), med_df.tail()

[26]: (   Unnamed: 0  CaseOrder Customer_id                           Interaction  \
      0           1          1     C412403  8cd49b13-f45a-4b47-a2bd-173ffa932c2f
      1           2          2     Z919181  d2450b70-0337-4406-bdbb-bc1037f1734c
      2           3          3     F995323  a2057123-abf5-4a2c-abad-8ffe33512562
      3           4          4     A879973  1dec528d-eb34-4079-adce-0d7a40e82205
      4           5          5     C544523  5885f56b-d6da-43a3-8760-83583af94266
```

Figure 4 - Using the head() and tail() methods to inspect the data's start and end.

The .describe() method also provided an overview of descriptive statistics per data field, e.g., count, min and max, mean, standard deviation of observations, data types and percentiles 25% 50% and 75%.

Inspect Column Data

```
[27]: med_df.describe()
```

[27]:

	Unnamed: 0	CaseOrder	Zip	Lat	Lng	Population
count	10000.00000	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000
mean	5000.50000	5000.50000	50159.323900	38.751099	-91.243080	9965.253800
std	2886.89568	2886.89568	27469.588208	5.403085	15.205998	14824.758614
min	1.00000	1.00000	610.000000	17.967190	-174.209690	0.000000
25%	2500.75000	2500.75000	27592.000000	35.255120	-97.352982	694.750000
50%	5000.50000	5000.50000	50207.000000	39.419355	-88.397230	2769.000000
75%	7500.25000	7500.25000	72411.750000	42.044175	-80.438050	13945.000000
max	10000.00000	10000.00000	99929.000000	70.560990	-65.290170	122814.000000

Figure 5 - Using the describe() method to provide basic statistics for each column.

Cleaning Findings

Through this exploration, the shape of the data was better understood, e.g., 10,000 rows by 53 columns. After running the .info() method, series with nulls were noted in addition to certain data types, which didn't seem to be the best fit. The Overweight and Anxiety columns were both float64; yet, contained categorical data. Both series were marked for conversion to an object data type. No rows containing only nulls were detected. The first two columns provided the case order data; column' Unnammed: 0' was marked for deletion.

After exploring the data and making some notes, I focused on researching null values. First, by listing only columns containing at least one NaN, med_df.loc[:, med_df.isnull().any()] was used. All columns but one were subsequently filled with the columns’ mean value. The Soft_drink column has categorical data and to fill this, med_df[['Soft_drink']] = med_df[['Soft_drink']].fillna(method='bfill') was used.

Next, I focused on outliers. I created a list of non-categorical columns to generate boxplots and basic statistics for Children, Age, Income, and Initial Days. For each series, a comparison was created to view changes between original values and the interpolation replacement choices of mean, mode, or NaN. Additionally, changes were noted on the boxplot diagram comparison, as seen in Figure 6 below.

```
compare = pd.DataFrame({'before':Children_bp, 'NaN':new_children, 'mean':children_mean, 'mode':children_mode})
compare.plot.box()
compare.describe()
```

	before	NaN	mean	mode
count	10000.000000	9543.000000	10000.000000	9543.000000
mean	2.098219	1.809828	1.823008	1.809828
std	1.855638	1.318207	1.289138	1.318207
min	0.000000	0.000000	0.000000	0.000000
25%	1.000000	1.000000	1.000000	1.000000
50%	2.098219	2.000000	2.098219	2.000000
75%	3.000000	2.098219	2.098219	2.098219
max	10.000000	6.000000	6.000000	6.000000

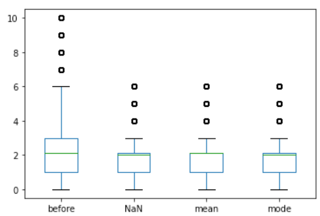

Figure 6 - Comparison of Children Variable Original Data with Interpolated Versions of the Series Using NaN, Mean, and Mode.

Once the missing data were filled in, the changes were committed and verified. Next, the data types of the Overweight and Anxiety columns were converted from float64 to objects. The UID column provides unique values, which were then set to an index. Some column header nomenclature was noted to be consistent earlier in data exploration, so these column names were reworded to adhere to the original style.

Principal Components

Principle Component Analysis (PCA) – First, the data frame was reduced to contain int and float data types. The dataset was verified to include only columns containing numbers while all NaNs were removed. The data subset was normalized, and a new PCA data set of components was created to fit. A scree plot of eigenvalues was displayed using this data. To help establish the optimal number of components for our analysis, two lines were designed to identify where the number of components crossed the horizontal dashed red line representing an eigenvalue value.

