WGU D212 Task2

July 4, 2024

[309]:
```
%%html
<style>
.toc-item > li {
    list-style-type: upper-alpha;
}
</style>
```

```
<IPython.core.display.HTML object>
```

0.1 Kamal Shaham

0.2 D212 Task 2: Dimension Reduction Methods

```
<h2>Table of Contents</h2>
<ul class="toc-item">
    <li><a href="#research-question">Research Question</a>
        <ul>
            <li><a href="#goal">Analysis Goal</a></li>
        </ul>
    </li>
    <li><a href="#justification">Principal Component Analysis</a>
        <ul>
            <li><a href="#assumption">Assumption of PCA</a></li>
        </ul>
    </li>
     <li><a href="#variables">Data Preparation</a>
        <ul>
            <li><a href="#standardized">Standardized Variables</a></li>
        </ul>
    </li>
    <li><a href="#matrix">PC Matrix</a>
    <ul>
            <li><a href="#components">Principal Components</a></li>
            <li><a href="#variances">Variances</a></li>
            <li><a href="#total-variance">Total Variance</a></li>
        <li><a href="#results">Analysis Results Summary</a></li>
        </ul>
    </li>
    <li><a href="#thirdparty">Third-party code references</a></li>
```

```
    <li><a href="#references">References</a></li>
</ul>
```

0.3 A. Research Question

The research question presented for this analysis is whether Principal Component Analysis (PCA) can be used to reduce the dimensionality of the medical dataset while still maintaining meaningful patient information. Understanding patient characteristics is an important part of any hospital's efforts to reduce hospital readmissions.

0.3.1 A2. Analysis Goal

A goal of this analysis is to reduce the dimensionality of the medical dataset to understand the remaining variables that explain the most variability. According to an article by Jaadi (2024), reducing the variables of a dataset loses some accuracy but increases simplicity. By making the dataset smaller, it normally becomes easier to explore and visualize. The goal of reducing the medical dataset is to make it easier to analyze while retaining the most significant information for this analysis. Understanding this significant patient information can be important to hospitals that are attempting to reduce their patient readmission rates.

0.4 B. Principal Component Analysis

As mentioned earlier, Principal Component Analysis (PCA) will be used to analyze the dataset. Principal components are new variables created as linear combinations or mixtures of the initial variables (Jaadi, 2024). While these combinations are mixtures of the initial variables, they are not correlated since most of the information from the initial variables is compressed into the first components.

PCA comprises the following steps after loading the medical dataset:

- Standardizing the continuous initial variables of the medical dataset allows them to contribute equally to the analysis. This is necessary because PCA is sensitive to the variance of the initial variables, meaning larger differences in the ranges of the variables may bias the analysis results.
- Computing the covariance matrix to understand the variable correlation. This is important because sometimes variables may be highly correlated, thus containing redundant information. The covariance matrix allows the analyst to view these correlations.
- Computing the initial principal components based on the features from the covariance matrix. With redundant variables removed, the principal components will be computed based on their order of explained variance.
- Using the components with the highest explained variance with the elbow rule to explain at least 80% of the variance. This step allows for filtering out components beyond the 80% variance threshold, reducing the dimensionality of the dataset by selecting only the most important components.

After following the above steps to perform the PCA, the results will be visualized, and the variance of each principal component will be identified. The expected outcome of this process is to view the principal components and their values.

0.4.1 B2. Assumption of Principal Component Analysis

An assumption of PCA is the standardization of the initial variables. PCA is sensitive to the scale of features, meaning differences in ranges may bias the PCA to focus on features with higher values if their ranges are higher than those of other features. It is expected that the initial variables will be standardized to mitigate this sensitivity.

0.5 C. Initial Variables

The continuous variables identified for this analysis are displayed below. Some variables such as population may not be deemed meaningful information, however, the PCA aims to reduce the dimensions of the dataset allowing for less meaningful features to be removed.

- Population - The population within a mile radius of the patient
- Children - Number of children living in the patient's household
- Age - Patient's age at the time of admission
- Income - Annual income of the patient/primary insurance holder
- VitD_levels - Patient's vitamin D levels
- Doc_visits - Number of times the primary physician visited the patient while in the hospital
- Full_meals_eaten - Number of full meals the patient ate in the hospital
- vitD_supp - The number of times that vitamin D supplements were administered to the patient
- Initial_days - The number of days the patient stayed in the hospital during the initial visit
- TotalCharge - The amount charged to the patient daily
- Additional_charges - The average amount charged to the patient for miscellaneous hospital services and medications.

0.5.1 C2. Standardized Variables

As in previous courses, the dataset was imported and checked for null and duplicate values. Summary statistics, column information, and row information were also outputted. The 11 features mentioned above were selected and standardized. The summary statistics were then displayed for the newly standardized variables to show their means and standard deviations. As mentioned in section B1, a covariance matrix was computed to understand the variable correlations.

