**Nefisa Hassen**

**D212 Task 2**

**Part I: Research Question**

A. Describe the purpose of your data mining report by doing the following:

1. Propose one question relevant to a real-world organizational situation that you will answer by using PCA.

Can we reduce the medical dataset by retaining only the features with higher variance?

**2. Define one goal of the data analysis. Ensure your goal is reasonable within the scope of the selected scenario and is represented in the available data.**

The goal of this analysis is to reduce medical datasets to retain only the features that explain most of the variance in the dataset by using PCA.

**Part II: Method Justification**

**B. Explain the reasons for using PCA by doing the following:**

**1. Explain how PCA analyzes the selected data set. Include expected outcomes.**

Principal Component Analysis (PCA) is a dimensionality reduction technique that simplifies complex datasets by transforming them into a smaller set of new variables. These new variables, or principal components, capture most of the original variance in the data, enabling a more manageable and interpretable representation with minimal information loss. The process transforms the data from its original high-dimensional space to a lower-dimensional space, where the selected principal components retain most of the variance from the original dataset, making it easier to analyze.

**2. Summarize one assumption of PCA.**

PCA assumes all variables have linear relationships, showing that the change in one variable is proportional to the change in another. For example, if you plot two variables that have a linear relationship, the data points will tend to cluster around a straight line.

**Part III: Data Preparation**

C. Perform data preparation for the chosen data set by doing the following:

1. Identify the continuous data set variables that you will need to answer the PCA question proposed in part A1.

Here are continues columns that will be used to answer the research question.

‘Lat', 'Lng','Income', 'VitD\_levels', 'Initial\_days', 'TotalCharge', 'Additional\_charges'

The data preparation process is shown below.

**import** pandas **as** pd

2

**import** numpy **as** np

3

**import** matplotlib.pyplot **as** plt

4

**from** scipy **import** stats

5

**from** sklearn.decomposition **import** PCA

6

In [5]:

1

med\_data**=** pd.read\_csv('medical\_clean.csv')

In [6]:

1

med\_data.head()

Out[6]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **CaseOrder** | **Customer\_id** | **Interaction** | **UID** | **City** | **State** | **County** | **Zip** | **Lat** | **Lng** | **...** | **TotalCharge** | **Additional\_charges** | **Item1** | **Item2** | **Item3** | **Item4** | **Item5** | **Item6** | **Item7** | **Item8** |
| **0** | 1 | C412403 | 8cd49b13-f45a-4b47-a2bd-173ffa932c2f | 3a83ddb66e2ae73798bdf1d705dc0932 | Eva | AL | Morgan | 35621 | 34.34960 | -86.72508 | ... | 3726.702860 | 17939.403420 | 3 | 3 | 2 | 2 | 4 | 3 | 3 | 4 |
| **1** | 2 | Z919181 | d2450b70-0337-4406-bdbb-bc1037f1734c | 176354c5eef714957d486009feabf195 | Marianna | FL | Jackson | 32446 | 30.84513 | -85.22907 | ... | 4193.190458 | 17612.998120 | 3 | 4 | 3 | 4 | 4 | 4 | 3 | 3 |
| **2** | 3 | F995323 | a2057123-abf5-4a2c-abad-8ffe33512562 | e19a0fa00aeda885b8a436757e889bc9 | Sioux Falls | SD | Minnehaha | 57110 | 43.54321 | -96.63772 | ... | 2434.234222 | 17505.192460 | 2 | 4 | 4 | 4 | 3 | 4 | 3 | 3 |
| **3** | 4 | A879973 | 1dec528d-eb34-4079-adce-0d7a40e82205 | cd17d7b6d152cb6f23957346d11c3f07 | New Richland | MN | Waseca | 56072 | 43.89744 | -93.51479 | ... | 2127.830423 | 12993.437350 | 3 | 5 | 5 | 3 | 4 | 5 | 5 | 5 |
| **4** | 5 | C544523 | 5885f56b-d6da-43a3-8760-83583af94266 | d2f0425877b10ed6bb381f3e2579424a | West Point | VA | King William | 23181 | 37.59894 | -76.88958 | ... | 2113.073274 | 3716.525786 | 2 | 1 | 3 | 3 | 5 | 3 | 4 | 3 |

5 rows × 50 columns

In [7]:

1

med\_data.shape

Out[7]:

(10000, 50)

In [8]:

1

med\_data.describe()