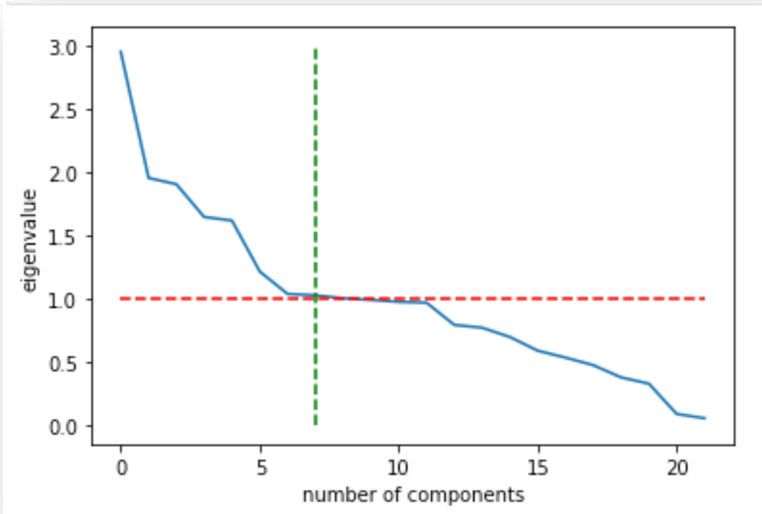

Figure 7 - Scree Plot of Eigenvalues

Justification of Mitigation Methods

The shape of the data helped conceptualize the data frame. Data types for two-column headers were changed to fit better with the categorical data contained. The first column was redundant since it was a duplicate of the second column; therefore, it was deleted. I followed Reindert-Jan Ekker's approach (*Pandas Playbook: Manipulating Data | Pluralsight*, 2021) in building boxplots with whiskers and descriptive statistics like the count, mean, standard deviation, min, max, and quartiles. Within the "box" or interquartile range (IQR), the median is displayed. From two lines of code, boxplots and descriptive statistics provide a lot of information within a graph and table. I preferred visual confirmation and its intuitiveness. After comparing each series' original data to various interpolation methods, I noticed in both the boxplot and using the .describe() method the data seemed tighter when using the mean. For instance, see in Figure 8 below that you can compare left-to-right different statistical values based on which data point was used to fill any outliers. Also, notice using the mean keeps the count to a full 10,000 while staying relatively unchanged.

[73]:

	before	NaN	mean	mode
count	9587.000000	9438.000000	10000.000000	9438.000000
mean	30468.913519	29530.337008	29583.085008	29530.337008
std	20818.317309	19582.143531	19025.100498	19582.143531
min	154.080000	154.080000	154.080000	154.080000
25%	14572.400000	14572.400000	14572.400000	14572.400000
50%	22745.070000	22230.885000	24254.650000	22230.885000
75%	43110.525000	41829.735000	40413.560000	41829.735000
max	94301.200000	85897.120000	85897.120000	85897.120000

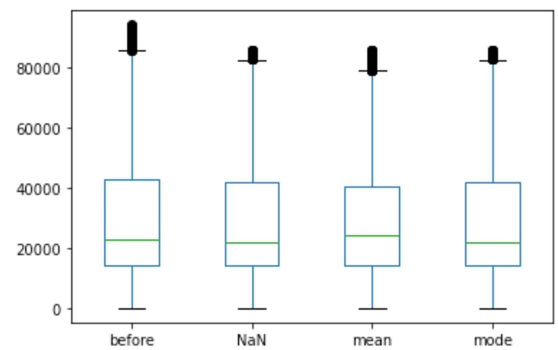

Figure 8 - Comparing Imputation Approaches for the Income Series

I did note a method for dropping rows that don't meet a predefined threshold: df.dropna(thresh=someNumber). At this point in the cleaning process, guessing at a threshold felt arbitrary, so I decided to keep all rows.

Criteria Used

PCA only works with numerical data types; therefore, the data must be reduced to only contain these types. Following lesson 6, the data needs to be normalized. Viewing the graph (Figure 7), especially with the help of the horizontal and vertical dashed lines, helped clearly understand how many components have an eigenvalue of at least 1. The benefit of using more than seven components quickly diminishes. Adding additional gradient styling to the PCA data frame helped illuminate which columns had an absolute effect on the first seven components.

