Data Cleaning - D206 Project

Medical Data

For this project, I have chosen the medical data set ("medical_raw data.csv"). This data contains information about patients at a hospital, including whether or not they were readmitted. Readmission of patients is a problem in the medical industry, and hospitals are penalized based on excessive readmissions.

In [127…
```
import pandas as pd
import numpy as np

dfMed = pd.read_csv('medical_raw data.csv')
dfMed.shape
```

Out[127…
```
(10000, 53)
```

The dataset contains 10000 rows and 53 columns.

Part I: Research Question

A. Describe one question or decision that you will address using the data set you chose. The summarized question or decision must be relevant to a realistic organizational need or situation.

We will attempt the most basic question in order to lower readmission rates - Can we determine which parameters drive higher rates of readmission?

B. Describe the variables in the data set and indicate the specific type of data being described. Use examples from the data set that support your claims.

First we will load the file and look at the top few records of each column.

In [128…
```
# Allows all columns to return
pd.set_option('max_columns', None)

# View first 5 records to get an idea of the data
dfMed.head()
```

Out[128…

	Unnamed: 0	ID	Customer_id	Interaction	UID	zip	Lat	Lng	City	State	Population	County	Area	Timezone	Jo
0	1	1	C412403	8cd49b13-f45a-4b47-a2bd-173ffa932c2f	3a83ddb66e2ae73798bdf1d705dc0932	35621	34.34960	-86.72508	Eva	AL	2951	Morgan	Suburban	America/Chicago	Psychologis sport ar exerci
1	2	2	Z919181	d2450b70-0337-4406-bdbb-bc1037f1734c	176354c5eef714957d486009feabf195	32446	30.84513	-85.22907	Marianna	FL	11303	Jackson	Urban	America/Chicago	Communi developme work
2	3	3	F995323	a2057123-abf5-4a2c-abad-8ffe33512562	e19a0fa00aeda885b8a436757e889bc9	57110	43.54321	-96.63772	Sioux Falls	SD	17125	Minnehaha	Suburban	America/Chicago	Chi Executi Offic
3	4	4	A879973	1dec528d-eb34-4079-adce-0d7a40e82205	cd17d7b6d152cb6f23957346d11c3f07	56072	43.89744	-93.51479	New Richland	MN	2162	Waseca	Suburban	America/Chicago	Early yea teach
4	5	5	C544523	5885f56b-d6da-43a3-8760-83583af94266	d2f0425877b10ed6bb381f3e2579424a	23181	37.59894	-76.88958	West Point	VA	5287	King William	Rural	America/New_York	Heal promoti speciali

In [129…
```
# Get an overview of the numeric fields
dfMed.describe()
```

Out[129…

	Unnamed: 0	ID	zip	Lat	Lng	Population	Children	Age	Income	vitD_levels	doc_visits	full_meals_eaten	vitD
count	10000.00000	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	7412.000000	7586.000000	7536.000000	10000.000000	10000.000000	10000.000000	10000.0(
mean	5000.50000	5000.50000	50159.323900	38.751099	-91.243080	9965.253800	2.098219	53.295676	40484.438268	19.412675	5.012200	1.001400	0.3!
std	2886.89568	2886.89568	27469.588208	5.403085	15.205998	14824.758614	2.155427	20.659182	28664.861050	6.723277	1.045734	1.008117	0.6:
min	1.00000	1.00000	610.000000	17.967190	-174.209690	0.000000	0.000000	18.000000	154.080000	9.519012	1.000000	0.000000	0.0(
25%	2500.75000	2500.75000	27592.000000	35.255120	-97.352982	694.750000	0.000000	35.000000	19450.792500	16.513171	4.000000	0.000000	0.0(
50%	5000.50000	5000.50000	50207.000000	39.419355	-88.397230	2769.000000	1.000000	53.000000	33942.280000	18.080560	5.000000	1.000000	0.0(
75%	7500.25000	7500.25000	72411.750000	42.044175	-80.438050	13945.000000	3.000000	71.000000	54075.235000	19.789740	6.000000	2.000000	1.0(
max	10000.00000	10000.00000	99929.000000	70.560990	-65.290170	122814.000000	10.000000	89.000000	207249.130000	53.019124	9.000000	7.000000	5.0(

In [130…
```
# Find the distinct count of values in each column
dfMed.nunique()
```

Out[130…
```
Unnamed: 0            10000
ID                    10000
Customer_id           10000
Interaction           10000
UID                   10000
zip                    8612
Lat                    8588
Lng                    8601
City                   6072
State                    52
Population             5951
County                 1607
Area                      3
Timezone                 26
Job                     639
Children                 11
Age                      72
Education                12
```

```
Employment                5
Income                 7531
Marital                   5
ReAdmis                   2
Gender                    3
vitD_levels           10000
doc_visits                9
full_meals_eaten          8
vitD_supp                 6
Soft_drink                2
Initial_Admin             3
HighBlood                 2
Stroke                    2
Complication_Risk         3
Overweight                2
Arthritis                 2
Diabetes                  2
Hyperlipidemia            2
BackPain                  2
Anxiety                   2
Allergic_rhinitis         2
Reflux_esophagitis        2
Asthma                    2
Services                  4
Initial_Days           8944
TotalCharge           10000
Additional_Charges     8888
item1                     8
item2                     7
item3                     8
item4                     7
item5                     7
item6                     7
item7                     7
item8                     7
dtype: int64
```

Findings

Based on the data, data dictionary, and numeric summaries, here is a brief summary review of each column.

