APRIL 8, 2024

PERFORMANCE ASSESSMENT

D212

ROBERT PATTON
WESTERN GOVERNORS' UNIVERSITY
Rpat33@wgu.edu

Table of Contents

Part I: Research Question **2**

A. Purpose of Data Mining Report **2**

1. Research Question **2**
2. Goal of Analysis **2**

Part II: Method Justification **2**

B. Reasoning for PCA **2**

1. PCA Analysis **2**
2. One Assumption of PCA **2**

Part III: Data Preparation **2**

C. Preparing the Data **2**

1. Continuous Data Set Variables **2**
2. Data Set Standardization **3**

Part IV: Analysis **3**

D. PCA Performance **3**

1. Matrix of Principal Components **3**
2. Elbow Method for Optimal Components **3-4**
3. Variance of Principal Components **5**
4. Total Variance Captured **5-6**
5. Summary of Analysis **6**

Part V: Sources **7**

Part I: Research Question

A. Purpose of Data Mining Report

1) Using principal component analysis, how many continuous patient variables can be used as quality variables for data analysis?
2) In an effort to describe variablity among patient variables in the dataset, the goal for this data anlysis is to reduce dimensions of the data and identify the most optimal number of principal components for PCA.

Part II: Method Justification

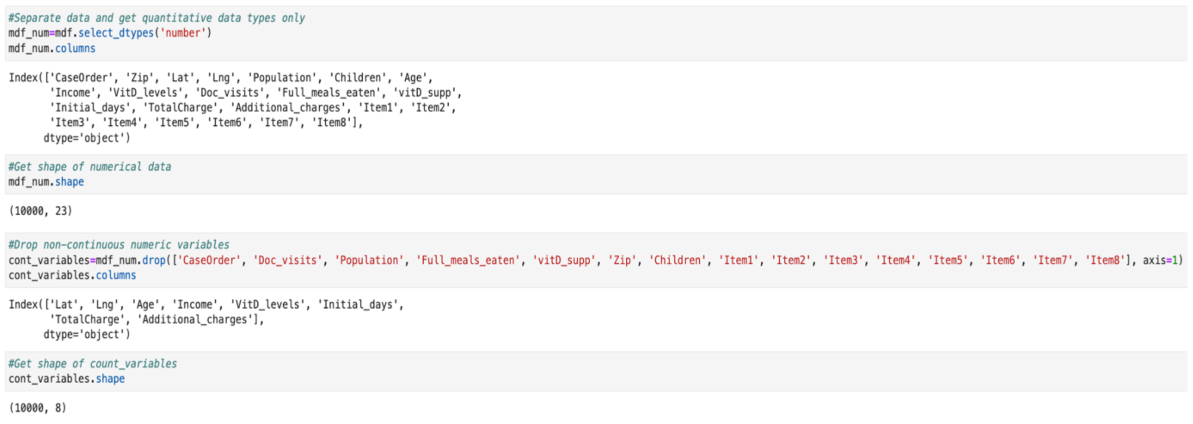
B. Reasoning for PCA

1) The use of PCA is to reduce dimensionality of data and speed up machine learning and model training (Whitfield and Galarnyk 2024). PCA is an unsupervised dimensionality reduction technique that aims to reduce orignial variables in large data sets, while retaining as much information as possible, into uncorrelated variables called principal components (avcontentteam 2024). PCA analyzes data by transforming original data points into new sets of orthogonal axes, those of which are linear combinations of the orignal variables (Aggarwal 2023). By transforming the original data points, redundancy of information is minimized, which is importan in PCA, and each principal component cultivates a unique aspect of the datas variance. After PCA is applied, the analyst can then look at the variance values, the amount of information being retained, for each component and determine which components are of value to the analyis.
2) One assumption of PCA is that variables with larger variance are more significant to the analysis versus those with smaller variance (Aggarwal 2023).

Part III: Data Preparation

C. Preparing the Data

1) A screen shot for identifying the continuous variables in the data set is provided below:



```
#Separate data and get quantitative data types only
mdf_num=mdf.select_dtypes('number')
mdf_num.columns
```

```
Index(['CaseOrder', 'Zip', 'Lat', 'Lng', 'Population', 'Children', 'Age',
       'Income', 'VitD_levels', 'Doc_visits', 'Full_meals_eaten', 'vitD_supp',
       'Initial_days', 'TotalCharge', 'Additional_charges', 'Item1', 'Item2',
       'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8'],
      dtype='object')
```

```
#Get shape of numerical data
mdf_num.shape
```

```
(10000, 23)
```