Out[8]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **CaseOrder** | **Zip** | **Lat** | **Lng** | **Population** | **Children** | **Age** | **Income** | **VitD\_levels** | **Doc\_visits** | **...** | **TotalCharge** | **Additional\_charges** | **Item1** | **Item2** | **Item3** | **Item4** | **Item5** | **Item6** | **Item7** | **Item8** |
| **count** | 10000.00000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | ... | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 |
| **mean** | 5000.50000 | 50159.323900 | 38.751099 | -91.243080 | 9965.253800 | 2.097200 | 53.511700 | 40490.495160 | 17.964262 | 5.012200 | ... | 5312.172769 | 12934.528587 | 3.518800 | 3.506700 | 3.511100 | 3.515100 | 3.496900 | 3.522500 | 3.494000 | 3.509700 |
| **std** | 2886.89568 | 27469.588208 | 5.403085 | 15.205998 | 14824.758614 | 2.163659 | 20.638538 | 28521.153293 | 2.017231 | 1.045734 | ... | 2180.393838 | 6542.601544 | 1.031966 | 1.034825 | 1.032755 | 1.036282 | 1.030192 | 1.032376 | 1.021405 | 1.042312 |
| **min** | 1.00000 | 610.000000 | 17.967190 | -174.209700 | 0.000000 | 0.000000 | 18.000000 | 154.080000 | 9.806483 | 1.000000 | ... | 1938.312067 | 3125.703000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |
| **25%** | 2500.75000 | 27592.000000 | 35.255120 | -97.352982 | 694.750000 | 0.000000 | 36.000000 | 19598.775000 | 16.626439 | 4.000000 | ... | 3179.374015 | 7986.487755 | 3.000000 | 3.000000 | 3.000000 | 3.000000 | 3.000000 | 3.000000 | 3.000000 | 3.000000 |
| **50%** | 5000.50000 | 50207.000000 | 39.419355 | -88.397230 | 2769.000000 | 1.000000 | 53.000000 | 33768.420000 | 17.951122 | 5.000000 | ... | 5213.952000 | 11573.977735 | 4.000000 | 3.000000 | 4.000000 | 4.000000 | 3.000000 | 4.000000 | 3.000000 | 3.000000 |
| **75%** | 7500.25000 | 72411.750000 | 42.044175 | -80.438050 | 13945.000000 | 3.000000 | 71.000000 | 54296.402500 | 19.347963 | 6.000000 | ... | 7459.699750 | 15626.490000 | 4.000000 | 4.000000 | 4.000000 | 4.000000 | 4.000000 | 4.000000 | 4.000000 | 4.000000 |
| **max** | 10000.00000 | 99929.000000 | 70.560990 | -65.290170 | 122814.000000 | 10.000000 | 89.000000 | 207249.100000 | 26.394449 | 9.000000 | ... | 9180.728000 | 30566.070000 | 8.000000 | 7.000000 | 8.000000 | 7.000000 | 7.000000 | 7.000000 | 7.000000 | 7.000000 |

8 rows × 23 columns

In [9]:

1

med\_data.columns *#looking at the columns*

2

Out[9]:

Index(['CaseOrder', 'Customer\_id', 'Interaction', 'UID', 'City', 'State',  
 'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job',  
 'Children', 'Age', 'Income', 'Marital', 'Gender', 'ReAdmis',  
 'VitD\_levels', 'Doc\_visits', 'Full\_meals\_eaten', 'vitD\_supp',  
 'Soft\_drink', 'Initial\_admin', 'HighBlood', 'Stroke',  
 'Complication\_risk', 'Overweight', 'Arthritis', 'Diabetes',  
 'Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic\_rhinitis',  
 'Reflux\_esophagitis', 'Asthma', 'Services', 'Initial\_days',  
 'TotalCharge', 'Additional\_charges', 'Item1', 'Item2', 'Item3', 'Item4',  
 'Item5', 'Item6', 'Item7', 'Item8'],  
 dtype='object')

In [10]:

1

med\_data.isnull().sum() *# checking for missing data*

Out[10]:

CaseOrder 0  
Customer\_id 0  
Interaction 0  
UID 0  
City 0  
State 0  
County 0  
Zip 0  
Lat 0  
Lng 0  
Population 0  
Area 0  
TimeZone 0  
Job 0  
Children 0  
Age 0  
Income 0  
Marital 0  
Gender 0  
ReAdmis 0  
VitD\_levels 0  
Doc\_visits 0  
Full\_meals\_eaten 0  
vitD\_supp 0  
Soft\_drink 0  
Initial\_admin 0  
HighBlood 0  
Stroke 0  
Complication\_risk 0  
Overweight 0  
Arthritis 0  
Diabetes 0  
Hyperlipidemia 0  
BackPain 0  
Anxiety 0  
Allergic\_rhinitis 0  
Reflux\_esophagitis 0  
Asthma 0  
Services 0  
Initial\_days 0  
TotalCharge 0  
Additional\_charges 0  
Item1 0  
Item2 0  
Item3 0  
Item4 0  
Item5 0  
Item6 0  
Item7 0  
Item8 0  
dtype: int64

