Significate Features That Cause Readmission to the Hospital

Christina LaPane

MSDA Program, Western Governors University

D206: Data Cleaning

Keiona Middleton

Table of Contents

Part I: Research Question

A. Research Question

Is there a way to determine which kind of patient is at risk for readmission to the hospital, within a month of initial hospitalization? If so, which features are most signification to readmission?

B. Variables Used in Data Set

The data file being used is medical_raw_data.csv which includes patient personal information. The data set includes the following information:

- Patient medical conditions (high blood pressure, stroke, diabetes, obesity, etc.)
- Patient information (services during hospitalization, days in hospital, total charges, etc.)
- Patient demographic (age, gender, marital status, employment status, job, etc)
- If patient has been readmitted to the hospital within a month of the initial hospitalization release

The data set includes 10,000 patients and 50 variables. The variables, type of variable, and description is listed below:

Variable	Type	Description	Example
CaseOrder	int64	A placeholder variable to preserve the order of raw data	1-10,000
Customer_id	object	Unique patient ID	C412403
Interaction	object	Notes about patient during visit	Interaction with nurses and doctors
UID	object	Notes about patient during visit	Payments, treatments, etc.
City	object	Patient city	Eva
State	object	Patient state	AL
County	object	Patient county	Morgan
Zip	int64	Patient zip code	35621
Lat	float64	GPS latitude coordinate based on patient address	34.3496
Lng	float64	GPS longitude coordinate based on patient address	-86.72508
Population	int64	Population within a mile of patient address	2951
Area	object	Area of patient address	Rural, Urban, Semi-Urban
Timezone	object	Time zone of patient, or primary insurance holder	America/Chicago
Job	object	Job of patient, or primary insurance holder	Psychologist
Children	float64	Number of children in patient household	1
Age	float64	Age of patient	54
Education	object	Highest earned degree of patient	None, 9th – 12th with no Diploma, High School Diploma, etc..
Employment	object	Employment status of patient	Unemployed, Student, Part-time, Full-time
Income	float64	Annual income of patient, or primary insurance holder	86575.93
Marital	object	Marital status of patient	Never Married, Married, Divorced, Separated, or Widowed

Gender	object	Patient self-identification	Male, Female, or Prefer not to Answer
ReAdmis	object	Patient readmitted within month of initial release	Yes or No
VitD_levels	float64	Patient's vitamin D levels (ng/mL)	17.80233049
Doc_visits	int64	Number of times physician visited patient	6
Full_meals_eaten	int64	Number of meals patient ate. Partial meals count as 0.	0
VitD_supp	int64	The number of times patient given Vitamin D supplements	0
Soft_drink	object	Patient drinks at least 3 soft drinks, daily	Yes or No
Initial_admin	object	Why patient was admitted initially	Emergency, Elective, or Observation
HighBlood	object	Patient has ever had high blood pressure	Yes or No
Stroke	object	Whether the patient has had a stroke	Yes or No
Complication_risk	object	Level of complication risk based on assessment	High, Medium, or Low
Overweight	float64	Patient overweight based off gender, age, and height	1 or 0
Arthritis	object	Patient has arthritis	Yes or No
Diabetes	object	Patient has diabetes	Yes or No
Hyperlipidemia	object	Patient has hyperlipidemia	Yes or No
BackPain	object	Patient has chronic back pain	Yes or No
Anxiety	float64	Patient has an anxiety disorder	1 or 0
Allergic_rhinitis	object	Patient has allergic rhinitis	Yes or No
Reflux_esophagitis	object	Patient has reflux esophagitis	Yes or No
Asthma	object	Patient has asthma	Yes or No
Services	object	Primary service received while hospitalized	Blood work, Intravenous, CT Scan, or MRI
Initial_days	float64	Number of days hospitalized in the initial visit	10.58576971
TotalCharge	float64	The average amount charged to patient daily. Does not include specialized treatments	Total charge / number of days in hospital
Additional_charges	float64	Price of additional charges	Misc. procedures, treatments, medicines, anesthesiology, etc.

Item1 (Timely_admission)	int64	How important is timely admission	Survey response. 1-8. 1=most important, 8= least important
Item2 (Timely_treatment)	int64	How important is timely treatment	Survey response. 1-8. 1=most important, 8= least important
Item3 (Timely_visits)	int64	How important is timely visits	Survey response. 1-8. 1=most important, 8= least important
Item4 (Reliability)	int64	How important is reliability	Survey response. 1-8. 1=most important, 8= least important
Item5 (Options)	int64	How important are options	Survey response. 1-8. 1=most important, 8= least important
Item6 (Hours_of_treatment)	int64	How important is hours of treatment	Survey response. 1-8. 1=most important, 8= least important
Item7 (Courteous_staff)	int64	How important is it to have courteous staff	Survey response. 1-8. 1=most important, 8= least important
Item8 (Doc_active_listening)	int64	How important is it that the doctor is actively listening	Survey response. 1-8. 1=most important, 8= least important

Part II: Data-Cleaning Plan

C. Plan for Cleaning Data

1. Relevant Techniques and Steps Needed to Identify Anomalies

There were plenty of techniques and steps needed to clean medical_raw_data.csv, for it to be useful. To begin the process, the data had to be prepared.

Loading the Data – While loading the data a list of missing values was created to check for null values, values that had white space, a dash, or a period. The index was assigned to the first column, which was CaseOrder, which was simply 1-10,000, the same range of the dataset. The medical_raw_data.csv file was read with the missing values and index as parameters. The dataset was named medical_df.

Evaluate the Data – After the dataset was loaded, the functions .head(), describe.() and .info(), were used to look over the data. .head() shows the first five rows, and from there it showed that 'Interaction' and 'UID' was not needed, so those columns were dropped. The name of the columns 'Item1' through 'Item8' were changed, based off the data dictionary was given with the raw data.

Find Missing Values – Once columns were dropped and named properly, the dataset was searched for missing values, based off the list made earlier. The functions isna() and sum() were used to produce the total number of missing values in each column. This showed all the columns, even the ones with zero, so a for loop was created to show only the columns with missing values, and the percentage of how many values were missing. It ranged from, rounding, 10% to 26%.

Find Outliers – Boxplots, bar graphs, and IQR were used to find outliers of the columns with missing values. Any outliers were removed and set to null so they would not affect the data analysis.

