Data Mining

2-Aug-2023

Chetan R Deshpande

Overview

In this project, we discover that PCA reveals essential patterns and reduces data dimensions, aiding visualization. Clustering groups similar data points, uncovering inherent structures. Combined, they improve data comprehension, decision-making, and problem-solving across diverse domains.

Contents

PCA

1.1 Perform Exploratory Data Analysis [both univariate and multivariate analysis to be performed]. The inferences drawn from this should be properly documented.

Firstly, we read the dataset and print the head to understand the data from the top 5 rows.

	ID	ProdQual	Ecom	TechSup	CompRes	Advertising	ProdLine	SalesFImage	ComPricing	WartyClaim	OrdBilling	DelSpeed	Satisfaction
0	1	8.5	3.9	2.5	5.9	4.8	4.9	6.0	6.8	4.7	5.0	3.7	8.2
1	2	8.2	2.7	5.1	7.2	3.4	7.9	3.1	5.3	5.5	3.9	4.9	5.7
2	3	9.2	3.4	5.6	5.6	5.4	7.4	5.8	4.5	6.2	5.4	4.5	8.9
3	4	6.4	3.3	7.0	3.7	4.7	4.7	4.5	8.8	7.0	4.3	3.0	4.8
4	5	9.0	3.4	5.2	4.6	2.2	6.0	4.5	6.8	6.1	4.5	3.5	7.1

Secondly, we check for the data types, shape, null values using info().