In [11]:

1

2

med\_data.duplicated().any() *# checking for duplicates*

Out[11]:

False

In [12]:

1

df\_numeric **=** med\_data.select\_dtypes(exclude**=**['object', 'category'])

In [48]:

1

cont\_df **=** df\_numeric.drop(['CaseOrder', 'Zip','Children','Doc\_visits','Full\_meals\_eaten','vitD\_supp','Age','Population','Item1', 'Item2',

2

'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8'],axis **=**1)

3

4

cont\_df.columns

Out[48]:

Index(['Lat', 'Lng', 'Income', 'VitD\_levels', 'Initial\_days', 'TotalCharge',  
 'Additional\_charges'],  
 dtype='object')

In [49]:

1

cont\_df.head()*# all columns are continuous*

Out[49]:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Lat** | **Lng** | **Income** | **VitD\_levels** | **Initial\_days** | **TotalCharge** | **Additional\_charges** |
| **0** | 34.34960 | -86.72508 | 86575.93 | 19.141466 | 10.585770 | 3726.702860 | 17939.403420 |
| **1** | 30.84513 | -85.22907 | 46805.99 | 18.940352 | 15.129562 | 4193.190458 | 17612.998120 |
| **2** | 43.54321 | -96.63772 | 14370.14 | 18.057507 | 4.772177 | 2434.234222 | 17505.192460 |
| **3** | 43.89744 | -93.51479 | 39741.49 | 16.576858 | 1.714879 | 2127.830423 | 12993.437350 |
| **4** | 37.59894 | -76.88958 | 1209.56 | 17.439069 | 1.254807 | 2113.073274 | 3716.525786 |

In [50]:

1

cont\_df.shape

Out[50]:

(10000, 7)

In [54]:

1

**from** scipy **import** stats

In [55]:

1

cont\_df.std() *# checking for outliers*

Out[55]:

Lat 5.403085  
Lng 15.205998  
Income 28521.153293  
VitD\_levels 2.017231  
Initial\_days 26.309341  
TotalCharge 2180.393838  
Additional\_charges 6542.601544  
dtype: float64

In [56]:

1

2

**from** scipy.stats **import** zscore

In [57]:

1

*# identifying and treating outliers*

2

3

*# Calculate Z-scores*

4

z\_scores **=** cont\_df.apply(zscore)

5

6

*# Filter out rows where any column's Z-score is greater than 3 or less than -3*

7

clean\_data **=**cont\_df [(z\_scores.abs() **<** 3).all(axis**=**1)]

8

9

print(clean\_data)

Lat Lng Income VitD\_levels Initial\_days TotalCharge \  
0 34.34960 -86.72508 86575.93 19.141466 10.585770 3726.702860   
1 30.84513 -85.22907 46805.99 18.940352 15.129562 4193.190458   
2 43.54321 -96.63772 14370.14 18.057507 4.772177 2434.234222   
3 43.89744 -93.51479 39741.49 16.576858 1.714879 2127.830423   
4 37.59894 -76.88958 1209.56 17.439069 1.254807 2113.073274   
... ... ... ... ... ... ...   
9995 36.42886 -78.23716 45967.61 16.980860 51.561220 6850.942000   
9996 39.43609 -74.87302 14983.02 18.177020 68.668240 7741.690000   
9997 36.36655 -87.29988 65917.81 17.129070 70.154180 8276.481000   
9998 44.10354 -102.01590 29702.32 19.910430 63.356900 7644.483000   
9999 40.49998 -80.19959 62682.63 18.388620 70.850590 7887.553000   
  
 Additional\_charges   
0 17939.403420   
1 17612.998120   
2 17505.192460   
3 12993.437350   
4 3716.525786   
... ...   
9995 8927.642000   
9996 28507.150000   
9997 15281.210000   
9998 7781.678000   
9999 11643.190000   
  
[9688 rows x 7 columns]

In [58]:

1

**C2. Standardize the continuous data set variables identified in part C1. Include a copy of the cleaned data set.**

**from** sklearn.preprocessing **import** StandardScaler

2

**from** sklearn.decomposition **import** PCA

3

In [59]:

1

*#normalizing data*

In [60]:

1

scaler **=** StandardScaler()