[315]:
```
%matplotlib inline

# importing our statistical libraries
import pandas as pd

# importing our initial dataset
wgu=pd.read_csv('medical_clean.csv')

#viewing first 5 rows and column information
print(wgu.head())
print(wgu.columns)

#checking for missing/null values
print(wgu.isnull().sum())
```

```python
#checking for duplicate values of any rows
print(wgu.duplicated().any())

# checking for duplicate values based on customer_id unique key
print(wgu.duplicated('Customer_id').any())

# view new dataset after selecting needed columns
print(wgu.head())
print(wgu.columns)
print(wgu.info())

# variable statistics to check distributions
print(wgu.describe(include='all'))
```

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

                                UID         City State        County    Zip  \
0  3a83ddb66e2ae73798bdf1d705dc0932          Eva    AL        Morgan  35621
1  176354c5eef714957d486009feabf195     Marianna    FL       Jackson  32446
2  e19a0fa00aeda885b8a436757e889bc9  Sioux Falls    SD     Minnehaha  57110
3  cd17d7b6d152cb6f23957346d11c3f07  New Richland    MN        Waseca  56072
4  d2f0425877b10ed6bb381f3e2579424a   West Point    VA  King William  23181

        Lat       Lng  …  TotalCharge Additional_charges Item1 Item2  Item3  \
0  34.34960 -86.72508  …  3726.702860       17939.403420     3     3      2
1  30.84513 -85.22907  …  4193.190458       17612.998120     3     4      3
2  43.54321 -96.63772  …  2434.234222       17505.192460     2     4      4
3  43.89744 -93.51479  …  2127.830423       12993.437350     3     5      5
4  37.59894 -76.88958  …  2113.073274        3716.525786     2     1      3

   Item4  Item5 Item6 Item7 Item8
0      2      4     3     3     4
1      4      4     4     3     3
2      4      3     4     3     3
3      3      4     5     5     5
4      3      5     3     4     3

[5 rows x 50 columns]
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')
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
False
False
   CaseOrder Customer_id                           Interaction  \
0          1     C412403  8cd49b13-f45a-4b47-a2bd-173ffa932c2f
1          2     Z919181  d2450b70-0337-4406-bdbb-bc1037f1734c
2          3     F995323  a2057123-abf5-4a2c-abad-8ffe33512562
3          4     A879973  1dec528d-eb34-4079-adce-0d7a40e82205
4          5     C544523  5885f56b-d6da-43a3-8760-83583af94266

                                UID         City State        County    Zip  \
0  3a83ddb66e2ae73798bdf1d705dc0932          Eva    AL        Morgan  35621
1  176354c5eef714957d486009feabf195     Marianna    FL       Jackson  32446
2  e19a0fa00aeda885b8a436757e889bc9  Sioux Falls    SD     Minnehaha  57110
3  cd17d7b6d152cb6f23957346d11c3f07  New Richland    MN        Waseca  56072
4  d2f0425877b10ed6bb381f3e2579424a   West Point    VA  King William  23181

        Lat       Lng  …  TotalCharge Additional_charges Item1 Item2  Item3  \
0  34.34960 -86.72508  …  3726.702860       17939.403420     3     3      2
1  30.84513 -85.22907  …  4193.190458       17612.998120     3     4      3
2  43.54321 -96.63772  …  2434.234222       17505.192460     2     4      4
3  43.89744 -93.51479  …  2127.830423       12993.437350     3     5      5
4  37.59894 -76.88958  …  2113.073274        3716.525786     2     1      3

   Item4  Item5 Item6 Item7 Item8
0      2      4     3     3     4
1      4      4     4     3     3
2      4      3     4     3     3
3      3      4     5     5     5
4      3      5     3     4     3

[5 rows x 50 columns]
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')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   CaseOrder           10000 non-null  int64
 1   Customer_id         10000 non-null  object
 2   Interaction         10000 non-null  object
 3   UID                 10000 non-null  object
 4   City                10000 non-null  object
 5   State               10000 non-null  object
 6   County              10000 non-null  object
 7   Zip                 10000 non-null  int64
 8   Lat                 10000 non-null  float64
 9   Lng                 10000 non-null  float64
 10  Population          10000 non-null  int64
 11  Area                10000 non-null  object
 12  TimeZone            10000 non-null  object
 13  Job                 10000 non-null  object
 14  Children            10000 non-null  int64
 15  Age                 10000 non-null  int64
 16  Income              10000 non-null  float64
 17  Marital             10000 non-null  object
 18  Gender              10000 non-null  object
 19  ReAdmis             10000 non-null  object
 20  VitD_levels         10000 non-null  float64
 21  Doc_visits          10000 non-null  int64
 22  Full_meals_eaten    10000 non-null  int64
 23  vitD_supp           10000 non-null  int64
 24  Soft_drink          10000 non-null  object
 25  Initial_admin       10000 non-null  object
 26  HighBlood           10000 non-null  object
 27  Stroke              10000 non-null  object
 28  Complication_risk   10000 non-null  object
 29  Overweight          10000 non-null  object
 30  Arthritis           10000 non-null  object
 31  Diabetes            10000 non-null  object
 32  Hyperlipidemia      10000 non-null  object
 33  BackPain            10000 non-null  object
 34  Anxiety             10000 non-null  object
 35  Allergic_rhinitis   10000 non-null  object
 36  Reflux_esophagitis  10000 non-null  object
 37  Asthma              10000 non-null  object
 38  Services            10000 non-null  object
```

