D212: Data Mining II – Task 2 Dimensionality Reduction Methods

Part I: Research Question

A1. Research Question

Can PCA be used to reduce dimensionality of the provided dataset?

A2. Goal

The goal is to use PCA to reduce our large dataset into a smaller one while keeping the significant variables.

Part II: Method Justification

B1. PCA Explained

Principal Component Analysis is a dimensionality reduction method used to transform a large dataset into a smaller set of variables, known as principal components, while retaining as much information as possible. Rather than simply selecting a few original variables and discarding the rest, PCA creates new composite variables that summarize the dataset. The significance of each principal component is evaluated using eigenvalues and eigenvectors. The eigenvalues indicate the variance captured by each principal component, while the corresponding eigenvectors represent the directions of these variances in the transformed matrix. If some principal components are found to be significant enough through this analysis, they are retained, resulting in a reduced dimensional representation of the original dataset. This process highlights the effectiveness of PCA in simplifying larger datasets without losing critical information (Brems, 2022).

The expected outcome of this analysis is to reduce dimensionality by transforming the provided large dataset into a smaller one, provide significant variables (principal components) and show a visualization of how the variables are correlated (Whitfield, 2024).

B2. Assumption of PCA

One assumption of PCA is that there is a correlation between the variables. If there wasn't a correlation between the variables, PCA wouldn't be able to determine what the principal components are.

Part III: Data Preparation

C1. Variables

Variable Name	Continuous or Categorical
Population	Continuous
Outage_sec_perweek	Continuous

Email	Continuous
Contacts	Continuous
Yearly_equip_failure	Continuous
Tenure	Continuous
MonthlyCharge	Continuous
Bandwidth_GB_Year	Continuous

C2. <u>Cleaned Dataset</u>

Attached as “D212_Task2.csv”

In [11]:
```
# Normalize the data by using standardization
scaler = StandardScaler()
scaled_df = pd.DataFrame(scaler.fit_transform(df), columns = df.columns)
scaled_df.head()
```

Out[11]:

	Population	Outage_sec_perweek	Email	Contacts	Yearly_equip_failure	Tenure	MonthlyCharge	Bandwidth_GB_Year
0	-0.673405	-0.679978	-0.666282	-1.005852	0.946658	-1.048746	-0.003943	-1.138487
1	0.047772	0.570331	-0.005288	-1.005852	0.946658	-1.262001	1.630326	-1.185876
2	-0.417238	0.252347	-0.996779	-1.005852	0.946658	-0.709940	-0.295225	-0.612138
3	0.284537	1.650506	0.986203	1.017588	-0.625864	-0.659524	-1.226521	-0.561857
4	0.110549	-0.623156	1.316700	1.017588	0.946658	-1.242551	-0.528086	-1.428184

Part IV: Analysis

D1. <u>Principal Components</u>

In [8]:
```
# Set the size of Principal Components (PCs)
pca = PCA(n_components=scaled_df.shape[1])
pca.fit_transform(scaled_df)
```

Out[8]:
```
array([[-1.51892308,  1.60028534, -0.4523488 , ..., -0.08091597,
         0.33172014, -0.06471949],
       [-1.64726761, -0.19569971, -1.16985152, ...,  0.05424105,
         0.59376954, -0.02164869],
       [-0.90873505,  1.33573398, -0.77372642, ...,  0.1390508 ,
        -0.44955339,  0.08152433],
       ...,
       [ 0.58521805,  1.2654814 ,  0.38099736, ...,  0.46821967,
        -0.06014122, -0.09310407],
       [ 2.0154624 , -2.21370971,  0.19807666, ...,  1.03924796,
         0.59469242, -0.06499388],
       [ 1.56615756, -1.73520578,  0.35112647, ..., -0.74056348,
         0.15772066, -0.01868389]])
```

In [9]:
```
# Generate the PCA loading matrix, the variables that contribute the most to the PCs
loadings = pd.DataFrame(pca.components_.T,
                        columns=['PC1','PC2','PC3','PC4','PC5','PC6','PC7','PC8'],
                        index=scaled_df.columns)

# Print the loading matrix
loadings
```

Out[9]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
Population	-0.005903	-0.344915	0.450352	0.499984	-0.274235	0.577916	0.137865	0.000027
Outage_sec_perweek	0.005890	-0.474455	-0.469356	0.279187	-0.215157	-0.106576	-0.647277	0.000048
Email	-0.020914	-0.457599	0.472623	-0.069708	-0.196284	-0.708907	0.144422	0.000188
Contacts	0.004417	-0.392605	-0.199278	0.314783	0.775157	-0.059891	0.320292	-0.000152
Yearly_equip_failure	0.017454	0.366576	-0.310883	0.624915	-0.336207	-0.332706	0.392966	-0.000056
Tenure	0.705576	0.014578	0.043649	0.016768	0.020054	-0.015216	-0.032897	-0.705725
MonthlyCharge	0.040592	-0.397095	-0.463750	-0.420975	-0.354491	0.193875	0.532048	-0.045358
Bandwidth_GB_Year	0.706883	-0.010812	0.013639	-0.010173	-0.002508	-0.002616	0.001397	0.707033