Imputation - Missing values were replaced with meaningful measures of central tendency (mean, median, or mode). In this dataset, mean was used to replace missing values.

2. Justify Approach for Assessing Quality of Data

Mean was used to replace missing values after looking at graphs and finding the IQR. The columns ‘Overweight’ and ‘Anxiety’ only had 0 and 1’s, so finding the mean was a simple solution. Aside from the outliers in ‘Children’, the range in values for each column, were very centralized. After analyzing those factors and also following the steps in the textbook, mean was the best option. (Larose & Larose, 2019)

3. Justify Selected Programming Language and Libraries

Programming Language - Python due to familiarity of the language, the environment, and the packages. The packages are helpful because they contain specially designed code that performs complex data tasks, without having to write out the code. (Larose & Larose, 2019)

<u>Environment</u> - Jupyter Notebook because it is a simple way to show the step-by-step process of cleaning the dataset.

<u>Packages</u> -

- <u>Pandas</u> – The Pandas package is for handling the dataset.
    - import pandas as pd
    - An example of a pandas function is to import the dataset. df = pd.read_csv(file)
- <u>NumPy</u> – The NumPy package is used to work with arrays
    - import numpy as np
    - An example of a numpy function is finding the mean of an entire column. mean = df[‘Col’].mean()
- <u>Sklearn</u> – The Sklearn package is primarily used for the principal component analysis (PCA). (scikit-learn, 2022)
    - from sklearn.decomposition import PCA
    - An example of a sklearn function is fitting and transforming the dataset, meaning it’s reduced to 2D or 3D for better analysis. projected = pca.fit_transform(df)
- <u>Seaborn</u> – The Seaborn package is for high level visualization
    - import seaborn as sns
    - An example of a seaborn is creating a boxplot. ax = sns.boxplot(x=df[‘Col’])

- Matplotlib – The Matplotlib package is for plotting graphs
    - import matplotlib.pyplot as plt
    - An example of a matplotlib function is editing a graph.

      plt.title('Title of Graph) / plt.xlabel('X axis) / plt.ylabel('Y axis)

4. Code to Identify Anomalies

- missing_values = ['N/A', 'NA', 'None', 'n/a', 'na', '-', '.', ' ']
- medical_df = pf.read_csv(file, index_col =[0], na_values = missing_values]
- missing_medical_df = medical_df.isna().sum()

While this is a snippet of the code for identifying anomalies, all the code can be seen multiple ways. It can be seen under the Code section of this paper on pages 25 and 26 . (Figure 2 and 3) There are also two attachments, which have the commented code as well. D206_Data_cleaning.ipynb and D206_Data_Cleaning.pdf.

Part III: Data Cleaning

D. Summarize Data Cleaning Process

1. Findings

There was a good amount of missing data, that would be beneficial to have, in trying to get to the root problem of readmission. All the missing values were numeric except for 'Soft_drink'. After looking at boxplots, mean, and IQR, it seemed reasonable to impute missing values with the mean of each column.

Some data was not important to the analysis, like 'UID' and 'Interaction' as it was a summary of the hospital visit and included notes, payments, and interactions with the doctor. Those two columns were removed at the beginning.

The columns 'CaseOrder' and 'Customer_id' were also not looked at as they were unique values that were used as a key and mapped to each patient's information.

2. Justify Methods for Mitigating Each Type of Anomaly

Detecting Missing Data – Children, Age, Income, Soft_drink, Overweight, Anxiety, and Initial_days contains rows with missing values. All attributes are numbers, except for Soft_drink. The missing values will be corrected with imputation using mean. Before correcting the missing values, outliers will be identified and removed.

Identifying and Removing Outliers – Boxplots were first used to identify outliers, which showed only Children and Income had them. To confirm, the IQR was used. (AskPython,

2022) The describe() method was used to show the standard deviation (std), lower quartile (q1), and upper quartile (q3).

The Interquartile Range (iqr) = q3 – q1

lower limit = q1 – 1.5 * iqr

upper limit = q3 + 1.5 * iqr

The same method and equation were used to determine the outliers for Income. Even though the graph looked like there were outliers, all the data was between the lower and upper limit.

Displaying Unique Values – The CaseOrder and Customer_id are unique values given to each row of data. They are not to be altered.

Detecting Duplicate Values - Due to the unique values, it is confirmed that there are no duplicate values Each column might have duplicate values, but there are no duplicate rows, which is what mattered. There is no need to alter the rows.

3. Outcome from Implementation of Each Data-Cleaning Step

Missing Values -

Children, Age, Income, Soft_drink, Overweight, Anxiety, and Initial_days

‘Children’ – Impute for NaN values

‘Age’ – Impute for NaN values

‘Income’ – Impute for NaN values

‘Soft_drink’ – Change values from NaN to ‘No’

‘Overweight’ – Impute for NaN values

‘Anxiety’ – Impute for NaN values

‘Initial_days’ - Impute for NaN values

Imputations were completed after outliers were identified and removed. (ResearchGate, 2019) Soft_drinks was the only object type and a bar graph was used to see how many Yes and No’s were charted. It was very simple to see that ‘No’ was the median value.

4. **Code to Mitigate Anomalies**

- children_no_null = round(medical_df[‘Children’].mean())
- medical_df[‘Children’].fillna(children_no_null, inplace=True)
- medical_df[‘Soft_drink’].fillna(‘No’, inplace=True)

While this is a snippet of the code for mitigating anomalies, all the code can be seen multiple ways. It can be seen under the Code section of this paper on pages 26 and 27 . (Figure 4 and 5) There are also two attachments, which have the commented code as well. D206_Data_cleaning.ipynb and D206_Data_Cleaning.pdf.

5. **Clean Data Set**

The clean data set was written to a csv file called Clean_orig_data.csv. (Figure 7) Please reference the attached file.

6. Limitations of Data-Cleaning Process

Some limitations of the data cleaning process were that a few columns weren't clearly defined. The Overweight and Anxiety columns were charted as 0,1 but were supposed to be Yes or No. While, in binary, 0 would be No and 1 would be Yes, it's unclear if that was true. If it was backwards, in the raw data, then that can affect the analysis.

The data was gathered instead of being directly from the source, so finding someone to help understand why there were missing values and inconsistent types is not an option. This means that there is no way to find the true data.