1. Unnamed: 0
   - This appears to be an order index ranging from 1-10000. The data dictionary listed CaseOrder as the first column and as a placeholder for the sort order, but that actually appears to be stored in both of the first two columns.
2. ID
   - See above
3. Customer_id
   - A unique patient ID, consisting of numbers and characters.
4. Interaction
   - A unique ID, storing a GUID
5. UID
   - A unique ID, consisting of numbers and characters.
6. zip
   - The postal code of the patient's residence. Over 8600 postal codes are included.
7. Lat
   - The latitude of the patient's residence.
8. Lng
   - The longitude of the patient's residence.
9. City
   - The city of the patient's residence. Over 6000 cities are included.
10. State
    - The state of the patient's residence. Over 50 are included. This could indicate Nulls, blanks, or smaller regions like the District of Columbia or Puerto Rico.
11. Population
    - The population with a mile radius of the patient's residence.
12. County
    - The county of the patient's residence.
13. Area
    - A classification of the patient's residence. Options are Rural, Urban, and Suburban.
14. Timezone
    - The timezone of the patient's residence.
15. Job
    - A string describing the occupation of the primary insurance holder.
16. Children
    - The number of children in the patient's household. This includes children that are not direct children of the patient.
17. Age
    - The age of the patient during admission.
18. Education
    - The education level of the primary insurance holder.
19. Employment
    - The employeement status of the primary insurance holder.
20. Income
    - The annual income of the primary insurance holder.
21. Marital
    - The marital status of the primary insurance holder.
22. ReAdmis
    - A Yes or No stating if the patient was readmitted.
23. Gender
    - The gender (male/female). Has 3 distinct values, so an empty, null, or 3rd option exists.
24. vitD_levels
    - The patient's vitamin D levels measured in ng/mL
25. doc_visits
    - The number of times the doctor visited the patient during the initial hospitalization.
26. full_meals_eaten

- The number of full meals the patient ate during hospitalization.

27. vitD_supp
    - The number of times vitamin D supplements were administered to the patient.
28. Soft_drink
    - Yes or No indicating if the patient drinks more than 3 sodas per day.
29. Initial_Admin
    - A categorical value describing how the patient was initially admitted to the hospital - Observation, Elective, or Emergency.
30. HighBlood
    - Yes or No indicating if the patient has high blood pressure.
31. Stroke
    - Yes or No indicating if the patient has had a stroke.
32. Complication_Risk
    - Level of complication risk for the patient as assessed by the doctor (high, medium, or low).
33. Overweight
    - Yes or No indicating if the patient is overweight based on age, gender, and height.
34. Arthritis
    - Yes or No indicating if the patient has arthritis.
35. Diabetes
    - Yes or No indicating if the patient has diabetes.
36. Hyperlipidemia
    - Yes or No indicating if the patient has hyperlipidemia.
37. BackPain
    - Yes or No indicating if the patient has back pain.
38. Anxiety
    - Yes or No indicating if the patient has anxiety.
39. Allergic_rhinitis
    - Yes or No indicating if the patient has allergic rhinitis.
40. Reflux_esophagitis
    - Yes or No indicating if the patient has reflux esophagitis.
41. Asthma
    - Yes or No indicating if the patient has asthma.
42. Services
    - A string representing the primary service that the patient received while hospitalized.
43. Initial_Days
    - The number of days the patient stayed during the initial visit.
44. TotalCharge
    - The average amount charged to the patient daily.
45. Additional_Charges
    - The average amount charged to the patient for miscellaneous procedures, treatments, medicines, anesthesiology, etc.
46. item1
    - How important timely admission is to the patient on a scale of 1 (most important) to 8 (least important).
47. item2
    - How important timely treatment is to the patient on a scale of 1 (most important) to 8 (least important).
48. item3
    - How important timely visits are to the patient on a scale of 1 (most important) to 8 (least important).
49. item4
    - How important reliability is to the patient on a scale of 1 (most important) to 8 (least important).
50. item5
    - How important options are to the patient on a scale of 1 (most important) to 8 (least important).
51. item6
    - How important hours of treatment are to the patient on a scale of 1 (most important) to 8 (least important).
52. item7
    - How important courteous staff is to the patient on a scale of 1 (most important) to 8 (least important).
53. item8
    - How important evidence of active listening from the doctor is to the patient on a scale of 1 (most important) to 8 (least important).

Part II: Data-Cleaning Plan

C. Explain the plan for cleaning the data by doing the following:

Plan to Clean Data

I will perform the following steps to clean and prep the dataset.

1. Remove unnecessary columns. I will base this process on the question we are trying to answer and the understanding of the data.

2. Detect and fill (or remove) NaN, null, and blank values where necessary. For each column, we will determine if blank values can be filled with a default or average, or if the records need to be removed from consideration.

3. Review each column to look for values that should be converted to another type (int to string, vice versa) for bucketing or other purposes.

4. Save the cleaned dataset.

Language and Justification

I will be using Python with the following supporting libraries. My choice of Python is driven by the desire to produce code that can easily be deployed into infrastructures and software companies, and blend in more general code with data analysis.

- Pandas (for dataframe capabilies)
- SciPy (for PCA capabilities)
- MatPlotLib (for graphing capabilities)

Part III: Data Cleaning

D. Summarize the data-cleaning process by doing the following:

Remove Unnecessary Columns

Both "Unnamed: 0" and "ID" appear to be some sort of index, ordering columns. The data dictionary lists "CaseOrder" as the first column and describes it as a placeholder to preserve the order, but both columns are holding the same values.

Below is the intial review of the columns and which will be kept and removed:

The value we are trying to predict.
ReAdmis

Will be kept and used in PCA.
Lat, Lng, Population, Children, Age, Income, vitD_levels, doc_visits, full_meals_eaten, vitD_supp, Overweight, Anxiety, Initial_Days, item1, item2, item3, item4, item5, item6, item7, item8

> I considered removing the survey results, but finally concluded that the mental priorities of a patient could indicate their predilection for hospital visits.