2

norm\_df **=** scaler.fit\_transform(clean\_data)

3

In [61]:

1

scaled\_df **=** pd.DataFrame(norm\_df, columns **=** clean\_data.columns)

In [62]:

1

scaled\_df.head()

Out[62]:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Lat** | **Lng** | **Income** | **VitD\_levels** | **Initial\_days** | **TotalCharge** | **Additional\_charges** |
| **0** | -0.920719 | 0.287065 | 1.874558 | 0.593823 | -0.907451 | -0.727175 | 0.762414 |
| **1** | -1.654735 | 0.396236 | 0.309406 | 0.492788 | -0.734780 | -0.513375 | 0.712571 |
| **2** | 1.004896 | -0.436304 | -0.967112 | 0.049268 | -1.128376 | -1.319537 | 0.696108 |
| **3** | 1.079090 | -0.208410 | 0.031381 | -0.694575 | -1.244558 | -1.459968 | 0.007145 |
| **4** | -0.240140 | 1.004806 | -1.485049 | -0.261421 | -1.262041 | -1.466731 | -1.409476 |

In [63]:

1

scaled\_df.shape

Out[63]:

(9688, 7)

In [64]:

**1**

**a copy of the cleaned data set**

scaled\_df.to\_csv ('scaled\_df.csv')

**Part IV: Analysis**

D. Perform PCA by doing the following:

1. Determine the matrix of *all* the principal components.

2. Identify the *total* number of principal components, using the elbow rule or the Kaiser criterion. Include a screenshot of the scree plot.

3. Identify the variance of *each* of the principal components identified in part D2.

4. Identify the *total* variance captured by the principal components identified in part D2.

*#Analyzing data*

pca**=** PCA()

norm\_data **=** pca.fit\_transform(scaled\_df)

leading\_matrix **=** pd.DataFrame(pca.components\_,columns**=** scaled\_df.columns, index **=** ['PC1','PC2','PC3','PC4','PC5','PC6','PC7'])

2

leading\_matrix

Out[69]:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Lat** | **Lng** | **Income** | **VitD\_levels** | **Initial\_days** | **TotalCharge** | **Additional\_charges** |
| **PC1** | -0.019689 | -0.004479 | -0.015216 | -0.001225 | 0.706650 | 0.706909 | 0.016859 |
| **PC2** | -0.067322 | -0.353084 | 0.658041 | -0.598889 | 0.012834 | 0.002886 | -0.280964 |
| **PC3** | -0.784544 | 0.487291 | 0.209308 | -0.110754 | -0.015497 | -0.006140 | 0.301136 |
| **PC4** | -0.060625 | -0.595006 | 0.183826 | 0.206375 | -0.019223 | 0.000136 | 0.752024 |
| **PC5** | -0.235278 | -0.122586 | 0.417641 | 0.757552 | 0.011201 | 0.001928 | -0.425652 |
| **PC6** | 0.566158 | 0.518459 | 0.561002 | 0.112188 | 0.009628 | 0.014826 | 0.288173 |
| **PC7** | 0.001928 | -0.000272 | 0.001407 | -0.001750 | -0.706861 | 0.707114 | -0.018120 |

In [70]:

1

*determine the components with highest variance using the kaiser plot*

In [40]:

1

pccomp **=** np.arange(pca.n\_components\_)**+**1

2

pccomp

Out[40]:

array([1, 2, 3, 4, 5, 6, 7])

In [30]:

1

*#kaiser rule*

In [41]:

1

var **=** pca.explained\_variance\_

2

var

Out[41]:

array([1.98880439, 1.02481587, 1.01561826, 1.00338778, 0.98471515,  
 0.97139656, 0.01198459])

In [32]:

1

plt.figure(figsize**=**(13, 6))

2

plt.plot(pccomp, var, 'b-')

3

plt.title('Scree Plot', fontsize**=**16)

4

plt.xlabel('Number of Components', fontsize**=**16)

5

plt.ylabel('Explained Variance Ratio', fontsize**=**16)