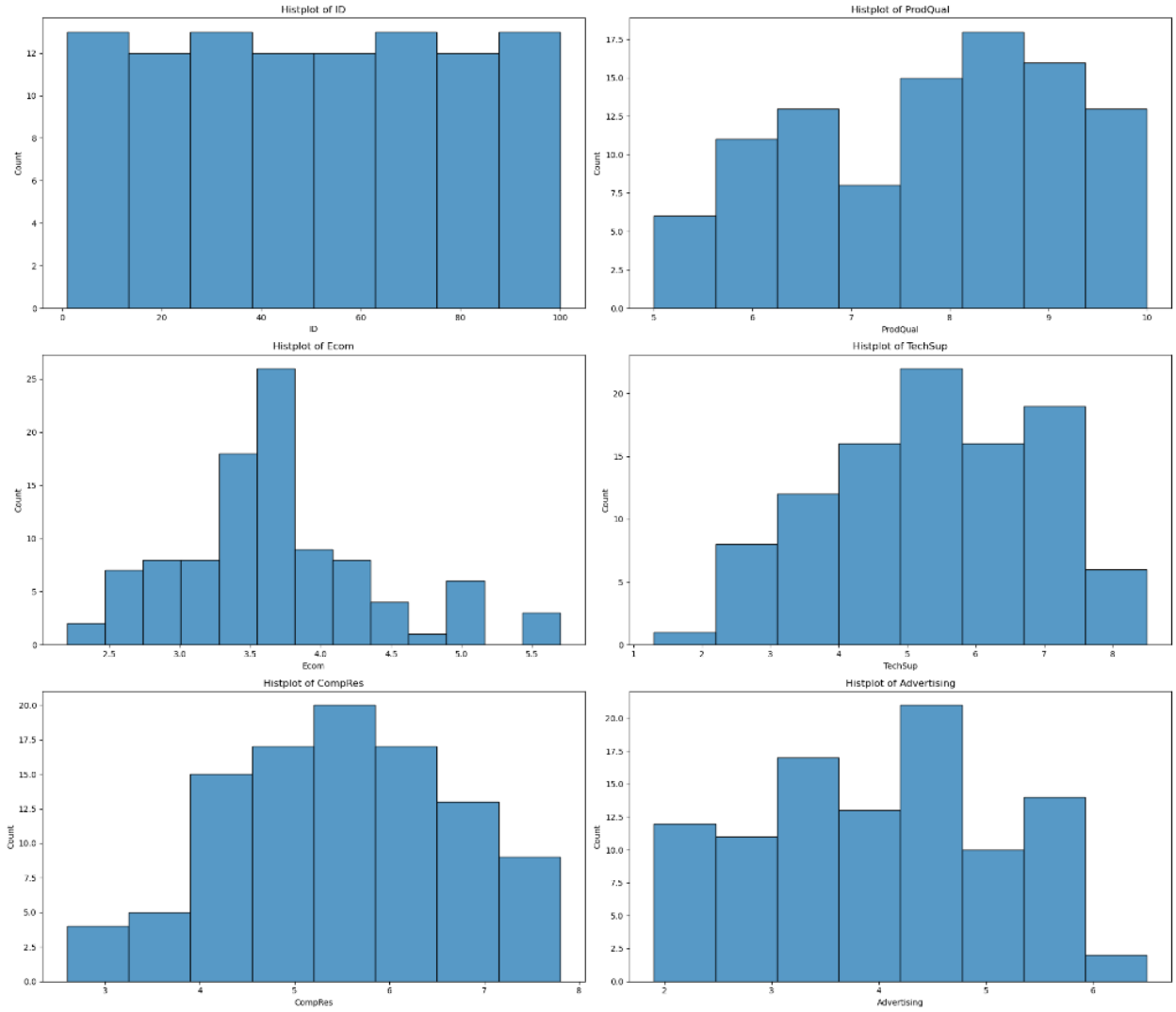
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   ID            100 non-null    int64  
 1   ProdQual      100 non-null    float64
 2   Ecom          100 non-null    float64
 3   TechSup       100 non-null    float64
 4   CompRes       100 non-null    float64
 5   Advertising   100 non-null    float64
 6   ProdLine      100 non-null    float64
 7   SalesFImage   100 non-null    float64
 8   ComPricing    100 non-null    float64
 9   WartyClaim    100 non-null    float64
 10  OrdBilling    100 non-null    float64
 11  DelSpeed      100 non-null    float64
 12  Satisfaction  100 non-null    float64
dtypes: float64(12), int64(1)
memory usage: 10.3 KB
```

Now, using describe(), we will print the statistical description.

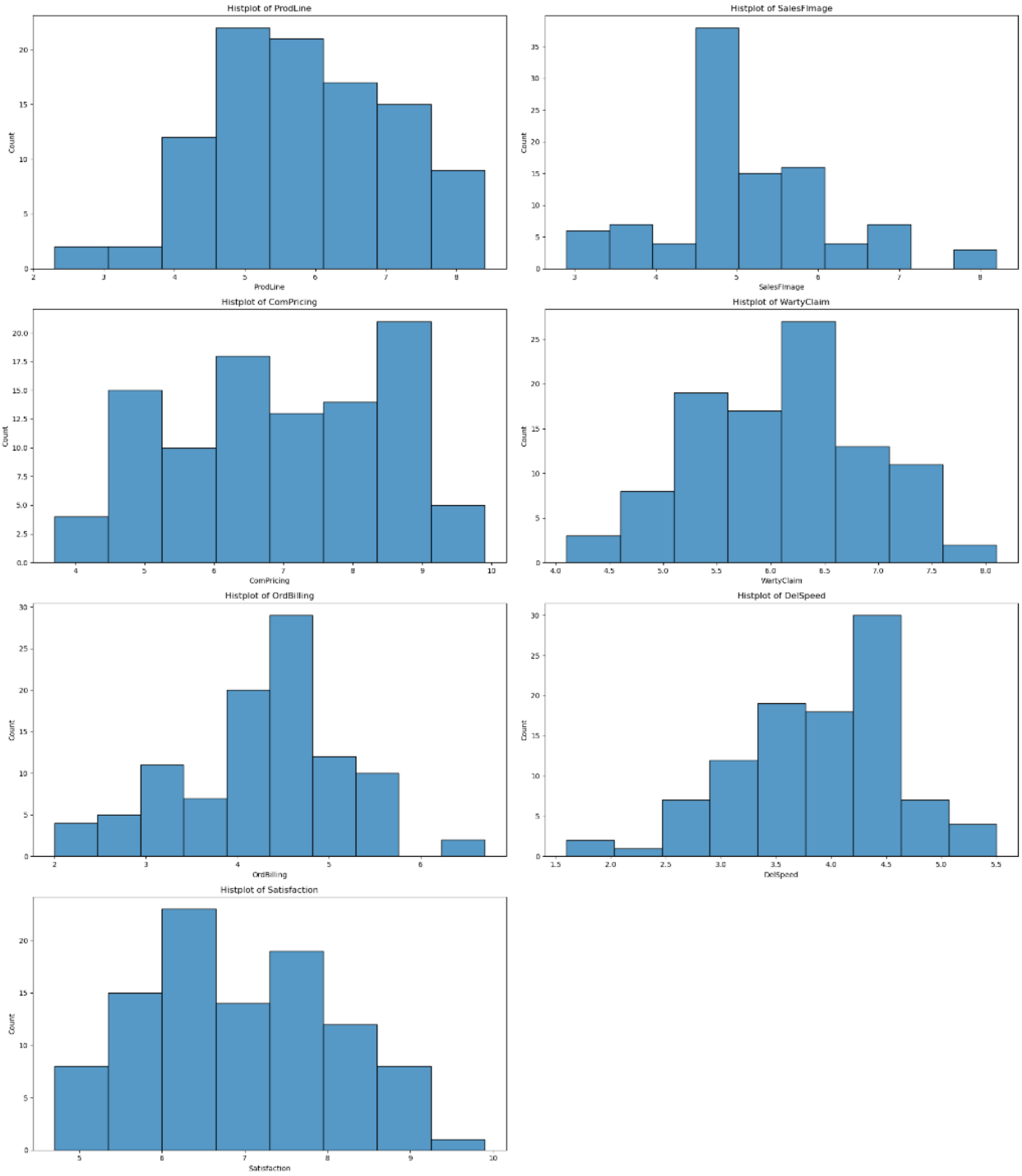
	count	mean	std	min	25%	50%	75%	max
ID	100.0	50.500	29.011492	1.0	25.750	50.50	75.250	100.0