Will be reviewed and kept in dataset.
Area, Education, Employment, Marital, Gender, Soft_drink, Initial_Admin, HighBlood, Stroke, Complication_Risk, Arthritis, Diabetes, Hyperlipidemia, BackPain, Allergic_rhinitis, Reflux_esophagitis, Asthma, Services

> Will explore to see if these can be transformed into into numeric values and used.

Will be removed.
Unnamed: 0, ID, Customer_Id, Interation, UID

> Ordering or Identity columns that will not impact Readmission.

zip, City, State, County, Timezone

> While address/location might impact Readmission, I prefer the latitude and longitude since they are continuous.

Job

> There are over 600 distinct jobs and we cannot convert to a quick numeric indicator to scale and detect outliers.

TotalCharge, Additional_Charges

> While I believe the charges and chance of readmission is correlated, I believe they would both be driven by the other statistics and not be causation.

In [131…
```
# Remove the unused columns
dfMedTr = dfMed.drop(columns = ['Unnamed: 0', 'ID', 'Customer_id', 'Interaction', 'UID', 'zip', 'State', 'County', 'Job', 'Timezone', 'TotalCharge', 'Additiona
```

Review and Handle Empty Values

First, get a list of the columns that have true nulls(NaN). We will pay extra special attention to these columns during the review.

In [132…
```
dfMedTr.columns[dfMedTr.isna().any()].tolist()
```

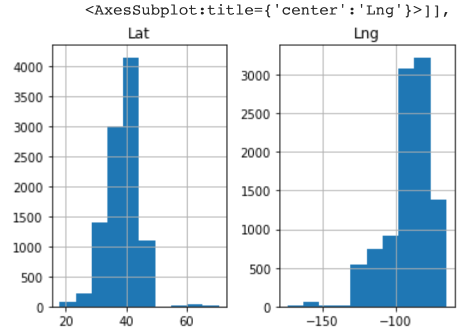
Out[132…
```
['Children',
 'Age',
 'Income',
 'Soft_drink',
 'Overweight',
 'Anxiety',
 'Initial_Days']
```

Lat, Lng

> No null values so just check data visually to make the data is valid.

In [133…
```
dfMedTr[['Lat', 'Lng']].hist()
```

Out[133…
```
array([[<AxesSubplot:title={'center':'Lat'}>,
        <AxesSubplot:title={'center':'Lng'}>]], dtype=object)
```



City

> We know there were no nulls, so just review some of the options and make sure there aren't blanks.

In [134…
```
dfMedTr.City.value_counts()
```

Out[134…
```
Houston           36
San Antonio       26
Springfield       22
Miami             21
New York          21
                  ..
Chelan             1
Beverly Shores     1
Hampden Sydney     1
Violet Hill        1
Hiawatha           1
Name: City, Length: 6072, dtype: int64
```

In [135…
```
dfMedTr[dfMedTr.City.str.strip() == ""].shape[0]
```

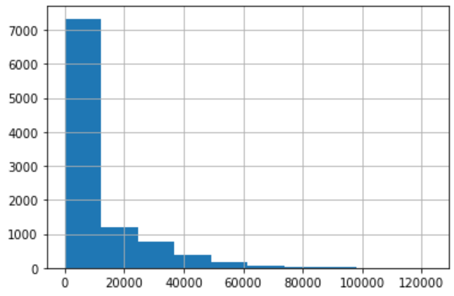
Out[135… 0

Population

> No null values so just check data visually to make the data is valid.

In [136…
```
dfMedTr.Population.hist()
```

Out[136… <AxesSubplot:>



In [138…
```
dfMedTr[dfMedTr.Population == 0].shape[0]
```

Out[138… 109

> There are some zero populations, but I could see the population being very low for the bulk of people (below 10,000). We will replace the zeros with 1 to at least account for the patient themselves.

In [139…
```
dfMedTr['Population'] = [i if i != 0 else 1 for i in dfMedTr['Population']]
```

Area

> There are no nulls present, but we will still review the value counts to look for blank strings.

In [140…
```
dfMedTr.Area.value_counts()
```

Out[140…
```
Rural       3369
Suburban    3328
Urban       3303
Name: Area, dtype: int64
```

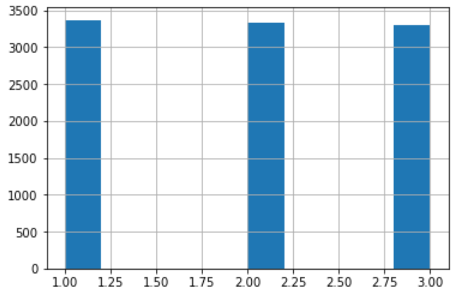
> Since this is an indicator of how close people are to services and other people, I believe we can change this to a sliding scale of 1 for Rural, 2 for Suburban, and 3 for Urban.

In [141…
```
# create a list of conditions
conditions = [(dfMedTr['Area'] == "Rural")
             , (dfMedTr['Area'] == "Suburban")
             , (dfMedTr['Area'] == "Urban")]

# create a list of the values
values = [1,2,3]

dfMedTr['AreaClass'] = np.select(conditions, values)
dfMedTr.AreaClass.value_counts()
```

Out[141…
```
1    3369
2    3328
3    3303
Name: AreaClass, dtype: int64
```

In [142…
```
dfMedTr.AreaClass.hist()
```

Out[142… <AxesSubplot:>

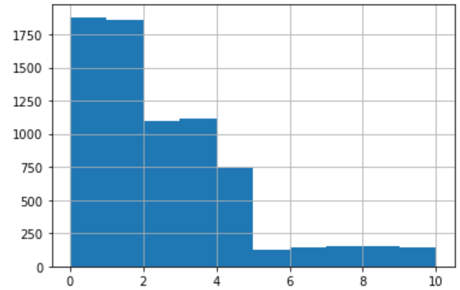


Children

> There are nulls present, we need to decide how to fill the null values (or remove).