6

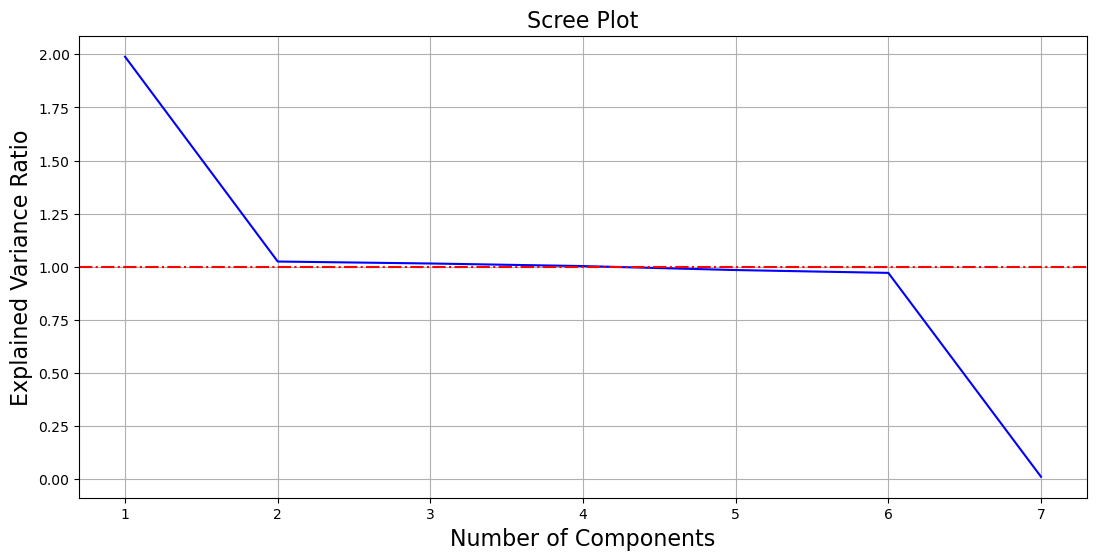
plt.axhline( y**=**1,color **=** 'r',linestyle **=** 'dashdot')

7

plt.grid(**True**)

8

plt.show()



In [36]:

1

print (dict(zip(['PC1','PC2','PC3','PC4'],pccomp)))

{'PC1': 1, 'PC2': 2, 'PC3': 3, 'PC4': 4}

In [39]:

1

print('Variance of the first four principal components:')

2

print (pca.explained\_variance\_[:4])

Variance of the first four principal components:  
[1.98880439 1.02481587 1.01561826 1.00338778]

In [59]:

1

*#The percentage of explained variance for the significant principal component captured after the Kaier plot*

2

3

explained\_var **=** pca.explained\_variance\_ratio\_[:4] **\*** 100

4

explained\_var

Out[59]:

array([28.4085587 , 14.63871551, 14.50733473, 14.33263163])

In [60]:

1

*#Total variance*

2

total\_var\_captured**=** np.sum(explained\_var)

3

total\_var\_captured

Out[60]:

71.88724057370509

**5. Summarize the results of your data analysis.**

After cleaning and scaling the data, PCA was applied to the dataset. The principal component loading matrix was then generated, showing how each original feature contributes to the principal components (PCs). The variance explained by each principal component was extracted to indicate how much of the total variance each component captures.

Based on the Kaiser plot, the first four principal components were retained, as they have an explained variance greater than 1. These four components together account for approximately 71% of the variability in the original dataset, with the percentages of explained variance being 28.41%, 14.64%, 14.51%, and 14.33%, respectively. This high level of explained variance suggests that these components provide an effective summary of the data with minimal information loss, making them ideal for further analysis.

**Part V: Attachments**

E. Record the web sources used to acquire data or segments of third-party code to support the analysis. Ensure the web sources are reliable.

F. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

* GeeksforGeeks. (n.d.). *Dimensionality reduction*. Retrieved August 5, 2024, from <https://www.geeksforgeeks.org/dimensionality-reduction/>
* Marcell, A. (2021, April 4). *Project 3: Analyzing cumulative variance explained with a line plot*. Medium. Retrieved August 5, 2024, from <https://medium.com/@averymarcell/project-3-analyzing-cumulative-variance-explained-with-a-line-plot-a28ba3fbd120>
* Codatalicious. (2020, December 28). *Limitations, assumptions, and watch-outs of principal component analysis*. Medium. Retrieved August 5, 2024, from <https://codatalicious.medium.com/limitations-assumptions-watch-outs-of-principal-component-analysis-8483ceaa2800>
* Bioturing. (2021, August 1). *How to read PCA biplots and scree plots*. Medium. Retrieved August 5, 2024, from <https://bioturing.medium.com/how-to-read-pca-biplots-and-scree-plots-186246aae063#:~:text=Figure%204.&text=A%20scree%20plot%20shows%20how,least%2080%25%20of%20the%20variance>.
* Mangale, S. (2021, November 10). *Scree plot*. Medium. Retrieved August 5, 2024, from <https://sanchitamangale12.medium.com/scree-plot-733ed72c8608>

G. Demonstrate professional communication in the content and presentation of your submission.

2