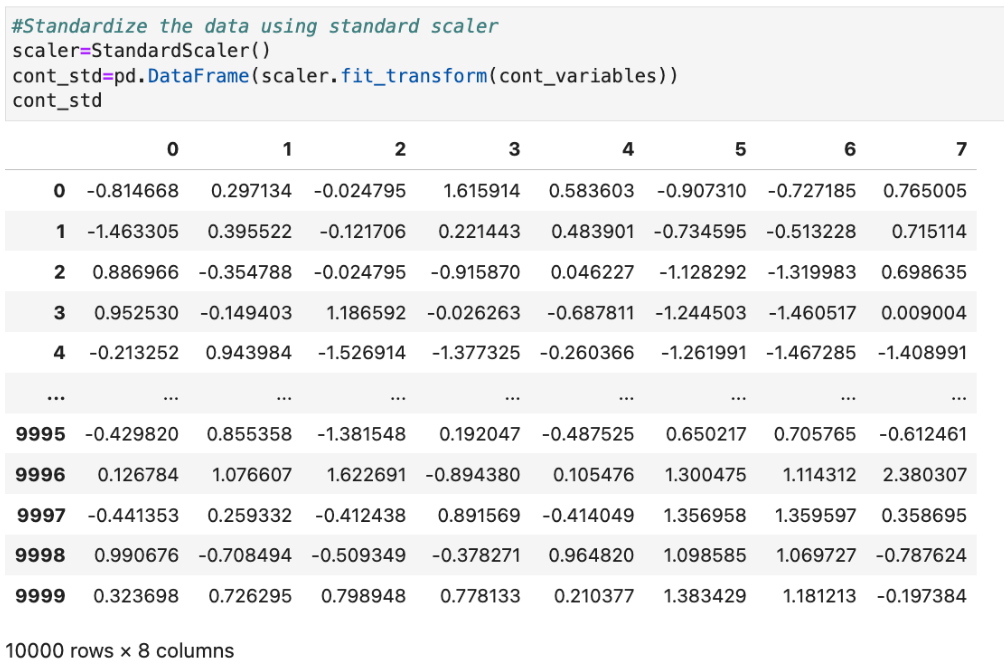
```
#Drop non-continuous numeric variables
cont_variables=mdf_num.drop(['CaseOrder', 'Doc_visits', 'Population', 'Full_meals_eaten', 'vitD_supp', 'Zip', 'Children', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8'], axis=1)
cont_variables.columns
```

```
Index(['Lat', 'Lng', 'Age', 'Income', 'VitD_levels', 'Initial_days',
       'TotalCharge', 'Additional_charges'],
      dtype='object')
```

```
#Get shape of count_variables
cont_variables.shape
```

```
(10000, 8)
```

2) A screen shot for standardizing the selected continuous variables can be see below and the cleaned data set is provided in the task 2 submission.



```python
#Standardize the data using standard scaler
scaler=StandardScaler()
cont_std=pd.DataFrame(scaler.fit_transform(cont_variables))
cont_std
```