In [143…
```
dfMedTr.Children.hist()
```

Out[143… <AxesSubplot:>



> A quick search on Google shows that the average number of children in the U.S. is 1.93. We will check the mean to see if we are already at that value.

In [144…
```
dfMedTr.Children.mean()
```
Out[144…
```
2.0982191041554237
```

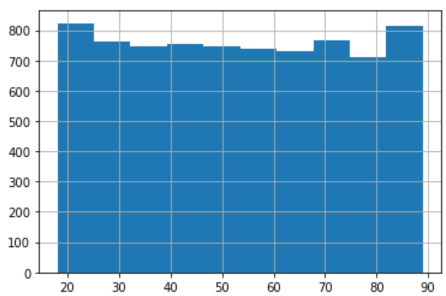
> I don't want to remove null records, because that would account for around 1/4th of the dataset.
>
> Since we are slightly above the correct mean, filling a quarter of the records pulled the average down below the correct mean. I do not want to assume the correct number of children or use a float, so we will consider the variance to be acceptable in this case and convert the null values to zero.

In [145…
```
dfMedTr['Children'].fillna(0, inplace = True)
dfMedTr.Children.mean()
```
Out[145…
```
1.5552
```

Age

> There are nulls present, so we will need to decide how to fill.

In [146…
```
dfMedTr.Age.hist()
```
Out[146…
```
<AxesSubplot:>
```



In [147…
```
dfMedTr.Age.describe()
```
Out[147…
```
count    7586.000000
mean       53.295676
std        20.659182
min        18.000000
25%        35.000000
50%        53.000000
75%        71.000000
max        89.000000
Name: Age, dtype: float64
```

In [148…
```
dfMedTr.Age.mode()
```
Out[148…
```
0    30.0
dtype: float64
```

> I do not want to use the mode, because it is skewed towards the bottom. We will fill the nulls with the average age of 53.

In [149…
```
dfMedTr['Age'].fillna(53, inplace = True)
```

Education

> There are no nulls present, so we will check the value counts to make sure there are no blank strings.

In [150…
```
dfMedTr.Education.value_counts()
```
Out[150…
```
Regular High School Diploma                 2444
Bachelor's Degree                           1724
Some College, 1 or More Years, No Degree    1484
9th Grade to 12th Grade, No Diploma          832
Associate's Degree                           797
Master's Degree                              701
Some College, Less than 1 Year               642
Nursery School to 8th Grade                  552
GED or Alternative Credential                389
Professional School Degree                   208
No Schooling Completed                       133
Doctorate Degree                              94
Name: Education, dtype: int64
```

> Since this is a progression of education, I am going to create a value that represents the amount of education as a number. This will allow it to be used in the PCA later.

In [151…
```
# create a list of conditions
conditions = [(dfMedTr['Education'] == "No Schooling Completed")
              , (dfMedTr['Education'] == "Nursery School to 8th Grade")
```

```
                     , (dfMedTr['Education'] == "9th Grade to 12th Grade, No Diploma")
                     , (dfMedTr['Education'] == "GED or Alternative Credential")
                     , (dfMedTr['Education'] == "Regular High School Diploma")
                     , (dfMedTr['Education'] == "Some College, Less than 1 Year")
                     , (dfMedTr['Education'] == "Some College, 1 or More Years, No Degree")
                     , (dfMedTr['Education'] == "Associate's Degree")
                     , (dfMedTr['Education'] == "Professional School Degree")
                     , (dfMedTr['Education'] == "Bachelor's Degree")
                     , (dfMedTr['Education'] == "Master's Degree")
                     , (dfMedTr['Education'] == "Doctorate Degree")]

# create a list of the values
values = [0,1,2,3,4,5,6,7,8,9,10,11]

dfMedTr['EducationClass'] = np.select(conditions, values)
dfMedTr.EducationClass.value_counts()
```

Out[151…

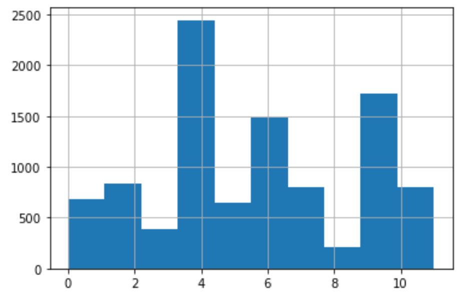
```
4     2444
9     1724
6     1484
2      832
7      797
10     701
5      642
1      552
3      389
8      208
0      133
11      94
Name: EducationClass, dtype: int64
```

In [152…

```
dfMedTr.EducationClass.hist()
```

Out[152…

```
<AxesSubplot:>
```



Employment

> There are no null values, so we will review the current values.

In [154…

```
dfMedTr.Employment.value_counts()
```

Out[154…

```
Full Time     6029
Student       1017
Part Time      991
Unemployed     983
Retired        980
Name: Employment, dtype: int64
```

> There are no blanks that need to be filled, and I don't believe this would be appropriate to convert to a numeric representation, so we will leave it as is.

Income

> There are nulls present, so we will review the data and make our conclusion on how to handle them.