```
 39  Initial_days        10000 non-null  float64
 40  TotalCharge         10000 non-null  float64
 41  Additional_charges  10000 non-null  float64
 42  Item1               10000 non-null  int64
 43  Item2               10000 non-null  int64
 44  Item3               10000 non-null  int64
 45  Item4               10000 non-null  int64
 46  Item5               10000 non-null  int64
 47  Item6               10000 non-null  int64
 48  Item7               10000 non-null  int64
 49  Item8               10000 non-null  int64
dtypes: float64(7), int64(16), object(27)
memory usage: 3.8+ MB
None
          CaseOrder Customer_id                           Interaction  \
count   10000.00000       10000                                 10000
unique          NaN       10000                                 10000
top             NaN     C412403  8cd49b13-f45a-4b47-a2bd-173ffa932c2f
freq            NaN           1                                     1
mean     5000.50000         NaN                                   NaN
std      2886.89568         NaN                                   NaN
min         1.00000         NaN                                   NaN
25%      2500.75000         NaN                                   NaN
50%      5000.50000         NaN                                   NaN
75%      7500.25000         NaN                                   NaN
max     10000.00000         NaN                                   NaN

                                     UID     City  State     County  \
count                              10000    10000  10000      10000
unique                             10000     6072     52       1607
top     3a83ddb66e2ae73798bdf1d705dc0932  Houston     TX  Jefferson
freq                                   1       36    553        118
mean                                 NaN      NaN    NaN        NaN
std                                  NaN      NaN    NaN        NaN
min                                  NaN      NaN    NaN        NaN
25%                                  NaN      NaN    NaN        NaN
50%                                  NaN      NaN    NaN        NaN
75%                                  NaN      NaN    NaN        NaN
max                                  NaN      NaN    NaN        NaN

                 Zip           Lat           Lng  …  TotalCharge  \
count   10000.000000  10000.000000  10000.000000  …  10000.000000
unique           NaN           NaN           NaN  …           NaN
top              NaN           NaN           NaN  …           NaN
freq             NaN           NaN           NaN  …           NaN
mean    50159.323900     38.751099    -91.243080  …   5312.172769
std     27469.588208      5.403085     15.205998  …   2180.393838
min       610.000000     17.967190   -174.209700  …   1938.312067
```

```
25%     27592.000000    35.255120   -97.352982  …   3179.374015
50%     50207.000000    39.419355   -88.397230  …   5213.952000
75%     72411.750000    42.044175   -80.438050  …   7459.699750
max     99929.000000    70.560990   -65.290170  …   9180.728000

       Additional_charges         Item1         Item2         Item3  \
count        10000.000000  10000.000000  10000.000000  10000.000000
unique                NaN           NaN           NaN           NaN
top                   NaN           NaN           NaN           NaN
freq                  NaN           NaN           NaN           NaN
mean         12934.528587      3.518800      3.506700      3.511100
std           6542.601544      1.031966      1.034825      1.032755
min           3125.703000      1.000000      1.000000      1.000000
25%           7986.487755      3.000000      3.000000      3.000000
50%          11573.977735      4.000000      3.000000      4.000000
75%          15626.490000      4.000000      4.000000      4.000000
max          30566.070000      8.000000      7.000000      8.000000

               Item4         Item5         Item6         Item7         Item8
count   10000.000000  10000.000000  10000.000000  10000.000000  10000.000000
unique           NaN           NaN           NaN           NaN           NaN
top              NaN           NaN           NaN           NaN           NaN
freq             NaN           NaN           NaN           NaN           NaN
mean        3.515100      3.496900      3.522500      3.494000      3.509700
std         1.036282      1.030192      1.032376      1.021405      1.042312
min         1.000000      1.000000      1.000000      1.000000      1.000000
25%         3.000000      3.000000      3.000000      3.000000      3.000000
50%         4.000000      3.000000      4.000000      3.000000      3.000000
75%         4.000000      4.000000      4.000000      4.000000      4.000000
max         7.000000      7.000000      7.000000      7.000000      7.000000

[11 rows x 50 columns]
```

[316]:

```
from sklearn.preprocessing import StandardScaler
import pandas as pd

# dataframe with only the relevant continuous features needed for this analysis
wgu = wgu[['Population', 'Children', 'Age', 'Income', 'VitD_levels',␣
 ↪'Doc_visits', 'Full_meals_eaten', 'vitD_supp',
       'Initial_days', 'TotalCharge', 'Additional_charges']]

# standardize the dataset
scaler = StandardScaler()
norm_wgu = scaler.fit_transform(wgu)
scaled_wgu = pd.DataFrame(norm_wgu, columns=wgu.columns)

# dataset first 5 rows
```

```python
print(scaled_wgu.head())

# variable statistics to check distributions
print(scaled_wgu.describe().round(2))
```

```
   Population  Children       Age    Income  VitD_levels  Doc_visits  \
0   -0.473168 -0.507129 -0.024795  1.615914     0.583603    0.944647
1    0.090242  0.417277 -0.121706  0.221443     0.483901   -0.967981
2    0.482983  0.417277 -0.024795 -0.915870     0.046227   -0.967981
3   -0.526393 -0.969332  1.186592 -0.026263    -0.687811   -0.967981
4   -0.315586 -0.507129 -1.526914 -1.377325    -0.260366   -0.011667