7. How Limitations Affect Analysis of Research Question

The limitations can alter the accuracy of the analysis. Imputation is a good way to assume the missing values, but it's not as correct as the real data. Since we cannot get the true data, the factors for readmission that were found, might not be correct. The analysis showed that readmission is dependent on the number of days spent in the initial hospital stay, which was dependent on vitamin D levels. Since the data has missing values and inconsistent types, readmission might be dependent on other factors.

E. Apply Principal Component Analysis (PCA)

1. Principal Components

The principal components in this dataset are (Figure 10) :

- TotalCharge
- Initial_days
- ReAdmis
- HighBlood
- Age

2. How Principal Components Were Identified

Finding the principal components started after the data was cleaned. The cleaned data was written to a csv file and a copy of the dataset was created to preserve the original data. (Figure 6)

Convert Categories to Numeric - The function LabelEncoder() was used to iterate through the copied data and convert all objects to numeric. (Figure 8)

Heatmap - The entire dataset was put into a heatmap to look for any sort of correlation. (seaborn, 2022) After examining the heatmap it showed that there was a high correlation between ‘ReAdmis’ and ‘Initial_days’. ‘Initial_days’ also had a high correlation with ‘TotalCharge’, which had a high correlation with ‘VitD_levels ’. ‘Additional_charges’ had a high correlation with ‘HighBlood’ and ‘Age’. A heatmap including those columns was created to make it clearer. (Graph 4)

Normalize Data – A copy was made off all the important columns, according to the heatmap. ‘Overweight’ and ‘Age’ were also included because their correlation wasn’t high, but it was the next highest correlation to ‘VitD_levels’.

deeper_correlation = copy_df[[‘ReAdmis’ , ‘Initial_days’, ‘TotalCharge’, ‘Additional_charges’, ‘VitD_levels’, ‘Gender’, ‘Age’, ‘Overweight’, ‘Diabetes’, ‘HighBlood’]] (Figure 10)

PCA – After the data was normalized, it was visualized on a Scree Plot(Graph 5). The linear data was used find and plot its eigenvalues, where the graph showed that any more than 6 components would be pointless, since the eigenvalue starts to drop below 1. (Graph 6)

The data was then put into a Component Matrix(Graph 7), where the focus was on the first two columns, PC1 and PC2. In these two columns the attributes that held the most weight were ‘TotalCharge’, ‘Initial_days’, ‘ReAdmis’, ‘Additional_charges’, ‘HighBlood’, and ‘Age’.

The PCA components mirrored the findings in the heatmap and that only six components are pertinent to the analysis.

3. How Organizations Can Benefit from the Results of the PCA

The weight of the six variables suggest that readmission is dependent on a patients age, blood pressure, the number of days initially spent in the hospital, and their charges. It would make since that total and additional charges and initial days are together, because the longer a patient stays, the higher their costs are going to be. With that being said, costs can be grouped into initial stay, which would then give the answer:

Readmission is based off age, blood pressure, and the number of days initially spent in the hospital.

While this narrows down the factors of readmission, more analysis would need to be done to truly answer the question. While common sense tells us that high blood pressure and being older has a higher rate of readmission, it’s unclear about the number of days in their initial stay. The data would need to be examined to see if a patient has a higher chance of being readmitted if they spend less or more time in their initial stay.

Part IV: Supporting Documents

F. Panopto Recording

Web Sources

AskPython. (2022, August). *Detection and Removal of Outliers in Python – An Easy to Understand Guide.* https://www.askpython.com/python/examples/detection-removal-outliers-in-python

References

Larose, C.D. & Larose, D.T. (2019). *Data Science Using Python and R.* John Wiley & Sons, Inc.

scikit-learn. (2022). *sklearn.decomposition.PCA.*

https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

ResearchGate (2019, May) *Data Preprocessing : Missing values or Outliers?*

https://www.researchgate.net/post/Data_Preprocessing_Missing_values_or_outliers

seaborn. (2022) *seaborn.heatmap.*

https://seaborn.pydata.org/generated/seaborn.heatmap.html

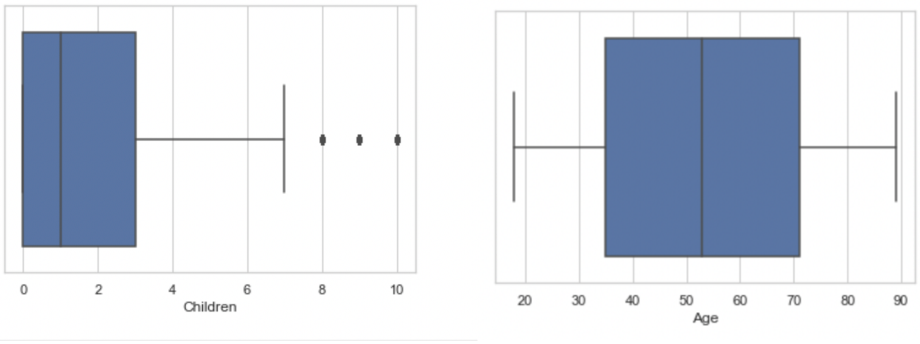
pandas. (2022, August). *User Guide 3.1.4 – Chart Visualization.*