In [163…

```
dfMedTr.Income.isna().sum()
```

Out[163…

```
2464
```

In [156…

```
dfMedTr.Income.describe()
```

Out[156…

```
count      7536.000000
mean      40484.438268
std       28664.861050
min         154.080000
25%       19450.792500
50%       33942.280000
75%       54075.235000
max      207249.130000
Name: Income, dtype: float64
```

In [159…

```
dfMedTr.Income.mode()
```

Out[159…

```
0    14572.40
1    20474.03
2    26915.85
3    37132.97
4    55506.92
dtype: float64
```

In [160…

```
dfMedTr.Income.median()
```

Out[160…

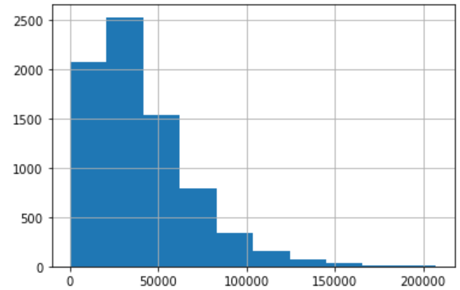
```
33942.28
```

In [158…

```
dfMedTr.Income.hist()
```

Out[158…

```
<AxesSubplot:>
```

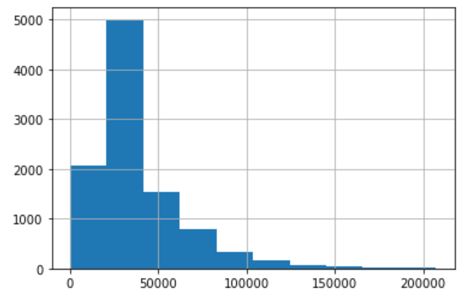


> Since this is a very continous number, there are multiple modes and none of them have very many instances. Due to this, we will use the mean to fill the null values.

In [164…
```
dfMedTr.Income.fillna(dfMedTr.Income.mean(), inplace = True)
```

In [165…
```
dfMedTr.Income.hist()
```

Out[165…
```
<AxesSubplot:>
```



Marital

> There are no null values, so we will review the current values.

In [167…
```
dfMedTr.Marital.value_counts()
```

Out[167…
```
Widowed          2045
Married          2023
Separated        1987
Never Married    1984
Divorced         1961
Name: Marital, dtype: int64
```

> There is no further processing recommended for this field.

ReAdmis

> There are no null values, so we will review the current values.

In [168…
```
dfMedTr.ReAdmis.value_counts()
```

Out[168…
```
No     6331
Yes    3669
Name: ReAdmis, dtype: int64
```

> This is the value we hope to predict. There are no blanks, so no further processing is recommended.

Gender

> There are no null values, so we will review the current values.

In [169…
```
dfMedTr.Gender.value_counts()
```

Out[169…
```
Female                  5018
Male                    4768
Prefer not to answer     214
Name: Gender, dtype: int64
```

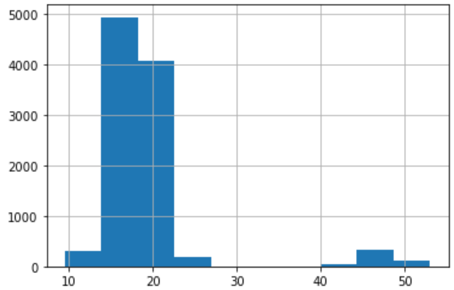
> There is no further processing recommended for this field.

vitD_levels

> There are no null values, so we will review the current values.

In [170…
```
dfMedTr.vitD_levels.hist()
```

Out[170…
```
<AxesSubplot:>
```



> There are no blank values and no outliers (besides a large group of low and a small group of high results). There is no further processing recommended for this field.

doc_visits

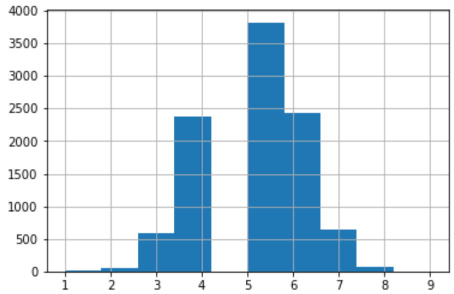
> There are no null values, so we will review the current values.

In [171…

```
dfMedTr.doc_visits.hist()
```

Out[171…

```
<AxesSubplot:>
```



> There is no further processing recommended for this field.

full_meals_eaten

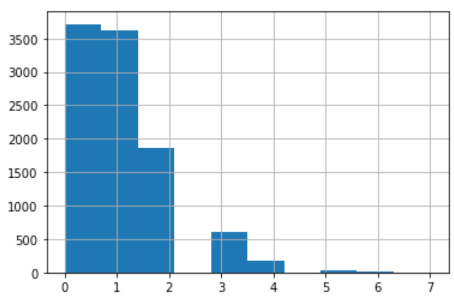
> There are no null values, so we will review the current values.

In [172…

```
dfMedTr.full_meals_eaten.hist()
```

Out[172…

```
<AxesSubplot:>
```



> There is no further processing recommended for this field.

vitD_supp

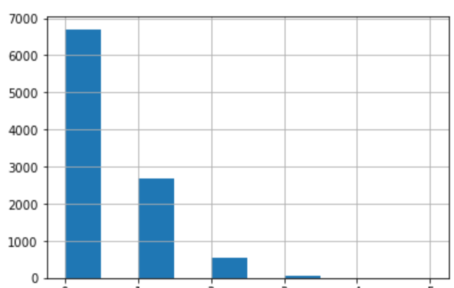
> There are no null values, so we will review the current values.

In [173…

```
dfMedTr.vitD_supp.hist()
```

Out[173…

```
<AxesSubplot:>
```



> There is no further processing recommended for this field.

Soft_drink

> There null values present, so we will review the current values and make a determination on how to proceed.

In [175…

```
dfMedTr.Soft_drink.value_counts()
```

Out[175… No   5589
Yes   1944
Name: Soft_drink, dtype: int64

> Since only half of the population drinks any soda daily, and the average for them is under 3, I believe that we would be safe filling the remaining 2,467 records with "No".

In [176…
```
dfMedTr.Soft_drink.fillna("No", inplace = True)
```

Initial_Admin

> There are no null values, so we will review the current values.

In [177…
```
dfMedTr.Initial_Admin.value_counts()
```

Out[177…
```
Emergency Admission      5060
Elective Admission       2504
Observation Admission    2436
Name: Initial_Admin, dtype: int64
```

> There is no further processing recommended for this field.