   Full_meals_eaten  vitD_supp  Initial_days  TotalCharge  Additional_charges
0         -0.993387  -0.634713     -0.907310    -0.727185            0.765005
1          0.990609   0.956445     -0.734595    -0.513228            0.715114
2         -0.001389  -0.634713     -1.128292    -1.319983            0.698635
3         -0.001389  -0.634713     -1.244503    -1.460517            0.009004
4         -0.993387   2.547602     -1.261991    -1.467285           -1.408991
       Population  Children       Age    Income  VitD_levels  Doc_visits  \
count    10000.00  10000.00  10000.00  10000.00     10000.00    10000.00
mean        -0.00      0.00      0.00      0.00        -0.00        0.00
std          1.00      1.00      1.00      1.00         1.00        1.00
min         -0.67     -0.97     -1.72     -1.41        -4.04       -3.84
25%         -0.63     -0.97     -0.85     -0.73        -0.66       -0.97
50%         -0.49     -0.51     -0.02     -0.24        -0.01       -0.01
75%          0.27      0.42      0.85      0.48         0.69        0.94
max          7.61      3.65      1.72      5.85         4.18        3.81

       Full_meals_eaten  vitD_supp  Initial_days  TotalCharge  \
count          10000.00   10000.00      10000.00     10000.00
mean              -0.00       0.00         -0.00         0.00
std                1.00       1.00          1.00         1.00
min               -0.99      -0.63         -1.27        -1.55
25%               -0.99      -0.63         -1.01        -0.98
50%               -0.00      -0.63          0.05        -0.05
75%                0.99       0.96          1.02         0.98
max                5.95       7.32          1.43         1.77

       Additional_charges
count            10000.00
mean                -0.00
std                  1.00
min                 -1.50
25%                 -0.76
50%                 -0.21
75%                  0.41
max                  2.70
```

[317]:

```python
# creating the covariance matrix based on our standardized data
cov_matrix = pd.DataFrame.cov(scaled_wgu)
print(cov_matrix)
```

```
                    Population  Children       Age    Income  VitD_levels  \
Population            1.000100  0.002462 -0.018989  0.005427     0.002652
Children              0.002462  1.000100  0.009837  0.007177     0.009488
Age                  -0.018989  0.009837  1.000100 -0.012229     0.010316
Income                0.005427  0.007177 -0.012229  1.000100    -0.013116
VitD_levels           0.002652  0.009488  0.010316 -0.013116     1.000100
Doc_visits            0.012647 -0.002292  0.006899  0.013465     0.010211
Full_meals_eaten     -0.025610  0.003835  0.008556 -0.011366     0.023226
vitD_supp             0.009782 -0.004320  0.010015  0.001254    -0.007204
Initial_days          0.017471  0.022469  0.016266 -0.012466    -0.003642
TotalCharge           0.019190  0.024103  0.016877 -0.014347    -0.001403
Additional_charges   -0.004821  0.013550  0.716925 -0.009826     0.008291

                    Doc_visits  Full_meals_eaten  vitD_supp  Initial_days  \
Population            0.012647         -0.025610   0.009782      0.017471
Children             -0.002292          0.003835  -0.004320      0.022469
Age                   0.006899          0.008556   0.010015      0.016266
Income                0.013465         -0.011366   0.001254     -0.012466
VitD_levels           0.010211          0.023226  -0.007204     -0.003642
Doc_visits            1.000100         -0.002768   0.005682     -0.006755
Full_meals_eaten     -0.002768          1.000100  -0.019982     -0.017269
vitD_supp             0.005682         -0.019982   1.000100      0.015976
Initial_days         -0.006755         -0.017269   0.015976      1.000100
TotalCharge          -0.005044         -0.014307   0.016926      0.987739
Additional_charges    0.008072          0.018765   0.010328      0.004409

                    TotalCharge  Additional_charges
Population             0.019190           -0.004821
Children               0.024103            0.013550
Age                    0.016877            0.716925
Income                -0.014347           -0.009826
VitD_levels           -0.001403            0.008291
Doc_visits            -0.005044            0.008072
Full_meals_eaten      -0.014307            0.018765
vitD_supp              0.016926            0.010328
Initial_days           0.987739            0.004409
TotalCharge            1.000100            0.029259
Additional_charges     0.029259            1.000100
```

According to the covariance matrix above, the correlation between the `TotalCharge` and `Initial_days` variables is 0.987739. This indicates that these variables provide nearly the same information, demonstrating the effectiveness of using a covariance matrix. By removing the `TotalCharge` variable, the dataset will become less redundant for this analysis. The remaining variables are standardized below and outputted to a CSV file. The variables all have means of 0 and standard deviations of 1.0, as displayed in the variable statistics.

[319]:
```python
# dataframe with only the relevant continuous features needed for this analysis
wgu = wgu[['Population', 'Children', 'Age', 'Income', 'VitD_levels',␣
↪'Doc_visits', 'Full_meals_eaten', 'vitD_supp',
       'Initial_days', 'Additional_charges']]

# standardize the dataset
scaler = StandardScaler()
norm_wgu = scaler.fit_transform(wgu)

scaled_wgu = pd.DataFrame(norm_wgu, columns=wgu.columns)
# output to a csv file
scaled_wgu.to_csv('scaled_wgu.csv', index=False)
# dataset first 5 rows
print(scaled_wgu.head())

# variable statistics to check distributions
print(scaled_wgu.describe().round(2))
```

```
   Population  Children       Age    Income  VitD_levels  Doc_visits  \
0   -0.473168 -0.507129 -0.024795  1.615914     0.583603    0.944647
1    0.090242  0.417277 -0.121706  0.221443     0.483901   -0.967981
2    0.482983  0.417277 -0.024795 -0.915870     0.046227   -0.967981
3   -0.526393 -0.969332  1.186592 -0.026263    -0.687811   -0.967981
4   -0.315586 -0.507129 -1.526914 -1.377325    -0.260366   -0.011667

