#Import my packages that I'll be using

import numpy as np

import pandas as pd

import seaborn as sns

from pandas import DataFrame

import scipy.stats as stats

from matplotlib import pyplot as plt

%matplotlib inline

#Import my data set, saved as df and run the info function to see the data types

df = pd.read\_csv('medical\_raw\_data.csv')

df.info()

#Using the isnull and sum functions we can see there are 7 columns with missing data

df.isnull().sum()

#Looking at the data types in info, we can see that 6 of the columns contain numeric variables and 1 contains a categorical variable

#For the column Soft\_drink, which contains a categorical variable, we know we're going to use the mode for imputation

#To better see whether to use mean or median for the rest, we're going to create histograms and look at the distributions

df[['Children', 'Age', 'Income', 'Overweight', 'Anxiety', 'Initial\_days']].hist(ec = "black", figsize = (14, 11))

#plt.savefig('Histograms Before Imputation.jpg')

#We can see that Children and Income are both skewed, so we're going to use the median for those two

#For the remaining four I'm going to impute using the mean

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

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

df['Anxiety'].fillna(df['Anxiety'].mean(), inplace = True)

df['Age'].fillna(df['Age'].mean(), inplace = True)

df['Overweight'].fillna(df['Overweight'].mean(), inplace = True)

df['Initial\_days'].fillna(df['Initial\_days'].mean(), inplace = True)

df['Soft\_drink'] = df['Soft\_drink'].fillna(df['Soft\_drink'].mode()[0])

#Now we run isnull().sum() again to verify all the nulls were imputed

df.isnull().sum()

#Now that we no long have any null values, we'll re-create our histograms and check that the distribution hasn't changed much

df[['Children', 'Age', 'Income', 'Overweight', 'Anxiety', 'Initial\_days']].hist(ec = "black", figsize = (14, 11))

#plt.savefig('Histograms After Imputation.jpg')

#To detect outliers I'm going to calculate the z-scores and make histograms of those

#I'm going to omit the anxiety and overweight columns since values that are strictly 0 and 1 won't have outliers

df['Children\_Z\_Scores'] = stats.zscore(df['Children'])

df['Income\_Z\_Scores'] = stats.zscore(df['Income'])

df['Age\_Z\_Scores'] = stats.zscore(df['Age'])

df['Initial\_days\_Z\_Scores'] = stats.zscore(df['Initial\_days'])

df[['Children\_Z\_Scores', 'Age\_Z\_Scores', 'Income\_Z\_Scores', 'Initial\_days\_Z\_Scores']].hist(ec = "black", figsize = (14, 11))

#plt.savefig('Z-Scores.jpg')

#Looks like age and initial\_days have no outliers, but children and income do, so I'll make boxplots for those

boxplot = sns.boxplot(x = 'Children\_Z\_Scores', data = df)

#plt.savefig('Children Boxplot.jpg')

boxplot = sns.boxplot(x = 'Income\_Z\_Scores', data = df)

#plt.savefig('Income Boxplot.jpg')

#With age and income, I'm going to leave the outliers since both of those variables are fairly likely to be correct

#Hospitals could have more young people, then one old person come in, same with income

#Now to extract my clean data set

df.to\_csv('medical\_data\_clean.csv')

#import packages

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.decomposition import PCA

%matplotlib inline

#import clean data set

df = pd.read\_csv('medical\_data\_clean.csv', index\_col = 0)

df = df[['Population', 'Children', 'Age', 'Income', 'Doc\_visits', 'Initial\_days', 'TotalCharge', 'Additional\_charges', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8']]

df.head()

#normalize the data

df\_normalized = (df - df.mean()) / df.std()

pca = PCA(n\_components = df.shape[1])

pca.fit(df\_normalized)

df\_pca = pd.DataFrame(pca.transform(df\_normalized),

columns = ['Population', 'Children', 'Age', 'Income', 'Doc\_visits', 'Initial\_days', 'TotalCharge', 'Additional\_charges', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8'])

#create scree plot

plt.plot(pca.explained\_variance\_ratio\_)

plt.xlabel('number of components')

plt.ylabel('explained variance')

#plt.savefig('PCA Scree Plot.jpg')

plt.show()

#get eigenvalues

cov\_matrix = np.dot(df\_normalized.T, df\_normalized) / df.shape[0]

eigenvalues = [np.dot(eigenvector.T, np.dot(cov\_matrix, eigenvector)) for eigenvector in pca.components\_]

#plot eigenvalues as screen plot

plt.plot(eigenvalues)

plt.xlabel('number of components')

plt.ylabel('eigenvalue')

#plt.savefig('Eigenvalue Plot.jpg')

plt.show()

#generate loading values

loadings = pd.DataFrame(pca.components\_.T,

columns = ['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10', 'PC11', 'PC12', 'PC13', 'PC14', 'PC15', 'PC16'],

index = df.columns)

loadings