Benefits

When comparing Figure 9 and Figure 10, the first apparent benefit is a reduction of data needed to process and analyze. Figure 10 provides this reduction from 22 to 7 main groups, e.g., the most important variable components to consider. By working with a subset of data, any computation performance time such as algorithms and future analysis, should also be improved.

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
Zip	-0.009220	0.086436	0.696792	-0.020780	0.024908	0.055877	0.011011	-0.000163	-0.000280	0.010158
Lat	0.008947	-0.015955	0.059795	0.008107	-0.002409	-0.699485	-0.146564	-0.012704	-0.007317	-0.061000
Lng	0.004891	-0.087058	-0.701363	0.017842	-0.020689	0.033878	0.000638	-0.002261	0.002832	0.001468
Population	0.009052	0.028359	0.033643	0.023195	-0.023036	0.695287	0.047750	-0.004605	-0.016700	-0.056496
Children	0.011703	0.009158	-0.007928	-0.021050	-0.016519	0.011484	-0.274797	0.464181	-0.613364	0.071009
Age	0.000474	0.079126	-0.038190	-0.014262	0.700224	0.011495	-0.023100	0.024471	0.006297	0.001021
Income	-0.009636	0.001964	0.005742	-0.009935	-0.006487	0.027342	-0.194617	-0.509968	-0.401373	0.676395
VitD_levels	-0.009552	0.532145	-0.073932	0.053899	-0.051998	-0.072988	0.357024	-0.257373	-0.030675	0.068823
Doc_visits	0.007119	-0.004071	0.001414	-0.007525	0.012581	0.016025	-0.071275	-0.464109	-0.499106	-0.711890
Full_meals_eaten	-0.000683	-0.008814	0.020089	0.018374	0.038828	-0.104522	0.598935	0.091294	-0.107838	-0.001131
VitD_supp	-0.005259	0.033593	0.004121	0.009670	0.011766	0.039951	-0.449154	-0.357144	0.443965	-0.082504
Initial_days	-0.020543	0.440549	-0.036461	0.067564	-0.070749	0.046659	-0.408221	0.318583	0.020240	-0.086619
Total_charge	-0.018610	0.692031	-0.081602	0.084234	-0.075849	-0.027351	0.015761	0.002355	-0.007129	-0.001902
Additional_charges	0.003956	0.082103	-0.027582	-0.024242	0.700400	0.009911	-0.021678	0.003319	-0.017146	0.015478
Timely_admission	0.454733	-0.023449	0.012047	0.295061	0.009869	0.002631	0.007328	-0.002834	0.003938	0.004045
Timely_treatment	0.428494	-0.023006	0.016632	0.291270	0.011687	0.015458	-0.017216	0.015732	-0.002483	-0.004274
Timely_visits	0.395257	-0.023473	0.011692	0.294548	0.009758	-0.017640	-0.005443	0.012844	0.020912	0.004519
Reliability	0.151964	0.061444	-0.019592	-0.553853	-0.025300	0.016333	0.004669	0.022621	0.039899	0.004015
Options	-0.189857	-0.069300	0.021386	0.578704	0.022555	0.006861	-0.003482	-0.016248	-0.019218	0.028300
Hours_of_treatment	0.410036	0.031953	0.005552	-0.161152	-0.015017	-0.003877	0.018567	-0.025961	-0.002702	0.003068
Courteous_staff	0.356451	0.037378	-0.004029	-0.170051	0.000435	-0.000670	0.009201	0.013731	-0.008897	-0.016114
Active_listening_evidence_from_dr	0.312413	0.028004	-0.015003	-0.164509	-0.015995	-0.000073	-0.031308	-0.050858	-0.016149	0.056651

	...	PC13	PC14	PC15	PC16	PC17	PC18	PC19	PC20	PC21	PC22
Zip	...	-0.116193	0.009127	-0.013300	-0.001869	-0.000450	0.005859	-0.001616	-0.010806	-0.698542	0.022671
Lat	...	0.675588	-0.025979	0.085712	-0.025889	0.014863	-0.001484	-0.014314	-0.007155	-0.115351	0.002949
Lng	...	-0.022747	-0.005864	-0.007955	0.007203	0.004629	0.005977	0.006880	-0.006032	-0.704651	0.023908
Population	...	0.704587	-0.012113	0.086016	-0.019453	-0.016106	-0.025765	-0.013844	-0.006916	-0.027551	-0.000850
Children	...	-0.023648	0.031118	-0.008804	0.021563	-0.001974	-0.011795	-0.007466	-0.003795	-0.001222	-0.005269
Age	...	-0.005927	-0.000873	0.023686	-0.004823	0.032007	-0.058399	-0.701299	-0.050004	-0.000240	-0.016643
Income	...	0.039259	-0.088571	-0.003892	-0.018287	-0.020164	0.009456	-0.022905	-0.006327	0.003952	-0.000776
VitD_levels	...	0.007765	-0.007208	-0.008287	-0.003667	-0.007810	0.009934	-0.024789	0.009590	0.012783	0.543912
Doc_visits	...	-0.079372	-0.012829	0.021971	-0.024173	0.008577	-0.010674	-0.003326	-0.001860	0.002573	-0.000249