   Full_meals_eaten  vitD_supp  Initial_days  Additional_charges
0         -0.993387  -0.634713     -0.907310            0.765005
1          0.990609   0.956445     -0.734595            0.715114
2         -0.001389  -0.634713     -1.128292            0.698635
3         -0.001389  -0.634713     -1.244503            0.009004
4         -0.993387   2.547602     -1.261991           -1.408991
       Population  Children       Age    Income  VitD_levels  Doc_visits  \
count    10000.00  10000.00  10000.00  10000.00     10000.00    10000.00
mean        -0.00      0.00      0.00      0.00        -0.00        0.00
std          1.00      1.00      1.00      1.00         1.00        1.00
min         -0.67     -0.97     -1.72     -1.41        -4.04       -3.84
25%         -0.63     -0.97     -0.85     -0.73        -0.66       -0.97
50%         -0.49     -0.51     -0.02     -0.24        -0.01       -0.01
75%          0.27      0.42      0.85      0.48         0.69        0.94
max          7.61      3.65      1.72      5.85         4.18        3.81

       Full_meals_eaten  vitD_supp  Initial_days  Additional_charges
count          10000.00   10000.00      10000.00            10000.00
mean              -0.00       0.00         -0.00               -0.00
std                1.00       1.00          1.00                1.00
min               -0.99      -0.63         -1.27               -1.50
```

```
25%                     -0.99      -0.63         -1.01                -0.76
50%                     -0.00      -0.63          0.05                -0.21
75%                      0.99       0.96          1.02                 0.41
max                      5.95       7.32          1.43                 2.70
```

D. Principal Component Matrix

As the data was standardized in the previous step and one feature was removed based on the covariance matrix, the remaining 10 variables can now be used with PCA to reduce the dimensionality of the dataset. The PCA object was instantiated, and the principal components were fit to the standardized dataset using the `fit_transform` function. The steps performed by the PCA function are then outputted, and the matrix of all the principal components is displayed.

Note: The same pca library was used as shown in Dr.Kamara's analysis webinar, ensure it is installed before running with: `pip install pca`

[321]:

```
from pca import pca

# creating the PCA object and passing the 10 variables
pca = pca(n_components = 10)

# fitting the PCA to the standardized dataset and transforming
principalComps = pca.fit_transform(scaled_wgu)

loadings = principalComps["loadings"]
loadings
```

```
[pca] >Extracting column labels from dataframe.
[pca] >Extracting row labels from dataframe.
[pca] >The PCA reduction is performed on the [10] columns of the input
dataframe.
[pca] >Fit using PCA.
[pca] >Compute loadings and PCs.
[pca] >Compute explained variance.
[pca] >Outlier detection using Hotelling T2 test with alpha=[0.05] and
n_components=[10]
[pca] >Multiple test correction applied for Hotelling T2 test: [fdr_bh]
[pca] >Outlier detection using SPE/DmodX with n_std=[3]
```

[321]:

```
      Population  Children       Age    Income  VitD_levels  Doc_visits  \
PC1    -0.023358  0.023518  0.705884 -0.022430     0.019681    0.013911
PC2     0.452197  0.080730  0.013160  0.161161    -0.277246    0.102807
PC3    -0.077229 -0.494308  0.024228  0.461923    -0.347531    0.280302
PC4     0.342053  0.264110 -0.013894  0.282271     0.524935    0.653837
PC5    -0.196083  0.615226  0.005101  0.587476    -0.305780   -0.190391
PC6    -0.546957  0.181316 -0.024289  0.124530     0.053374    0.175988
PC7     0.292072 -0.026989  0.006391  0.339398     0.439885   -0.642919
PC8     0.434050 -0.136659 -0.011245  0.130000    -0.381415   -0.027659
PC9     0.245925  0.498032 -0.009845 -0.433293    -0.301795    0.042836
```

```
PC10    0.014584  0.003942  0.707033  0.002118    -0.002469    0.000829

      Full_meals_eaten  vitD_supp  Initial_days  Additional_charges
PC1           0.027671   0.019052      0.020365            0.705693
PC2          -0.577243   0.416054      0.411119            0.009479
PC3          -0.156815   0.031173     -0.556715            0.029904
PC4           0.126118  -0.082562     -0.087113           -0.002341
PC5           0.006146  -0.325781      0.043989            0.006142
PC6           0.216341   0.753368      0.052015           -0.031174
PC7          -0.007682   0.306152     -0.312782            0.016430
PC8           0.760558   0.128683      0.186015            0.006819
PC9          -0.008418   0.186445     -0.613050            0.008178
PC10          0.010406   0.000651     -0.011899           -0.706835
```

[322]:
```
principalComps
```

[322]:
```
{'loadings':       Population  Children       Age    Income  VitD_levels
Doc_visits  \
 PC1    -0.023358  0.023518  0.705884 -0.022430     0.019681    0.013911
 PC2     0.452197  0.080730  0.013160  0.161161    -0.277246    0.102807
 PC3    -0.077229 -0.494308  0.024228  0.461923    -0.347531    0.280302
 PC4     0.342053  0.264110 -0.013894  0.282271     0.524935    0.653837
 PC5    -0.196083  0.615226  0.005101  0.587476    -0.305780   -0.190391
 PC6    -0.546957  0.181316 -0.024289  0.124530     0.053374    0.175988
 PC7     0.292072 -0.026989  0.006391  0.339398     0.439885   -0.642919
 PC8     0.434050 -0.136659 -0.011245  0.130000    -0.381415   -0.027659
 PC9     0.245925  0.498032 -0.009845 -0.433293    -0.301795    0.042836
 PC10    0.014584  0.003942  0.707033  0.002118    -0.002469    0.000829