https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html

Graphs

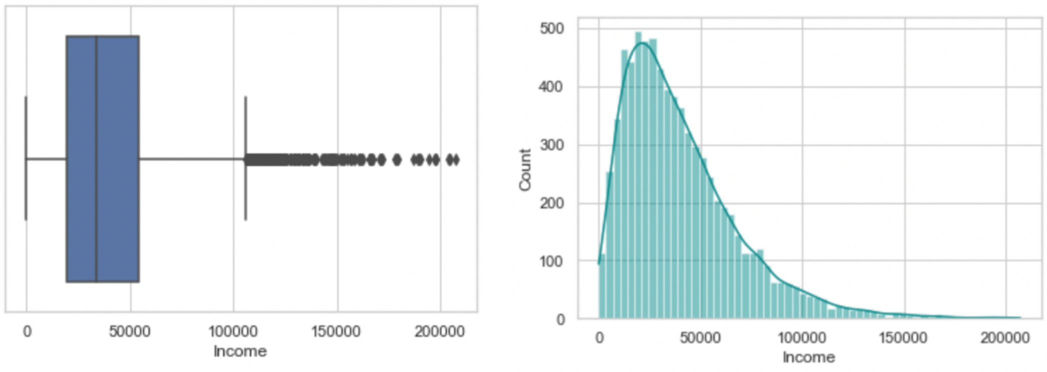
Graph 1

Outliers for Children and Age



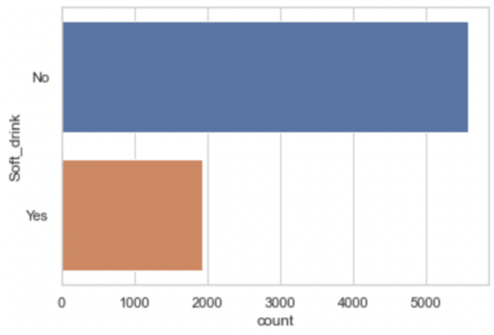
Graph 2

Outliers for Income



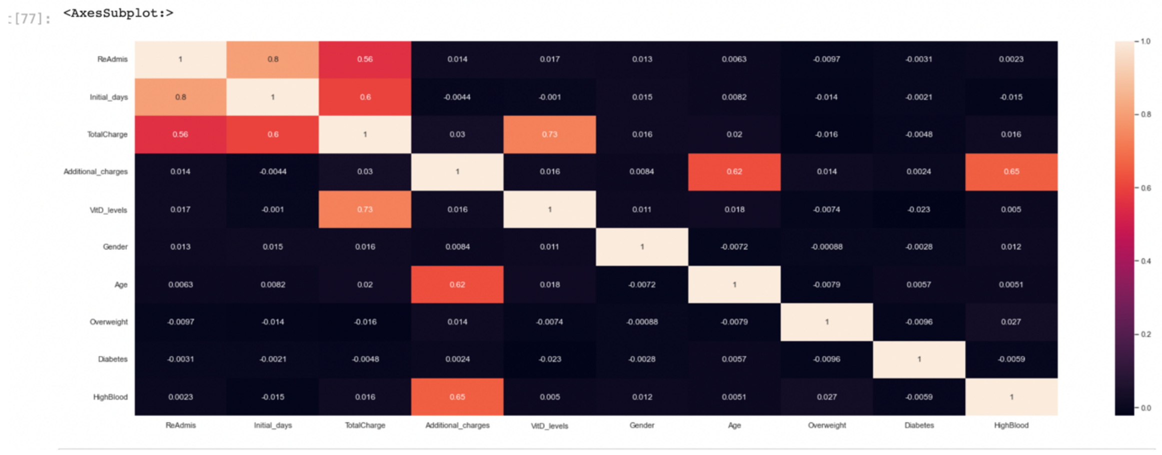
Graph 3

Finding Mean Value for Soft_drink



Graph 4

Heatmap for Principal Components

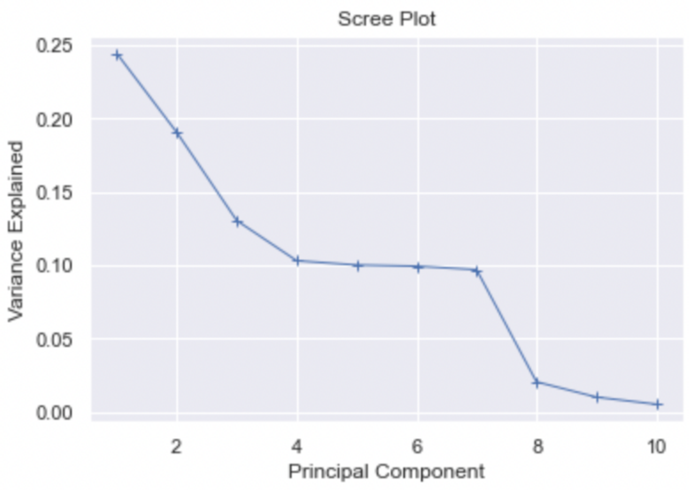


```
:[77]: <AxesSubplot:>
```

	ReAdmis	Initial_days	TotalCharge	Additional_charges	VitD_levels	Gender	Age	Overweight	Diabetes	HighBlood
ReAdmis	1	0.8	0.56	0.014	0.017	0.013	0.0063	-0.0097	-0.0031	0.0023
Initial_days	0.8	1	0.6	-0.0044	-0.001	0.015	0.0082	-0.014	-0.0021	-0.015
TotalCharge	0.56	0.6	1	0.03	0.73	0.016	0.02	-0.016	-0.0048	0.016
Additional_charges	0.014	-0.0044	0.03	1	0.016	0.0084	0.62	0.014	0.0024	0.65
VitD_levels	0.017	-0.001	0.73	0.016	1	0.011	0.018	-0.0074	-0.023	0.005
Gender	0.013	0.015	0.016	0.0084	0.011	1	-0.0072	-0.00088	-0.0028	0.012
Age	0.0063	0.0082	0.02	0.62	0.018	-0.0072	1	-0.0079	0.0057	0.0051
Overweight	-0.0097	-0.014	-0.016	0.014	-0.0074	-0.00088	-0.0079	1	-0.0096	0.027
Diabetes	-0.0031	-0.0021	-0.0048	0.0024	-0.023	-0.0028	0.0057	-0.0096	1	-0.0059
HighBlood	0.0023	-0.015	0.016	0.65	0.005	0.012	0.0051	0.027	-0.0059	1