Full_meals_eaten	...	0.071578	0.056681	0.041556	-0.012913	0.000686	0.014024	-0.010513	0.000464	-0.001945	-0.001411
VitD_supp	...	-0.019726	0.041140	-0.029517	0.014742	-0.002617	0.009679	-0.005982	-0.000123	0.005389	-0.001728
Initial_days	...	-0.015067	-0.003318	0.023226	0.008778	0.012380	-0.002275	-0.003785	-0.019135	0.018548	0.450863
Total_charge	...	-0.004721	-0.001611	0.004061	-0.005611	-0.000077	-0.001832	0.021597	0.003045	-0.025086	-0.706153
Additional_charges	...	0.031041	0.003387	-0.006271	0.014078	-0.027096	0.051576	0.701726	0.048691	0.002984	0.025952
Timely_admission	...	-0.027069	-0.095938	0.071458	-0.010610	0.082971	0.187278	0.048958	-0.803242	0.010951	-0.004554
Timely_treatment	...	-0.020987	-0.147174	0.131187	-0.061132	0.097164	0.623168	-0.076276	0.532057	-0.002920	-0.003163
Timely_visits	...	-0.061676	-0.203519	0.204769	-0.239478	-0.430350	-0.621952	0.027989	0.193051	-0.009768	0.006752
Reliability	...	-0.054991	-0.365325	0.354994	-0.392741	0.480819	-0.106546	0.040851	-0.009494	0.000524	0.000931
Options	...	-0.007392	0.121816	-0.059034	-0.135623	0.695182	-0.299551	0.050103	0.095487	-0.000033	0.003949
Hours_of_treatment	...	0.027015	-0.054472	-0.054385	0.793562	0.272934	-0.275619	0.009139	0.126230	-0.002842	-0.001915
Courteous_staff	...	0.094264	0.042667	-0.841430	-0.332232	0.069371	-0.061555	-0.013973	0.050312	0.000888	0.005353
Active_listening_evidence_from_dr	...	-0.015060	0.871623	0.275503	-0.151087	0.041686	-0.037146	0.005111	0.033581	-0.002340	0.005068

22 rows × 22 columns

Figure 9 - All 22 Components Prior to Data Reduction

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Zip	-0.009220	0.086415	0.696788	-0.020785	0.024914	0.055879	0.011222
Lat	0.008949	-0.015942	0.059819	0.008167	-0.002423	-0.699525	-0.147828
Lng	0.004892	-0.087040	-0.701368	0.017832	-0.020695	0.033847	0.000661
Population	0.009054	0.028365	0.033655	0.023258	-0.023086	0.695203	0.046835
Children	0.011699	0.009144	-0.007962	-0.021162	-0.016478	0.011324	-0.268949
Age	0.000460	0.079220	-0.038110	-0.014520	0.700614	0.013692	-0.018574
Income	-0.009637	0.001958	0.005729	-0.009971	-0.006473	0.027374	-0.195025
VitD_levels	-0.009554	0.532142	-0.073930	0.053862	-0.051989	-0.073058	0.358183
Doc_visits	0.007119	-0.004061	0.001425	-0.007543	0.012617	0.016427	-0.068931
Full_meals_eaten	-0.000684	-0.008819	0.020079	0.018351	0.038836	-0.104581	0.602233
VitD_supp	-0.005263	0.033575	0.004083	0.009554	0.011807	0.039879	-0.446301
Initial_days	-0.020542	0.440559	-0.036434	0.067581	-0.070751	0.046828	-0.409395
Total_charge	-0.018610	0.692031	-0.081584	0.084245	-0.075868	-0.027398	0.015516
Additional_charges	0.003971	0.082014	-0.027649	-0.023851	0.700010	0.007778	-0.026513
Timely_admission	0.454727	-0.023448	0.011966	0.295120	0.009645	0.001267	0.005930
Timely_treatment	0.428465	-0.023204	0.016324	0.290567	0.011877	0.014695	-0.010981
Timely_visits	0.395271	-0.023405	0.011834	0.294785	0.009739	-0.016829	-0.006814
Reliability	0.151977	0.061525	-0.019436	-0.553627	-0.025240	0.017282	0.003543
Options	-0.189825	-0.069114	0.021740	0.579305	0.022511	0.009045	-0.007822
Hours_of_treatment	0.410069	0.032148	0.005925	-0.160525	-0.015016	-0.001707	0.013700
Courteous_staff	0.356446	0.037349	-0.004080	-0.170150	0.000454	-0.001062	0.010063
Active_listening_evidence_from_dr	0.312419	0.028041	-0.014931	-0.164392	-0.015980	0.000429	-0.032941

Figure 10 - Heat Map of the Seven Principal Components

Summary of Outcomes

There were about six phases to cleaning this data, depending on how they could be broken up:

- Detection and Inspection: exploring the data, noting discrepancies in column null counts, naming conventions, and data types. Look for rows or columns that may contain only NaNs and any missing data; consider how you might fill these.