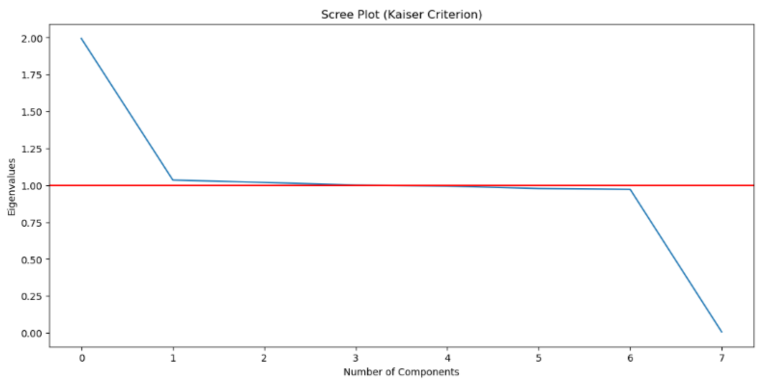
In [10]:
```
# Calculate the covariance and vectors then define the eigenvalues
cov_matrix = np.dot(scaled_df.T, scaled_df) / df.shape[0]
eigenvalues = [np.dot(eigenvector.T, np.dot(cov_matrix, eigenvector)) for eigenvector in pca.components_]

print(cov_matrix)
```

```
[[ 1.          0.00548327  0.01796155  0.00401876 -0.0044829  -0.00355944
  -0.00477827 -0.00390183]
 [ 0.00548327  1.          0.00399373  0.01509168  0.00290873  0.00293196
   0.02049607  0.00417566]
 [ 0.01796155  0.00399373  1.          0.00304036 -0.01635434 -0.01446788
   0.00199655 -0.01457915]
 [ 0.00401876  0.01509168  0.00304036  1.         -0.00603225  0.00282009
   0.00425865  0.00329872]
 [-0.0044829   0.00290873 -0.01635434 -0.00603225  1.          0.01243491
  -0.00717228  0.01203369]
 [-0.00355944  0.00293196 -0.01446788  0.00282009  0.01243491  1.
  -0.00333681  0.99149519]
 [-0.00477827  0.02049607  0.00199655  0.00425865 -0.00717228 -0.00333681
   1.          0.06040643]
 [-0.00390183  0.00417566 -0.01457915  0.00329872  0.01203369  0.99149519
   0.06040643  1.        ]]
```

D2. Total Number of Components

According to the Kaiser criterion, to retain a principal component, the eigenvalue must be greater than 1. I have displayed the eigenvalues in the screenshot below to ensure the correct number of PCs have been kept. There are four components.

```
In [11]: # Create a scree plot to identify which PCs to keep
plt.figure(figsize = [13,6])
plt.plot(eigenvalues)
plt.title('Scree Plot (Kaiser Criterion)')
plt.xlabel('Number of Components')
plt.ylabel('Eigenvalues')
plt.axhline(y=1, color='red')
plt.show()
```




```
In [12]: # Print the eigenvalues to ensure the number of PCs to keep (values greater than 1)
eigenvalues

Out[12]: [1.9939477111706485,
 1.0352380857299903,
 1.0189087404723431,
 1.0014679192636666,
 0.99435052787904 7,
 0.9776288051387267,
 0.9719999630069431,
 0.006458247338637216]
```

D3. Explained Variance

```
In [13]: # Get the explaine variance per principal component as percentages
captured_variance = pca.explained_variance_ratio_ * 100

# Print the captured variance for each component
for i, var in enumerate(captured_variance):
    print(f"Principal Component {i+1}: {var:.2f}%")

Principal Component 1: 24.92%
Principal Component 2: 12.94%
Principal Component 3: 12.74%
Principal Component 4: 12.52%
Principal Component 5: 12.43%
Principal Component 6: 12.22%
Principal Component 7: 12.15%
Principal Component 8: 0.08%
```

D4. Total Variance

```
In [14]: # Print the total variance captured by the PCs
total_variance_captured = np.sum(pca.explained_variance_ratio_[:4])
total_variance_captured

Out[14]: 0.6311953070795806
```

D5. Summary

This data analysis used PCA to reduce the dimensionality of the provided dataset. I started with eight continuous variables. Then I created a loading matrix to represent the correlation between the original dataset variables versus the transformed principal components (PCs). Using the Kaiser criterion, I identified the four PCs that explained 63.12% variance of the original dataset. To determine which components to retain, you have to examine the scree plot and only keep the eigenvalues greater than 1. According to the graph, only PC1 through PC4 would be kept. The other values would be discarded since those values are less than 1.

Part V: Attachments

E. Sources for Third-Party Code

N/A

F. Sources

Brems, M. (2022, January 26). A one-stop shop for principal component analysis. Medium. https://medium.com/towards-data-science/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c

Whitfield, B. (2024, February 23). Principal Component Analysis (PCA) explained. Built In. https://builtin.com/data-science/step-step-explanation-principal-component-analysis

G. Professional Communication