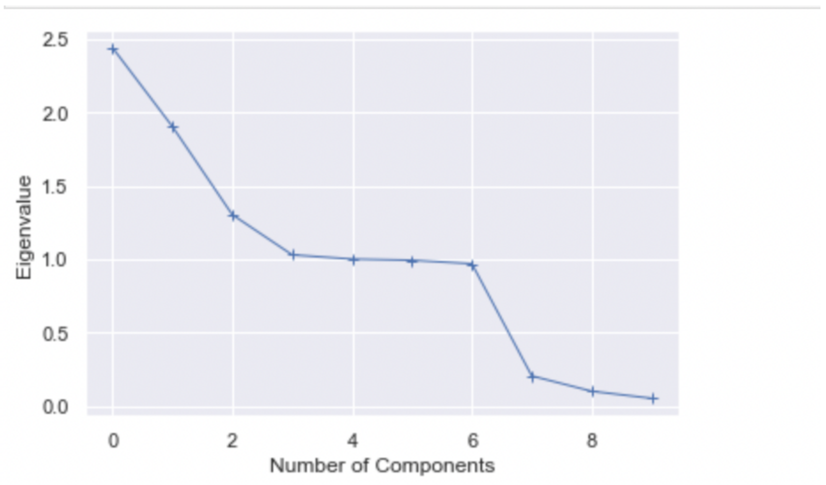
Graph 5

Scree Plot



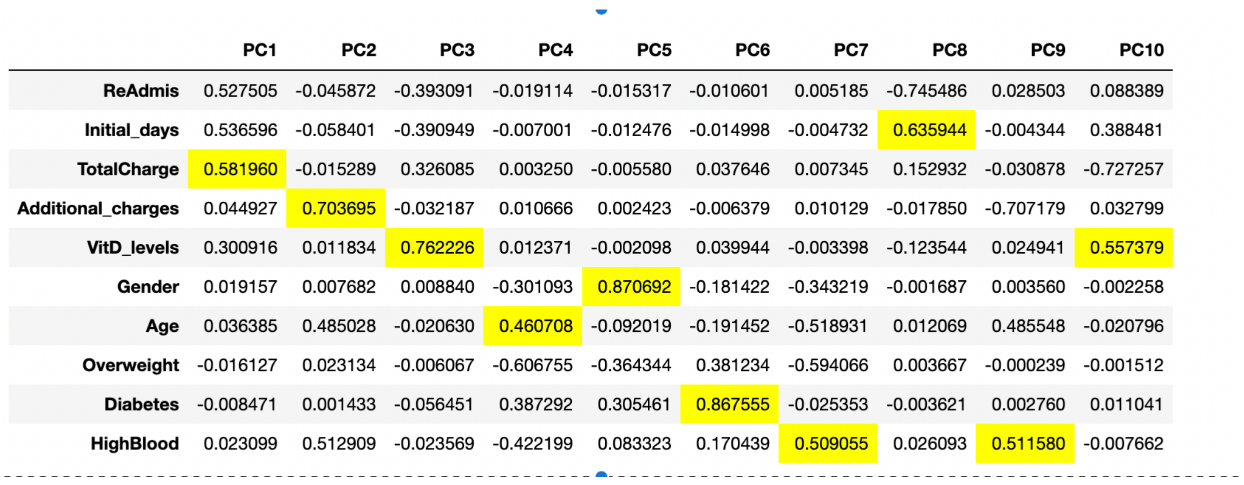
Graph 6

Eigenvalue



Graph 7

Component Matrix

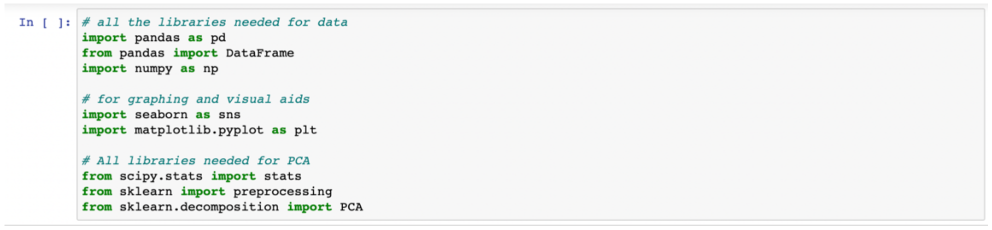


	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
ReAdmis	0.527505	-0.045872	-0.393091	-0.019114	-0.015317	-0.010601	0.005185	-0.745486	0.028503	0.088389
Initial_days	0.536596	-0.058401	-0.390949	-0.007001	-0.012476	-0.014998	-0.004732	0.635944	-0.004344	0.388481
TotalCharge	0.581960	-0.015289	0.326085	0.003250	-0.005580	0.037646	0.007345	0.152932	-0.030878	-0.727257
Additional_charges	0.044927	0.703695	-0.032187	0.010666	0.002423	-0.006379	0.010129	-0.017850	-0.707179	0.032799
VitD_levels	0.300916	0.011834	0.762226	0.012371	-0.002098	0.039944	-0.003398	-0.123544	0.024941	0.557379
Gender	0.019157	0.007682	0.008840	-0.301093	0.870692	-0.181422	-0.343219	-0.001687	0.003560	-0.002258
Age	0.036385	0.485028	-0.020630	0.460708	-0.092019	-0.191452	-0.518931	0.012069	0.485548	-0.020796
Overweight	-0.016127	0.023134	-0.006067	-0.606755	-0.364344	0.381234	-0.594066	0.003667	-0.000239	-0.001512
Diabetes	-0.008471	0.001433	-0.056451	0.387292	0.305461	0.867555	-0.025353	-0.003621	0.002760	0.011041
HighBlood	0.023099	0.512909	-0.023569	-0.422199	0.083323	0.170439	0.509055	0.026093	0.511580	-0.007662

Code

Figure 1

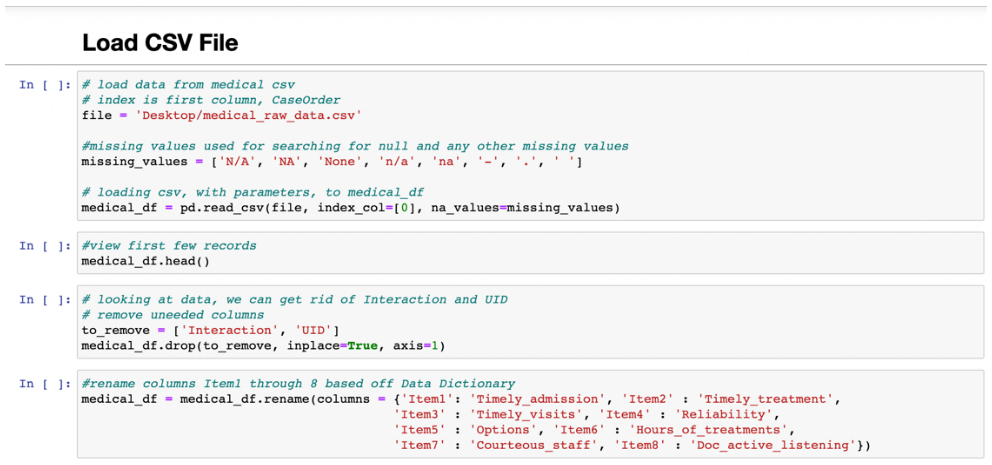
Libraries Used for Data



```
In [ ]: # all the libraries needed for data
        import pandas as pd
        from pandas import DataFrame
        import numpy as np

        # for graphing and visual aids
        import seaborn as sns
        import matplotlib.pyplot as plt

        # All libraries needed for PCA
        from scipy.stats import stats
        from sklearn import preprocessing
        from sklearn.decomposition import PCA
```