       Full_meals_eaten  vitD_supp  Initial_days  Additional_charges
 PC1           0.027671   0.019052      0.020365            0.705693
 PC2          -0.577243   0.416054      0.411119            0.009479
 PC3          -0.156815   0.031173     -0.556715            0.029904
 PC4           0.126118  -0.082562     -0.087113           -0.002341
 PC5           0.006146  -0.325781      0.043989            0.006142
 PC6           0.216341   0.753368      0.052015           -0.031174
 PC7          -0.007682   0.306152     -0.312782            0.016430
 PC8           0.760558   0.128683      0.186015            0.006819
 PC9          -0.008418   0.186445     -0.613050            0.008178
 PC10          0.010406   0.000651     -0.011899           -0.706835  ,
 'PC':            PC1       PC2       PC3       PC4       PC5       PC6
PC7  \
 0     0.451806 -0.115905  1.758995  1.089050  0.537119 -0.198045  0.182826
 1     0.448211 -0.594211 -0.248571 -0.065326  0.071399  0.786668  1.451393
 2     0.446930 -0.728683 -0.325157 -0.442521 -0.044576 -1.028789  0.631548
 3     0.769560 -0.988554  1.177663 -1.293301  0.044175 -0.670484  0.385960
 4    -2.055738  0.746455  0.584682 -0.976138 -1.885450  1.612934  0.496427
```

```
 …         …         …         …         …         …         …         …
 9995 -1.368771 -0.026308 -0.503171 -0.956583  0.199668  1.032713  0.434693
 9996  2.868489  0.518516 -1.319903 -0.387802  0.381665 -0.365865 -0.989057
 9997 -0.011986 -0.375156 -0.802548 -0.642025  1.488832  0.147844 -0.077429
 9998 -0.794955 -0.328094 -1.442085  0.237772 -0.394691  1.464614  0.026657
 9999  0.444543  2.797569 -1.801104  1.435809  1.402125  0.000698  0.781896

           PC8       PC9      PC10
 0    -1.175211 -0.752010 -0.564352
 1     0.599233  0.573932 -0.570412
 2    -0.244877  1.247349 -0.492520
 3    -0.137824  0.198288  0.836318
 4    -0.801980  1.605207 -0.086203
 …         …         …         …
 9995  1.100280 -0.306276 -0.545185
 9996 -1.128835 -0.255101 -0.568684
 9997  0.898607 -1.203060 -0.556924
 9998  0.323481 -0.585712  0.183366
 9999  0.186999  0.810273  0.721145

 [10000 rows x 10 columns],
 'explained_var': array([0.17195029, 0.27800427, 0.3807181 , 0.48263541,
0.58325652,
       0.68200635, 0.77938297, 0.8757476 , 0.9717193 , 1.        ]),
 'variance_ratio': array([0.17195029, 0.10605398, 0.10271384, 0.10191731,
0.1006211 ,
       0.09874983, 0.09737662, 0.09636463, 0.0959717 , 0.0282807 ]),
 'model': PCA(n_components=10),
 'scaler': None,
 'pcp': 1.0000000000000002,
 'topfeat':       PC             feature   loading  type
 0    PC1                 Age  0.705884  best
 1    PC2    Full_meals_eaten -0.577243  best
 2    PC3        Initial_days -0.556715  best
 3    PC4          Doc_visits  0.653837  best
 4    PC5            Children  0.615226  best
 5    PC6           vitD_supp  0.753368  best
 6    PC7          Doc_visits -0.642919  best
 7    PC8    Full_meals_eaten  0.760558  best
 8    PC9        Initial_days -0.613050  best
 9   PC10                 Age  0.707033  best
 10   PC6          Population -0.546957  weak
 11   PC5              Income  0.587476  weak
 12   PC4          VitD_levels  0.524935  weak
 13  PC10  Additional_charges -0.706835  weak,
 'outliers':        y_proba     p_raw    y_score  y_bool  y_bool_spe
y_score_spe
```

```
0     0.990918  0.718078  15.976555   False       False     0.466436
1     0.990918  0.926484  11.685239   False       False     0.744298
2     0.990918  0.915679  12.011523   False       False     0.854825
3     0.990918  0.800256  14.573755   False       False     1.252782
4     0.826474  0.091857  28.795899   False       False     2.187065
…        …         …         …      …        …          …
9995  0.990918  0.804145  14.502262   False       False     1.369024
9996  0.990918  0.243219  23.981093   False       False     2.914977
9997  0.990918  0.823281  14.140903   False       False     0.375348
9998  0.990918  0.866389  13.250291   False       False     0.860000
9999  0.798458  0.081517  29.327026   False       False     2.832669

[10000 rows x 6 columns],
'outliers_params': {'paramT2': (-1.1368683772161604e-18, 1.0),
 'paramSPE': (array([-2.50521826e-17, -1.75681691e-16]),
  array([[ 1.71967482e+00, -3.26349386e-16],
         [-3.26349386e-16,  1.06064589e+00]]))}}
```

[323]:
```
# showing the features that have the most variance on each component
print(pca.results['topfeat'])
```