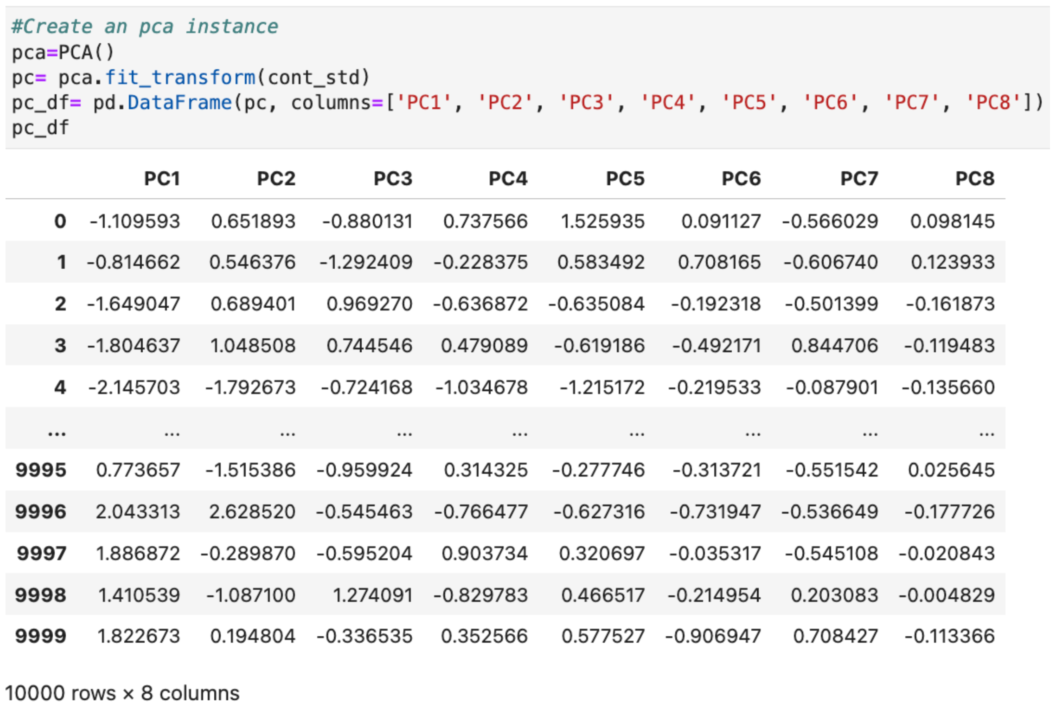
	0	1	2	3	4	5	6	7
0	-0.814668	0.297134	-0.024795	1.615914	0.583603	-0.907310	-0.727185	0.765005
1	-1.463305	0.395522	-0.121706	0.221443	0.483901	-0.734595	-0.513228	0.715114
2	0.886966	-0.354788	-0.024795	-0.915870	0.046227	-1.128292	-1.319983	0.698635
3	0.952530	-0.149403	1.186592	-0.026263	-0.687811	-1.244503	-1.460517	0.009004
4	-0.213252	0.943984	-1.526914	-1.377325	-0.260366	-1.261991	-1.467285	-1.408991
...	...	...	...	...	...	...	...	...
9995	-0.429820	0.855358	-1.381548	0.192047	-0.487525	0.650217	0.705765	-0.612461
9996	0.126784	1.076607	1.622691	-0.894380	0.105476	1.300475	1.114312	2.380307
9997	-0.441353	0.259332	-0.412438	0.891569	-0.414049	1.356958	1.359597	0.358695
9998	0.990676	-0.708494	-0.509349	-0.378271	0.964820	1.098585	1.069727	-0.787624
9999	0.323698	0.726295	0.798948	0.778133	0.210377	1.383429	1.181213	-0.197384

10000 rows × 8 columns

Part IV: Analysis

D. PCA Performance

1) The PCA syntax and matrix created are provided as screen shots below:



```python
#Create an pca instance
pca=PCA()
pc= pca.fit_transform(cont_std)
pc_df= pd.DataFrame(pc, columns=['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8'])
pc_df
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
0	-1.109593	0.651893	-0.880131	0.737566	1.525935	0.091127	-0.566029	0.098145
1	-0.814662	0.546376	-1.292409	-0.228375	0.583492	0.708165	-0.606740	0.123933
2	-1.649047	0.689401	0.969270	-0.636872	-0.635084	-0.192318	-0.501399	-0.161873
3	-1.804637	1.048508	0.744546	0.479089	-0.619186	-0.492171	0.844706	-0.119483
4	-2.145703	-1.792673	-0.724168	-1.034678	-1.215172	-0.219533	-0.087901	-0.135660
...	...	...	...	...	...	...	...	...
9995	0.773657	-1.515386	-0.959924	0.314325	-0.277746	-0.313721	-0.551542	0.025645
9996	2.043313	2.628520	-0.545463	-0.766477	-0.627316	-0.731947	-0.536649	-0.177726
9997	1.886872	-0.289870	-0.595204	0.903734	0.320697	-0.035317	-0.545108	-0.020843
9998	1.410539	-1.087100	1.274091	-0.829783	0.466517	-0.214954	0.203083	-0.004829
9999	1.822673	0.194804	-0.336535	0.352566	0.577527	-0.906947	0.708427	-0.113366

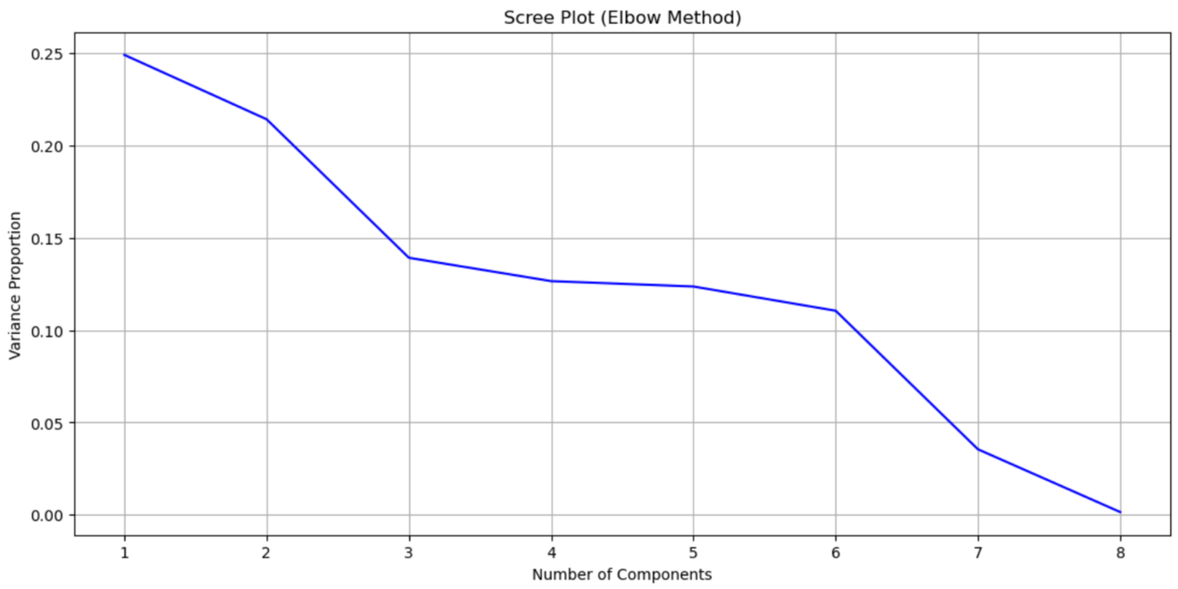
10000 rows × 8 columns