Figure 2

Loading and Evaluating the Data

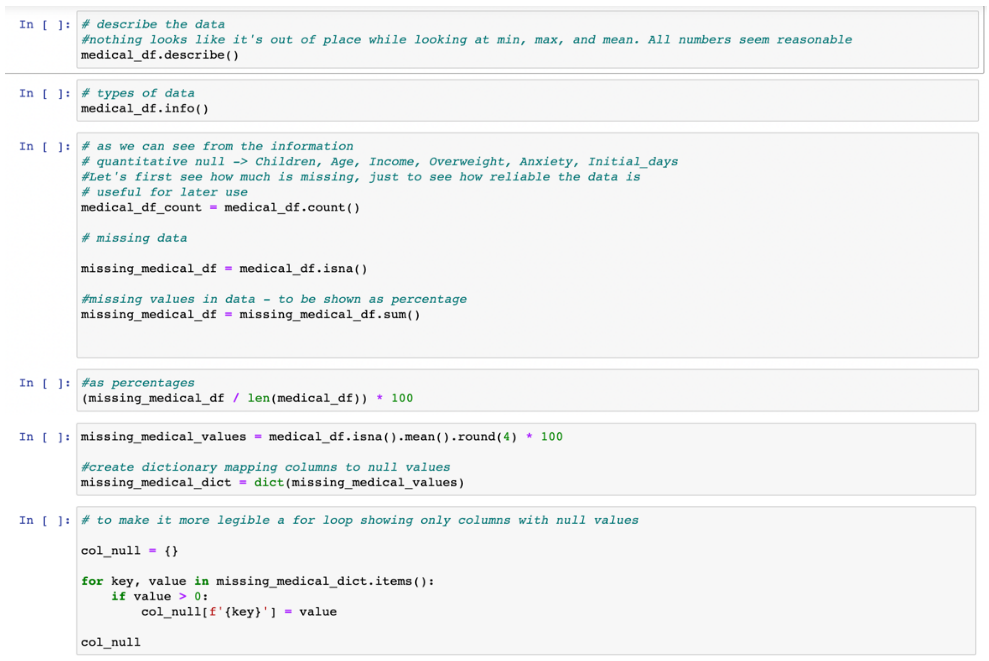


Load CSV File

```
In [ ]: # load data from medical csv
        # index is first column, CaseOrder
        file = 'Desktop/medical_raw_data.csv'

        #missing values used for searching for null and any other missing values
        missing_values = ['N/A', 'NA', 'None', 'n/a', 'na', '-', '.', ' ']

        # loading csv, with parameters, to medical_df
        medical_df = pd.read_csv(file, index_col=[0], na_values=missing_values)

In [ ]: #view first few records
        medical_df.head()

In [ ]: # looking at data, we can get rid of Interaction and UID
        # remove uneeded columns
        to_remove = ['Interaction', 'UID']
        medical_df.drop(to_remove, inplace=True, axis=1)

In [ ]: #rename columns Item1 through 8 based off Data Dictionary
        medical_df = medical_df.rename(columns = {'Item1': 'Timely_admission', 'Item2' : 'Timely_treatment',
                                                  'Item3' : 'Timely_visits', 'Item4' : 'Reliability',
                                                  'Item5' : 'Options', 'Item6' : 'Hours_of_treatments',
                                                  'Item7' : 'Courteous_staff', 'Item8' : 'Doc_active_listening'})
```

Figure 3

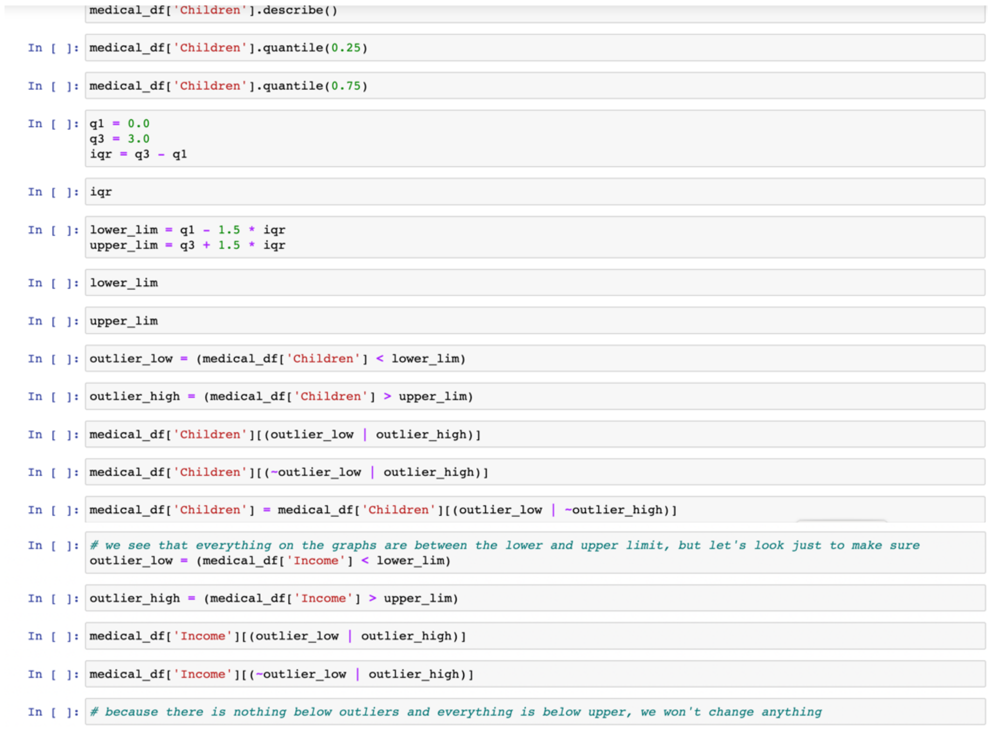
Finding Missing Values



```
In [ ]: # describe the data
        #nothing looks like it's out of place while looking at min, max, and mean. All numbers seem reasonable
        medical_df.describe()

In [ ]: # types of data
        medical_df.info()

In [ ]: # as we can see from the information
        # quantitative null -> Children, Age, Income, Overweight, Anxiety, Initial_days
        #Let's first see how much is missing, just to see how reliable the data is
        # useful for later use
        medical_df_count = medical_df.count()

        # missing data

        missing_medical_df = medical_df.isna()

        #missing values in data - to be shown as percentage
        missing_medical_df = missing_medical_df.sum()

In [ ]: #as percentages
        (missing_medical_df / len(medical_df)) * 100

In [ ]: missing_medical_values = medical_df.isna().mean().round(4) * 100

        #create dictionary mapping columns to null values
        missing_medical_dict = dict(missing_medical_values)

In [ ]: # to make it more legible a for loop showing only columns with null values

        col_null = {}

        for key, value in missing_medical_dict.items():
            if value > 0:
                col_null[f'{key}'] = value

        col_null
```