```
      PC            feature   loading  type
0    PC1                Age  0.705884  best
1    PC2   Full_meals_eaten -0.577243  best
2    PC3       Initial_days -0.556715  best
3    PC4         Doc_visits  0.653837  best
4    PC5           Children  0.615226  best
5    PC6          vitD_supp  0.753368  best
6    PC7         Doc_visits -0.642919  best
7    PC8   Full_meals_eaten  0.760558  best
8    PC9       Initial_days -0.613050  best
9   PC10                Age  0.707033  best
10   PC6         Population -0.546957  weak
11   PC5             Income  0.587476  weak
12   PC4        VitD_levels  0.524935  weak
13  PC10 Additional_charges -0.706835  weak
```

D2. Principal Components

The PCA object from the PCA library supports the `n_components` attribute. When this attribute is given a percentage, it will capture the specified percentage of explained variance from the provided features. By setting `n_components` to 0.8, the PCA is instructed to determine the principal components that capture at least 80% of the explained variance, following the elbow rule.

The PCA object is instantiated, and the components are fit to the standardized dataset again. The output of the loadings shows that 8 principal components were selected, which explain at least 80% of the variance. The scree plot is also displayed, showing the explained variance by principal components. Incrementing from 7 to 8 PCs allows the explained variance to cover at least 80%, meeting the elbow rule criterion if relying purely on the skree plot.

[325]:
```python
from pca import pca

# creating the PCA object, capturing 80% of explained variance from the 10␣
↪features passed in
# based on elbow rule method
pca = pca(n_components = 0.8)

# fitting the PCA to the standardized dataset and transforming
principalComps = pca.fit_transform(scaled_wgu)

loadings = principalComps["loadings"]
loadings
```

```
[pca] >Extracting column labels from dataframe.
[pca] >Extracting row labels from dataframe.
[pca] >The PCA reduction is performed to capture [80.0%] explained variance
using the [10] columns of the input data.
[pca] >Fit using PCA.
[pca] >Compute loadings and PCs.
[pca] >Compute explained variance.
[pca] >Number of components is [8] that covers the [80.00%] explained variance.
[pca] >The PCA reduction is performed on the [10] columns of the input
dataframe.
[pca] >Fit using PCA.
[pca] >Compute loadings and PCs.
[pca] >Outlier detection using Hotelling T2 test with alpha=[0.05] and
n_components=[8]
[pca] >Multiple test correction applied for Hotelling T2 test: [fdr_bh]
[pca] >Outlier detection using SPE/DmodX with n_std=[3]
```

[325]:

	Population	Children	Age	Income	VitD_levels	Doc_visits	\
PC1	-0.023358	0.023518	0.705884	-0.022430	0.019681	0.013911	
PC2	0.452197	0.080730	0.013160	0.161161	-0.277246	0.102807	
PC3	-0.077229	-0.494308	0.024228	0.461923	-0.347531	0.280302	
PC4	0.342053	0.264110	-0.013894	0.282271	0.524935	0.653837	
PC5	-0.196083	0.615226	0.005101	0.587476	-0.305780	-0.190391	
PC6	-0.546957	0.181316	-0.024289	0.124530	0.053374	0.175988	
PC7	0.292072	-0.026989	0.006391	0.339398	0.439885	-0.642919	
PC8	0.434050	-0.136659	-0.011245	0.130000	-0.381415	-0.027659	

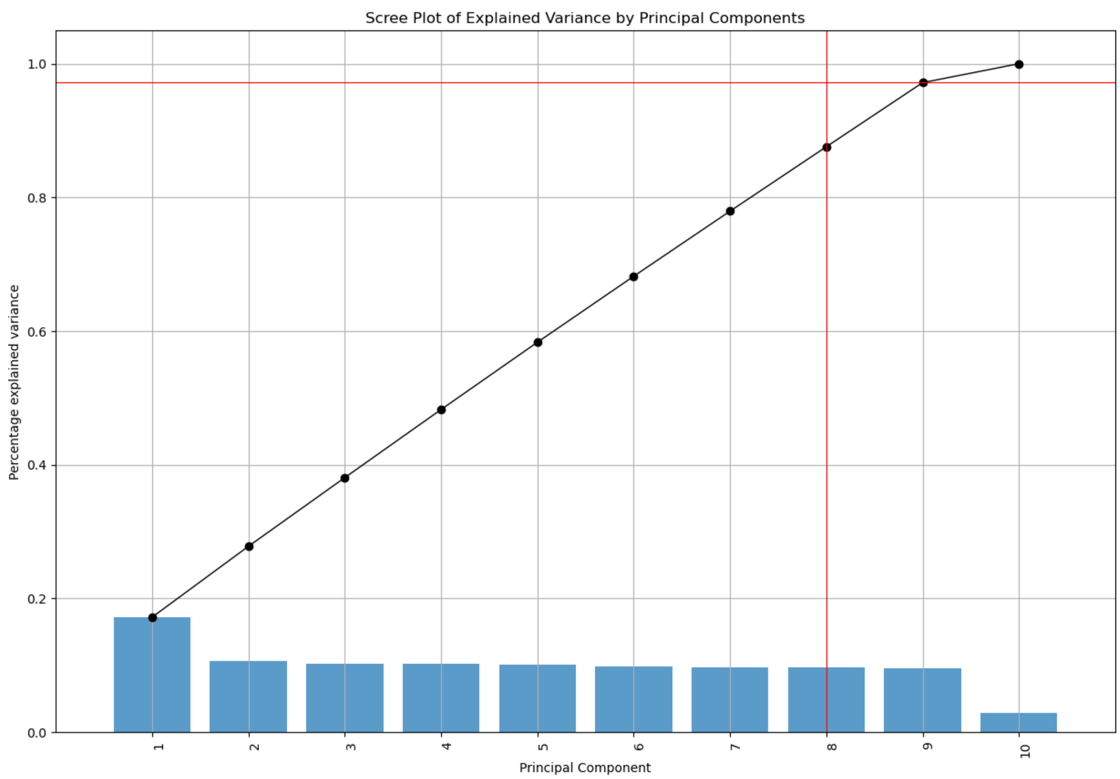
	Full_meals_eaten	vitD_supp	Initial_days	Additional_charges
PC1	0.027671	0.019052	0.020365	0.705693
PC2	-0.577243	0.416054	0.411119	0.009479
PC3	-0.156815	0.031173	-0.556715	0.029904
PC4	0.126118	-0.082562	-0.087113	-0.002341
PC5	0.006146	-0.325781	0.043989	0.006142
PC6	0.216341	0.753368	0.052015	-0.031174

PC7	-0.007682	0.306152	-0.312782	0.016430
PC8	0.760558	0.128683	0.186015	0.006819

[326]:
```
# generating the skree plot to show the explained variance by principal␣
↪components
fig, ax = pca.plot()