HighBlood

> There are no null values, so we will review the current values.

In [178…
```
dfMedTr.HighBlood.value_counts()
```

Out[178…
```
No      5910
Yes     4090
Name: HighBlood, dtype: int64
```

> There is no further processing recommended for this field.

Stroke

> There are no null values, so we will review the current values.

In [179…
```
dfMedTr.Stroke.value_counts()
```

Out[179…
```
No      8007
Yes     1993
Name: Stroke, dtype: int64
```

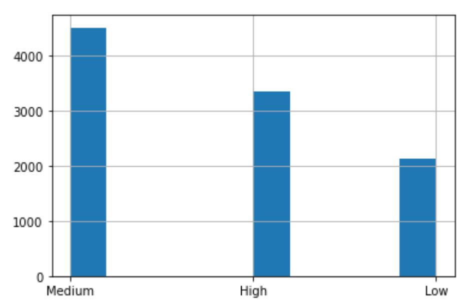
> There is no further processing recommended for this field.

Complication_Risk

> There are no null values, so we will review the current values.

In [180…
```
dfMedTr.Complication_Risk.hist()
```

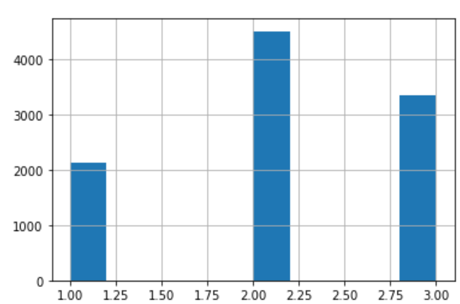
Out[180… <AxesSubplot:>



> Since this is a progression, we will encode this as a number so it can be used in the PCA.

In [181…
```
# create a list of conditions
conditions = [(dfMedTr['Complication_Risk'] == "Low")
              , (dfMedTr['Complication_Risk'] == "Medium")
              , (dfMedTr['Complication_Risk'] == "High")]

# create a list of the values
values = [1,2,3]

dfMedTr['Complication_RiskClass'] = np.select(conditions, values)
dfMedTr.Complication_RiskClass.hist()
```

Out[181… <AxesSubplot:>



Overweight

> There are null values, so we will review and decide how to proceed.

In [183…
```
dfMedTr.Overweight.value_counts()
```

Out[183…
```
1.0    6395
0.0    2623
Name: Overweight, dtype: int64
```

> Since almost 40% of Americans are overweight, we will default the null values to 0.0 so we get closer to the national average.

In [185…
```
dfMedTr.Overweight.fillna(0.0, inplace = True)
```

Arthritis

> There are no null values, so we will review the current values.

In [186…
```
dfMedTr.Arthritis.value_counts()
```

Out[186…
```
No     6426
Yes    3574
Name: Arthritis, dtype: int64
```

> There is no further processing recommended for this field.

Diabetes

> There are no null values, so we will review the current values.

In [187…
```
dfMedTr.Diabetes.value_counts()
```

Out[187…
```
No     7262
Yes    2738
Name: Diabetes, dtype: int64
```

> There is no further processing recommended for this field.

Hyperlipidemia

> There are no null values, so we will review the current values.

In [188…
```
dfMedTr.Hyperlipidemia.value_counts()
```

Out[188…
```
No     6628
Yes    3372
Name: Hyperlipidemia, dtype: int64
```

> There is no further processing recommended for this field.

BackPain

> There are no null values, so we will review the current values.

In [189…
```
dfMedTr.BackPain.value_counts()
```

Out[189…
```
No     5886
Yes    4114
Name: BackPain, dtype: int64
```

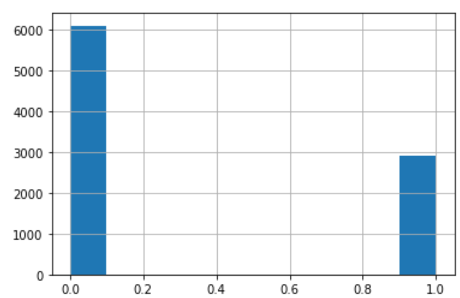
> There is no further processing recommended for this field.

Anxiety

> There are null values, so we will review and decide how to proceed.

In [194…
```
dfMedTr.Anxiety.hist()
```

Out[194…
```
<AxesSubplot:>
```



> Since only 18% of the population has anxiety disorders, we will assume that the null values should be 0 (False).

In [196…
```
dfMedTr.Anxiety.fillna(0.0, inplace = True)
```

Allergic_rhinitis

> There are no null values, so we will review the current values.

In [190…
```
dfMedTr.Allergic_rhinitis.value_counts()
```

Out[190…
```
No     6059
Yes    3941
Name: Allergic_rhinitis, dtype: int64
```

> There is no further processing recommended for this field.

Reflux_esophagitis

> There are no null values, so we will review the current values.

In [191…
```
dfMedTr.Reflux_esophagitis.value_counts()
```

Out[191…
```
No     5865
Yes    4135
Name: Reflux_esophagitis, dtype: int64
```

> There is no further processing recommended for this field.

Asthma

> There are no null values, so we will review the current values.

In [192…
```
dfMedTr.Asthma.value_counts()
```

Out[192…
```
No     7107
Yes    2893
Name: Asthma, dtype: int64
```

> There is no further processing recommended for this field.

Services

> There are no null values, so we will review the current values.

In [193…
```
dfMedTr.Services.value_counts()
```

Out[193…
```
Blood Work     5265
Intravenous    3130
CT Scan        1225
MRI             380
Name: Services, dtype: int64
```

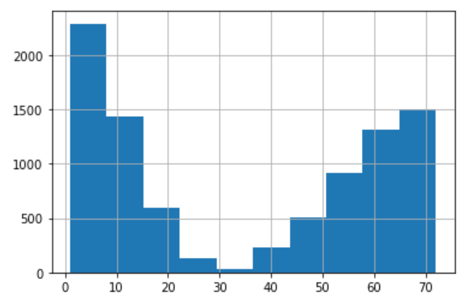
> There is no further processing recommended for this field.