Figure 4

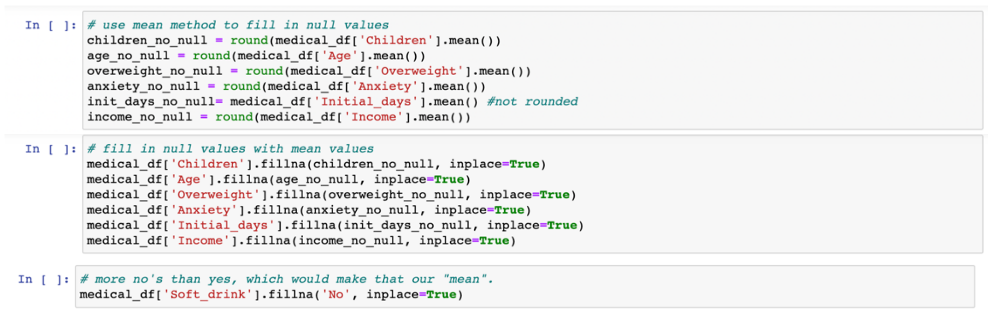
Find Outliers



```
        medical_df['Children'].describe()

In [ ]: medical_df['Children'].quantile(0.25)

In [ ]: medical_df['Children'].quantile(0.75)

In [ ]: q1 = 0.0
        q3 = 3.0
        iqr = q3 - q1

In [ ]: iqr

In [ ]: lower_lim = q1 - 1.5 * iqr
        upper_lim = q3 + 1.5 * iqr

In [ ]: lower_lim

In [ ]: upper_lim

In [ ]: outlier_low = (medical_df['Children'] < lower_lim)

In [ ]: outlier_high = (medical_df['Children'] > upper_lim)

In [ ]: medical_df['Children'][(outlier_low | outlier_high)]

In [ ]: medical_df['Children'][(~outlier_low | outlier_high)]

In [ ]: medical_df['Children'] = medical_df['Children'][(outlier_low | ~outlier_high)]

In [ ]: # we see that everything on the graphs are between the lower and upper limit, but let's look just to make sure
        outlier_low = (medical_df['Income'] < lower_lim)

In [ ]: outlier_high = (medical_df['Income'] > upper_lim)

In [ ]: medical_df['Income'][(outlier_low | outlier_high)]

In [ ]: medical_df['Income'][(~outlier_low | outlier_high)]

In [ ]: # because there is nothing below outliers and everything is below upper, we won't change anything
```

Figure 5

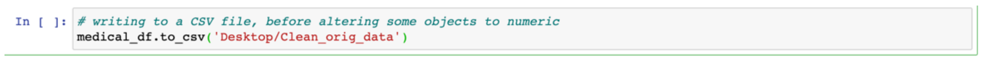
Imputation



```
In [ ]: # use mean method to fill in null values
        children_no_null = round(medical_df['Children'].mean())
        age_no_null = round(medical_df['Age'].mean())
        overweight_no_null = round(medical_df['Overweight'].mean())
        anxiety_no_null = round(medical_df['Anxiety'].mean())
        init_days_no_null= medical_df['Initial_days'].mean() #not rounded
        income_no_null = round(medical_df['Income'].mean())

In [ ]: # fill in null values with mean values
        medical_df['Children'].fillna(children_no_null, inplace=True)
        medical_df['Age'].fillna(age_no_null, inplace=True)
        medical_df['Overweight'].fillna(overweight_no_null, inplace=True)
        medical_df['Anxiety'].fillna(anxiety_no_null, inplace=True)
        medical_df['Initial_days'].fillna(init_days_no_null, inplace=True)
        medical_df['Income'].fillna(income_no_null, inplace=True)

In [ ]: # more no's than yes, which would make that our "mean".
        medical_df['Soft_drink'].fillna('No', inplace=True)
```

Figure 6

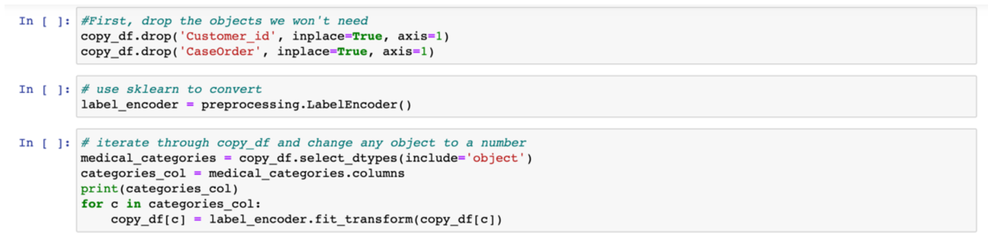
Cleaned Data to CSV



```
In [ ]: # writing to a CSV file, before altering some objects to numeric
        medical_df.to_csv('Desktop/Clean_orig_data')
```