# based on the explained var array, display the total variance for 8 PCs
total_variance = principalComps['explained_var'][7]  # starts from 0
title = f'Scree Plot of Explained Variance by Principal Components'
ax.set_title(title)

# Show the plot
plt.show()
```



D3. Variances

Based on the pca function in the previous section the 8 principal components selected make up at least 80% of the explained variance. The variance of each principal component is displayed below by referring to the variance_ratio property:

[328]:
```
# selecting the variances of the first 8 PCs from the pca
variances = principalComps["variance_ratio"][:8] # first 8 PCs
```

```
variance_df = pd.DataFrame(
    [f"{(v * 100):.2f}%" for v in variances],
    index=[f"PC{i+1}" for i in range(len(variances))],
    columns=['Variance']
)
# display variance table
variance_df
```

[328]:

	Variance
PC1	17.20%
PC2	10.61%
PC3	10.27%
PC4	10.19%
PC5	10.06%
PC6	9.87%
PC7	9.74%
PC8	9.64%

D4. Total Variance

The variances of each principal component were displayed in the previous section. The `explained_var` property of the pca shows the cumulative variance based on the number of principal components. The total variance captured by the principal components identified in section D2 is displayed below by referring to the 7th index in the `explained_var` array, as it represents the cumulative variance up to the 8th principal component:

[330]:
```
total_variance = principalComps['explained_var'][7]  # starts from 0
print(f'The {len(variances)} principal components explain {total_variance*100:.
 ↪2f}% of the variance.')
```

```
The 8 principal components explain 87.57% of the variance.
```

D5. Analysis Results Summary

The analysis used Principal Component Analysis (PCA) to reduce the dimensionality of the dataset, which consisted of 11 continuous variables. After standardizing the data, a covariance matrix was created, revealing a strong correlation between `TotalCharge` and `Initial_days`. The `TotalCharge` variable was removed before PCA was performed as it became redundant. A loading matrix was then generated using the 10 remaining variables.

The elbow rule was used to retain 8 principal components that explained 87.57% of the variance. The elbow rule has several variations for defining a threshold. One article (What Is Principal Component Analysis (PCA)? | IBM, n.d.) mentions that the point at which the Y-axis of the total variance explained creates an "elbow" generally indicates how many PCA components to include. Another article (Determining the Number of Components in Principal Components Analysis - Displayr, n.d.) suggests using the point prior to where the "elbow" is created. In this case, the 8 principal components were selected because that was the number of components that appeared prior to the "elbow" in the scree plot.

With the analysis completed, the dimensionality of the data was reduced while still maintaining an explained variance of 87.57%. The number of principal components retained is higher than what is normally considered optimal for PCA, with anything higher than 3 being less optimal (Mangale, 2021). The number of components can be adjusted by using a different criterion, such as the Kaiser criterion, or by using a different dimensionality reduction technique.

E. Third-party Code References

Algorithm — pca pca documentation. (n.d.). https://erdogant.github.io/pca/pages/html/Algorithm.html#explai variance

Panopto. (n.d.). Panopto. https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=8c618a96-fdb8-4757-abe2-b023018520ac

F. References

A Guide to Principal Component Analysis (PCA) for Machine learning. (n.d.). https://www.keboola.com/blog/pca-machine-learning

Determining the number of components in principal components analysis - Displayr. (n.d.). https://docs.displayr.com/wiki/Determining_the_Number_of_Components_in_Principal_Components_Analy

Jaadi, Z. (2024, February 23). A Step-by-Step Explanation of Principal Component Analysis (PCA). Built In. https://builtin.com/data-science/step-step-explanation-principal-component-analysis

Mangale, S. (2021, December 15). Scree Plot - SANCHITA MANGALE - Medium. Medium. https://sanchitamangale12.medium.com/scree-plot-733ed72c8608

What is Principal Component Analysis (PCA)? | IBM. (n.d.). https://www.ibm.com/topics/principal-component-analysis