Initial_Days

> There are null values present, so we will review and decide how to proceed.

In [197…
```
dfMedTr.Initial_Days.hist()
```

Out[197…
```
<AxesSubplot:>
```



In [198…
```
dfMedTr.Initial_Days.describe()
```

Out[198…
```
count    8944.000000
mean       34.432082
std        26.287050
min         1.001981
25%         7.911709
50%        34.446941
75%        61.124654
max        71.981486
Name: Initial_Days, dtype: float64
```

In [200…
```
dfMedTr.Initial_Days.median()
```

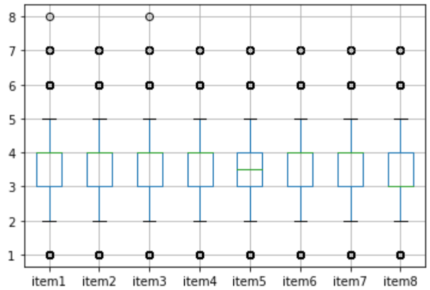
Out[200…
```
34.4469412918404
```

> Since I feel like this would have a large impact (or at least correlation) with readmissions, I don't feel comfortable defaulting this to ANY value. Therefore we will remove the records that are missing this information.

In [202…
```
dfMedTr.dropna(subset = ['Initial_Days'], inplace = True)
```

In [203…
```
## item1-8 - no nulls present, make sure the values all range between 1 and 8
dfMedTr.boxplot(['item1','item2','item3','item4','item5','item6','item7','item8'])
```

Out[203…
```
<AxesSubplot:>
```



In [204…
```
## Remove the columns that we converted to numeric representations
dfMedTr.drop(columns = ['Area','Education','Complication_Risk'], inplace = True)
```

In [205…
```
dfMedTr.shape
```

Out[205…
```
(8944, 41)
```

Data Cleaning Findings

The only real limitation during the cleaning was the empty records for Initial_days. Due to my perceived sensitvity of that value, I don't feel comfortable defaulting to any value. Because of this, our dataset shrank by around 10%.

We will now save the dataset and begin Principal Component Analysis.

In [207…
```
dfMedTr.to_csv('medical_cleaned.csv')
```

E. Apply principal component analysis (PCA) to identify the significant features of the data set by doing the following:

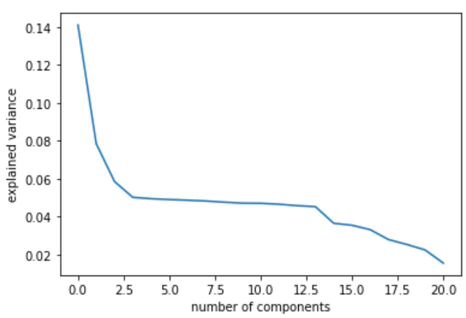
The Prinicpal Components will now be analyzed. They are the continuous columns, that have impact on the question/result. Since we removed the columns that had no impact - we will now grab the subset of continuous columns.

I originally considerd converted the yes/no columns to 0 and 1, but research suggested that binary columns are not best suited by PCA. Therefore I am only including columns that go beyond true/false, 0/1, yes/no.

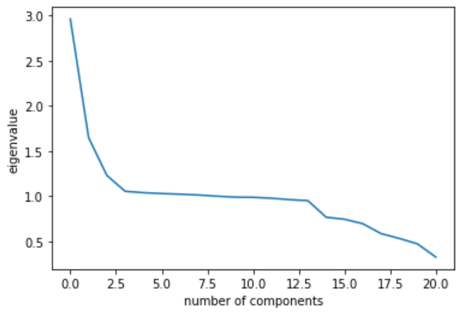
In [219…
```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

dfMedForPCA = dfMedTr[['Lat','Lng','Population','AreaClass','Children','Age','EducationClass'
                      ,'Income','vitD_levels','doc_visits','full_meals_eaten','vitD_supp'
                      ,'Complication_RiskClass','item1','item2','item3'
                      ,'item4','item5','item6','item7','item8']]
dfMedForPCA.shape
```

Out[219…
```
(8944, 21)
```

In [220…
```
# Normalize
dfMedForPCA = (dfMedForPCA - dfMedForPCA.mean()) / dfMedForPCA.std()

# Extracts ALL - shape[1] is the number of columns
pca = PCA(n_components = dfMedForPCA.shape[1])

# Fit and collect data on the PCA
pca.fit(dfMedForPCA)
med_pca = pd.DataFrame(pca.transform(dfMedForPCA)
                      , columns=['PC1','PC2','PC3','PC4','PC5','PC6','PC7','PC8','PC9','PC10'
                                 ,'PC11','PC12','PC13','PC14','PC15','PC16','PC17','PC18'
                                 ,'PC19','PC20','PC21'])

# Display scree plot
plt.plot(pca.explained_variance_ratio_)
plt.xlabel('number of components')
plt.ylabel('explained variance')
plt.show()
```



In [221…
```
# Display eigenvalues scree plot
import numpy as np
cov_matrix = np.dot(dfMedForPCA.T, dfMedForPCA) / dfMedForPCA.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.show()
```



In [222…
```
eigenvalues
```

Out[222…
```
[2.9614769985357228,
 1.6484717185216233,
 1.2297449016612911,
 1.0546414122212788,
 1.0392182980612388,
 1.0299903019017713,
 1.022339896832639,
 1.013976773448205,
 1.000646344372877,
 0.9901484889025732,
 0.9880842493558784,
 0.9780110701140265,
 0.9622439825230933,
 0.9511387091719605,
 0.7674526191124847,
 0.7452758311222754,
 0.6965162244483725,
 0.586770867284266,
 0.5328844918022342,
 0.47236824988594595,
 0.3262506279653295]
```