```python
#Create loadings matrix
loadings=pd.DataFrame(pca.components_.T, columns=['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8'], index=cont_variables.columns).T
loadings
```

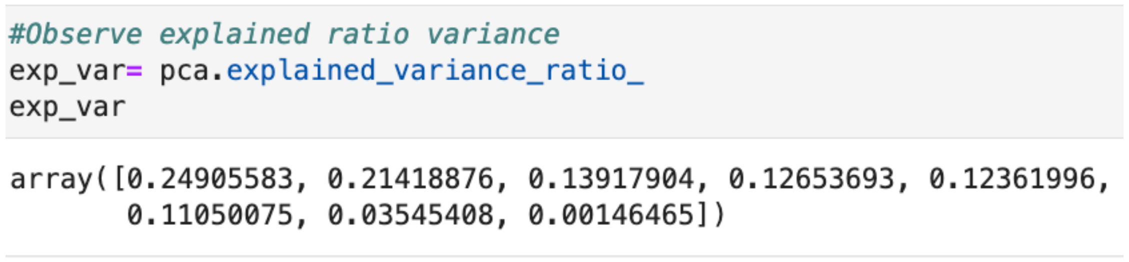
	Lat	Lng	Age	Income	VitD_levels	Initial_days	TotalCharge	Additional_charges
PC1	-0.013079	-0.010528	0.085809	-0.020516	-0.001636	0.701050	0.702184	0.086013
PC2	-0.008216	0.010934	0.701541	-0.018690	0.019146	-0.091424	-0.080868	0.701455
PC3	0.707443	-0.698601	0.002471	-0.087846	0.059987	-0.000290	-0.001272	0.012542
PC4	0.010501	-0.137959	0.017929	0.725834	-0.673096	0.007525	0.005167	0.023631
PC5	-0.093762	-0.114966	-0.000156	0.661202	0.735289	0.008485	0.009042	0.000546
PC6	-0.700262	-0.692410	-0.003099	-0.165777	-0.048152	-0.014659	-0.013262	-0.000853
PC7	0.003858	-0.006681	0.706716	0.002308	-0.001932	0.031677	-0.031469	-0.706038
PC8	0.001483	-0.000357	0.026313	0.001312	-0.001552	-0.706274	0.706486	-0.036819

2) Using an elbow scree plot, we can see the total number of components for analysis, using the explained_variance_ratio values on the y-axis and the indexes for number of principal components in the x-axis. There are a total of 8 principal components.