Figure 7

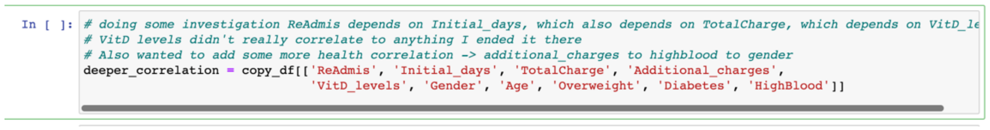
Category to Numeric



```
In [ ]: #First, drop the objects we won't need
        copy_df.drop('Customer_id', inplace=True, axis=1)
        copy_df.drop('CaseOrder', inplace=True, axis=1)

In [ ]: # use sklearn to convert
        label_encoder = preprocessing.LabelEncoder()

In [ ]: # iterate through copy_df and change any object to a number
        medical_categories = copy_df.select_dtypes(include='object')
        categories_col = medical_categories.columns
        print(categories_col)
        for c in categories_col:
            copy_df[c] = label_encoder.fit_transform(copy_df[c])
```

Figure 8

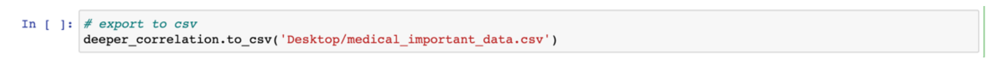
Principal Components



```
In [ ]: # doing some investigation ReAdmis depends on Initial_days, which also depends on TotalCharge, which depends on VitD_le
        # VitD levels didn't really correlate to anything I ended it there
        # Also wanted to add some more health correlation -> additional_charges to highblood to gender
        deeper_correlation = copy_df[['ReAdmis', 'Initial_days', 'TotalCharge', 'Additional_charges',
                                      'VitD_levels', 'Gender', 'Age', 'Overweight', 'Diabetes', 'HighBlood']]
```

Figure 9

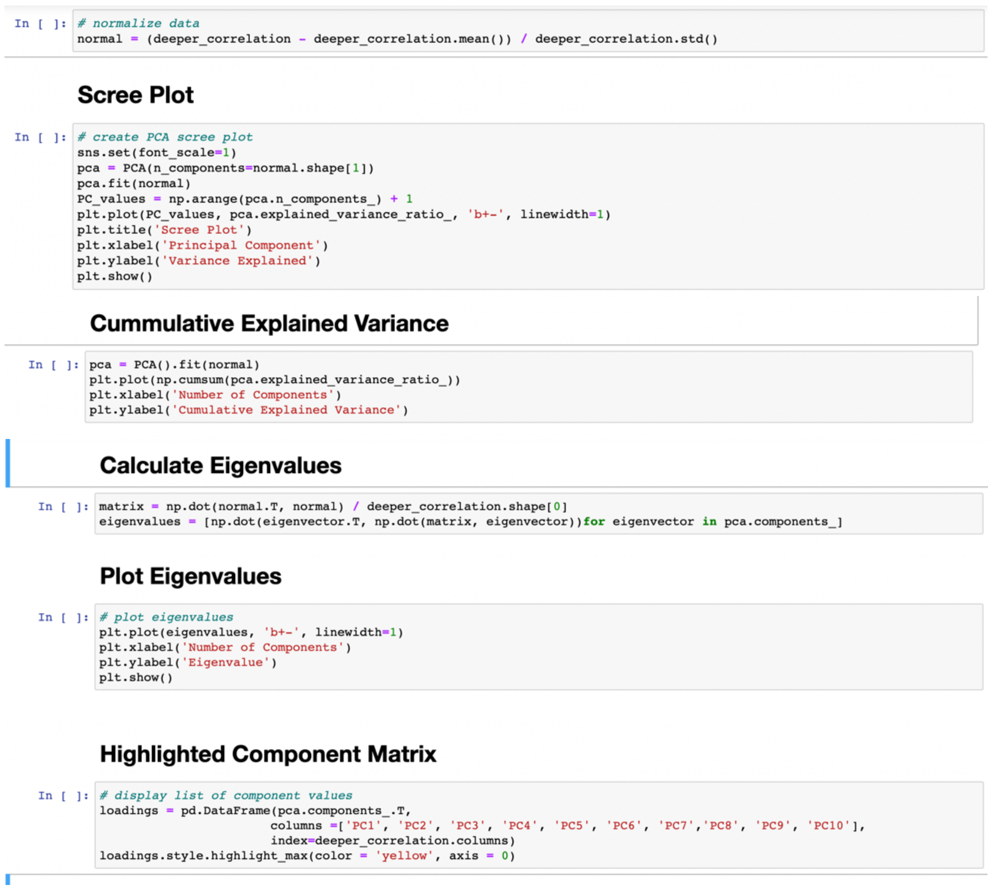
Important Data to CSV



```
In [ ]: # export to csv
        deeper_correlation.to_csv('Desktop/medical_important_data.csv')
```

Figure 10

PCA Process



```
In [ ]: # normalize data
        normal = (deeper_correlation - deeper_correlation.mean()) / deeper_correlation.std()
```

Scree Plot

```
In [ ]: # create PCA scree plot
        sns.set(font_scale=1)
        pca = PCA(n_components=normal.shape[1])
        pca.fit(normal)
        PC_values = np.arange(pca.n_components_) + 1
        plt.plot(PC_values, pca.explained_variance_ratio_, 'b+-', linewidth=1)
        plt.title('Scree Plot')
        plt.xlabel('Principal Component')
        plt.ylabel('Variance Explained')
        plt.show()
```

Cummulative Explained Variance

```
In [ ]: pca = PCA().fit(normal)
        plt.plot(np.cumsum(pca.explained_variance_ratio_))
        plt.xlabel('Number of Components')
        plt.ylabel('Cumulative Explained Variance')
```

Calculate Eigenvalues

```
In [ ]: matrix = np.dot(normal.T, normal) / deeper_correlation.shape[0]
        eigenvalues = [np.dot(eigenvector.T, np.dot(matrix, eigenvector))for eigenvector in pca.components_]
```

Plot Eigenvalues

```
In [ ]: # plot eigenvalues
        plt.plot(eigenvalues, 'b+-', linewidth=1)
        plt.xlabel('Number of Components')
        plt.ylabel('Eigenvalue')
        plt.show()
```

Highlighted Component Matrix

```
In [ ]: # display list of component values
        loadings = pd.DataFrame(pca.components_.T,
                                columns =['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7','PC8', 'PC9', 'PC10'],
                                index=deeper_correlation.columns)
        loadings.style.highlight_max(color = 'yellow', axis = 0)
```