- Handling Missing Data: During this assessment, I decided to NaNs with either the column's mean value (numerical) or backfill (categorical). I did not find rows or columns with only null values and didn't use a threshold. I did drop the first column since it was redundant.

- Handling Outliers: Boxplots and descriptive statistics were used to compare interpolation methods when replacing outliers with the mean, mode, or NaNs. After viewing the comparisons, I chose to replace outliers with the column mean.
- Handling Duplicates: I looked but did not find duplicates.
- Adding an Index, Data Type Conversions, and Column Header Cleanup: Researched the UID for duplicates; since it was truly unique, I set it as index. The Overweight and Anxiety column data types were converted from float64 to objects. Twelve column headers were renamed to conform to the nomenclature.
- PCA: reduced dataset to only contain numbers, verified all NaNs are removed. Next, the data is normalized, and components were extracted to view a Scree plot of eigenvalues. I placed both a horizontal and vertical dashed line to help increase the accuracy of selecting only seven components from the total 22. To help with my understanding, I created another view with the PCAs reduced to 7 and styled it as a heatmap.

Mitigation Code

Provided: JWillis_D206DataCleaning.ipynb

Clean Data

Provided: medical_raw_cleaned.csv

Limitations and Impact of the Limitations

Outside of my limited knowledge and experience, the approach felt logical most of the time. For me, it's tough to make decisions that feel blurred. For instance, I went back and forth comparing outliers to choose an interpolation approach. I wasn't sure if I should try to convert categorical data into numerical, which would allow it to be analyzed. Occasionally, JupyterLab was a bit clunky and didn't perform as expected. It's hard to deeply understand PCA and how it works behind the code at this point since we will learn about it deeper in future courses.

Video

Panopto video uploaded.

Sources for Third-Party Code

- Help using Markdown: https://www.markdownguide.org/basic-syntax/
- Help with Indexing: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
- Help to create dashed line style on scree plot: https://matplotlib.org/2.1.2/api/_as_gen/matplotlib.pyplot.plot.html
- Leveraging pandas.DataFrame.fillna: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.fillna.html
- Building the heat map: https://pandas.pydata.org/docs/user_guide/style.html

References

Ekker, Reindert-Jan. Pandas Playbook: Manipulating Data, Pluralsight. (2021). Pluralsight.com. https://app.pluralsight.com/library/courses/pandas-playbook-manipulating-data/

Larose, C. D., Larose D. T. (2020). Data Science using Python and R - Chapter 3: DATA PREPARATION. O'Reilly Media. https://learning.oreilly.com/library/view/Data+Science+Using+Python+and+R/9781119526810/c03.xhtml

Hunter, J., Dale, D., et al. — Matplotlib 2.1.2 documentation for Styling Output. (2012). Matplotlib.org. https://matplotlib.org/2.1.2/api/_as_gen/matplotlib.pyplot.plot.html

Pugh, J., Penney, L. S., Noël, P. H., Neller, S., Mader, M., Finley, E. P., Lanham, H. J., & Leykum, L. (2021). Evidence-based processes to prevent readmissions: more is better, a ten-site observational study. *BMC Health Services Research*, *21*(1), 189. https://doi.org/10.1186/s12913-021-06193-x

Vallisneri, M. (2018). Python Statistics Essential Training: Data Cleaning. LinkedIn Learning