```python
#Construct Elbow plot to confirm optimal number of components
plt.figure(figsize=(13,6))
plt.plot(pcomp, exp_var, 'b-')
plt.title('Scree Plot (Elbow Method)')
plt.xlabel('Number of Components')
plt.ylabel('Variance Proportion')
plt.grid()
plt.show()
```



3) The variance of each principal component can been in the screen shot below:

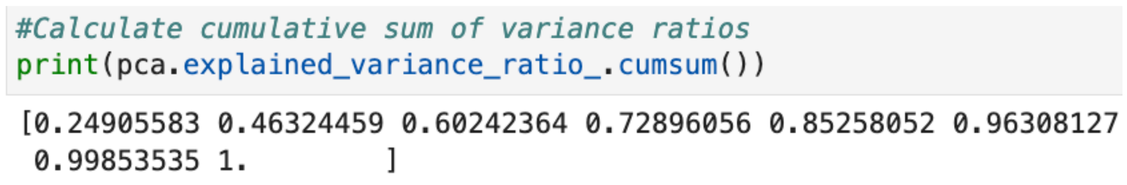


```
#Observe explained ratio variance
exp_var= pca.explained_variance_ratio_
exp_var

array([0.24905583, 0.21418876, 0.13917904, 0.12653693, 0.12361996,
       0.11050075, 0.03545408, 0.00146465])
```

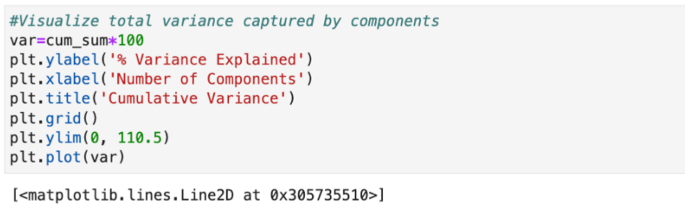
1. Principal component 1 explains 24.9% of the total variance.
2. Principal component 2 explains 21.4% of the total variance.
3. Principal component 3 explains 13.9% of the total variance.
4. Principal component 4 explains 12.6% of the total variance.
5. Principal component 5 explains 12.3% of the total variance.
6. Principal component 6 explains 11% of the total variance.
7. Principal component 7 explains 3.5% of the total variance.
8. Principal component 8 explains 0.14% of the total variance.

4) The total variance of components from section D2 is shown in the screen shots below:



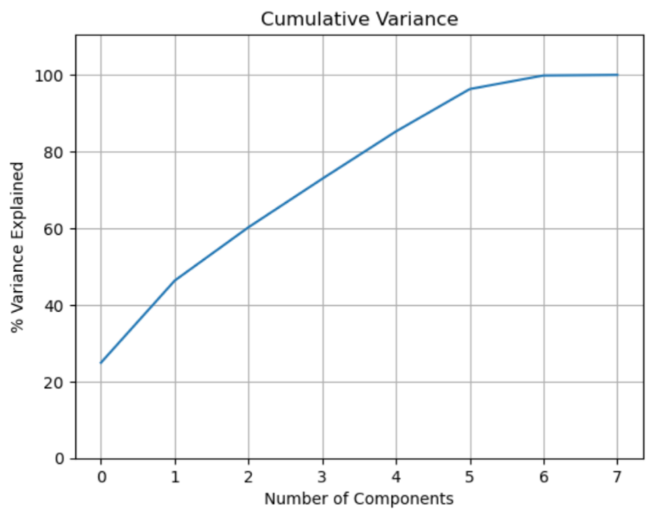
```
#Calculate cumulative sum of variance ratios
print(pca.explained_variance_ratio_.cumsum())

[0.24905583 0.46324459 0.60242364 0.72896056 0.85258052 0.96308127
 0.99853535 1.        ]
```



```
#Visualize total variance captured by components
var=cum_sum*100
plt.ylabel('% Variance Explained')
plt.xlabel('Number of Components')
plt.title('Cumulative Variance')
plt.grid()
plt.ylim(0, 110.5)
plt.plot(var)

[<matplotlib.lines.Line2D at 0x305735510>]
```



5) Performing PCA on the medical data set allowed me to determine the most optimal principal components for data analysis on patients. Specifically, continuous variables were selected for PCA and began with 8 principal components. Looking at the cumulative scree plot above, we can see that the first six principal components are responsible for 96% of the variance in the data set. However, we can still explain up to 72% in the first four principal components or up to 85% in the first five. Therefore, an analyst could use dimensional reduction to just four or five components without losing too much information. After running the PCA model, I would reduce the components to the first five as I feel enough information is retained and could be used for analysis.

Part V: Sources

Whitfield, B., & Galarnyk, M. (2024, February 23). *PCA using python: A tutorial*. Built In. https://builtin.com/machine-learning/pca-in-python#

Aggarwal, T. (2023, July 31). *Harnessing the essence: Principal component analysis (PCA)*. Medium. https://medium.com/@tushar_aggarwal/harnessing-the-essence-principal-component-analysis-pca-72bfd99ab1ea

Av content team. (2024, April 4). *PCA: What is Principal Component Analysis & How It Works? (updated 2024)*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2016/03/pca-practical-guide-principal-component-analysis-python/