In [223…
```
# Output loadings
loadings = pd.DataFrame(pca.components_.T
                        , columns=['PC1','PC2','PC3','PC4','PC5','PC6','PC7','PC8','PC9'
                                   ,'PC10','PC11','PC12','PC13','PC14','PC15','PC16','PC17'
                                   ,'PC18','PC19','PC20','PC21']
                        , index=dfMedForPCA.columns)
loadings
```

Out[223…

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13	PC14	PC15
Lat	0.008710	0.001053	-0.715304	0.089132	0.063578	0.023064	0.018933	-0.010820	0.008821	-0.025137	-0.046676	-0.031378	-0.015550	-0.104309	0.133992
Lng	0.002599	-0.002246	0.260457	-0.608438	-0.272427	0.331488	0.162968	0.093598	-0.101845	0.207658	0.234163	-0.219840	-0.160611	0.074451	0.030929
Population	0.006047	0.033015	0.626842	0.320697	0.070157	-0.168930	-0.079989	-0.123143	0.086470	-0.120919	-0.166893	0.093817	0.095939	0.039763	0.138927
AreaClass	-0.003756	-0.006510	0.074807	0.169639	0.209057	-0.433032	0.328055	0.303349	-0.330640	-0.140892	0.293219	-0.267200	-0.467227	-0.156312	0.036233
Children	0.000342	0.021292	0.017054	0.257086	-0.077145	-0.081375	0.295094	0.191204	0.479462	0.731753	0.014675	0.090312	-0.098558	-0.105946	0.035517
Age	-0.001813	0.006600	0.012650	-0.431315	0.045426	-0.280270	0.293836	-0.021883	0.346659	-0.317314	0.171944	0.479808	0.131720	-0.382259	0.037549
EducationClass	0.003588	0.018282	0.069661	0.033212	0.015963	0.375361	-0.337994	0.563164	0.191914	-0.219570	-0.244786	0.164775	-0.406515	-0.286412	0.020758
Income	0.000572	-0.014397	0.053273	0.140167	0.442562	0.295799	-0.092326	0.319888	-0.161890	0.115785	0.547727	0.188573	0.447680	-0.016580	-0.047426
vitD_levels	-0.004683	-0.043230	0.008210	-0.280384	0.217141	-0.219936	0.157065	0.489197	-0.061622	0.058624	-0.539932	-0.255511	0.434064	0.078571	0.012876
doc_visits	0.010128	-0.011262	0.012482	-0.022288	0.429508	0.188729	0.103580	-0.136076	0.613059	-0.254439	0.123814	-0.496581	-0.070714	0.194605	-0.038822
full_meals_eaten	0.001280	0.029860	-0.069722	-0.298452	0.244036	-0.406509	-0.471854	0.088201	0.107038	0.187048	0.117461	0.237392	-0.264963	0.504790	0.067575
vitD_supp	-0.009403	-0.010117	0.000139	0.001196	0.358567	0.322837	0.504950	-0.118652	-0.195394	-0.004928	-0.287593	0.429435	-0.259799	0.349489	0.037743
Complication_RiskClass	0.012459	-0.007663	0.087456	-0.218974	0.498006	0.023913	-0.219267	-0.382003	-0.161390	0.345553	-0.178372	-0.105604	-0.124481	-0.543801	0.007348
item1	0.454195	0.296731	-0.000474	-0.008414	0.003850	-0.005823	0.018951	0.010984	-0.018094	0.007979	-0.016998	-0.004282	0.002516	0.004414	-0.095230
item2	0.428028	0.292373	0.011593	0.016931	-0.003285	0.003366	0.018785	0.026690	0.002208	-0.010926	0.000689	0.005906	-0.002784	-0.006235	-0.147078
item3	0.395270	0.296671	-0.018142	-0.022924	0.020009	-0.012220	0.020247	-0.011939	-0.037360	0.006874	-0.001530	0.015521	-0.017144	-0.006370	-0.201999
item4	0.154121	-0.556868	0.021304	0.007995	-0.016032	-0.026528	0.005212	-0.002799	0.010545	0.030506	-0.014314	0.064536	-0.030295	0.007326	-0.355831
item5	-0.191442	0.582459	-0.004523	-0.013284	-0.001559	0.014899	0.013081	-0.004742	0.002081	0.009426	-0.016473	-0.009450	0.053897	0.016370	0.114282
item6	0.409927	-0.162883	-0.007963	0.008062	-0.005549	-0.019225	-0.017267	0.012464	0.011115	-0.012321	-0.010022	0.001752	-0.003675	0.025952	-0.033337
item7	0.356391	-0.170817	-0.002257	0.002053	-0.010396	-0.008937	-0.012628	-0.024649	0.042887	-0.003275	-0.021053	-0.002067	0.031836	0.009453	0.017928
item8	0.313502	-0.167809	0.009687	-0.005450	-0.019981	0.066011	-0.002476	-0.005962	-0.022958	-0.000974	0.054418	-0.032403	0.044216	-0.002362	0.857201

Conclusion

Based on the eigenvalues (above 1), we should use 9 groupings of the principal components. The fields that have the strongest affect on the 9th grouping the are the following -

- doc_visits: 0.6131
- Children: 0.4795
- Age: 0.3467

This PCA analysis will help us optimize the performance while retaining the most important groupings, and increase accrruacy.

Part IV. Supporting Documents

References

Google Search for Average Number of Children per Household

Google Search for Average Soda Consumption per Day

Google Search for Rate of Anxiety

Stack Overflow Discussion on PCA for Categorical Values

UCertify Course D206 for Data Cleaning