ProdQual	100.0	7.810	1.396279	5.0	6.575	8.00	9.100	10.0
Ecom	100.0	3.672	0.700516	2.2	3.275	3.60	3.925	5.7
TechSup	100.0	5.365	1.530457	1.3	4.250	5.40	6.625	8.5
CompRes	100.0	5.442	1.208403	2.6	4.600	5.45	6.325	7.8
Advertising	100.0	4.010	1.126943	1.9	3.175	4.00	4.800	6.5
ProdLine	100.0	5.805	1.315285	2.3	4.700	5.75	6.800	8.4
SalesFImage	100.0	5.123	1.072320	2.9	4.500	4.90	5.800	8.2
ComPricing	100.0	6.974	1.545055	3.7	5.875	7.10	8.400	9.9
WartyClaim	100.0	6.043	0.819738	4.1	5.400	6.10	6.600	8.1
OrdBilling	100.0	4.278	0.928840	2.0	3.700	4.40	4.800	6.7
DelSpeed	100.0	3.886	0.734437	1.6	3.400	3.90	4.425	5.5
Satisfaction	100.0	6.918	1.191839	4.7	6.000	7.05	7.625	9.9

Further, using isnull() and duplicated() we can check for null values and duplicated data which are 0 in this dataset.

Now, we will perform multivariate analysis -

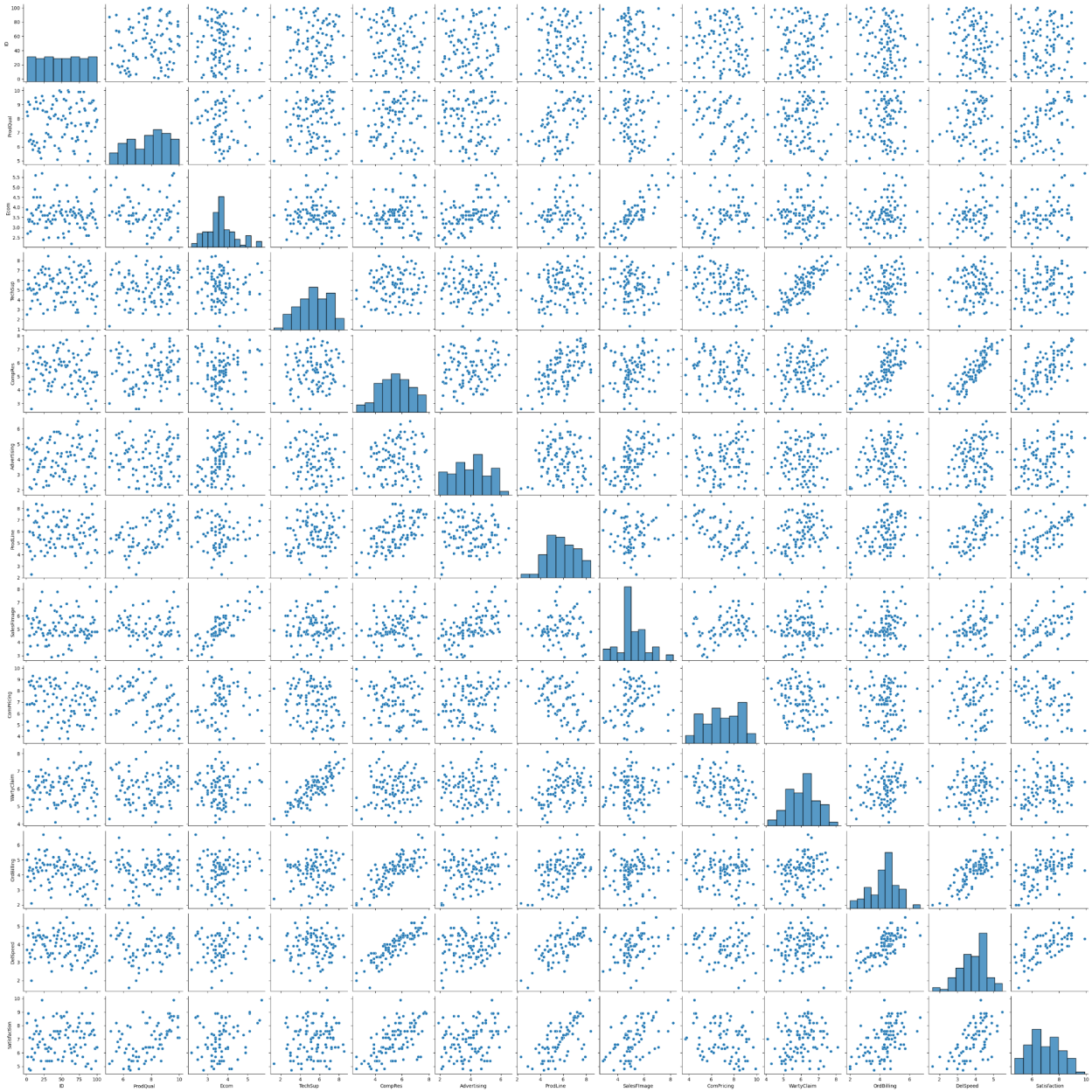


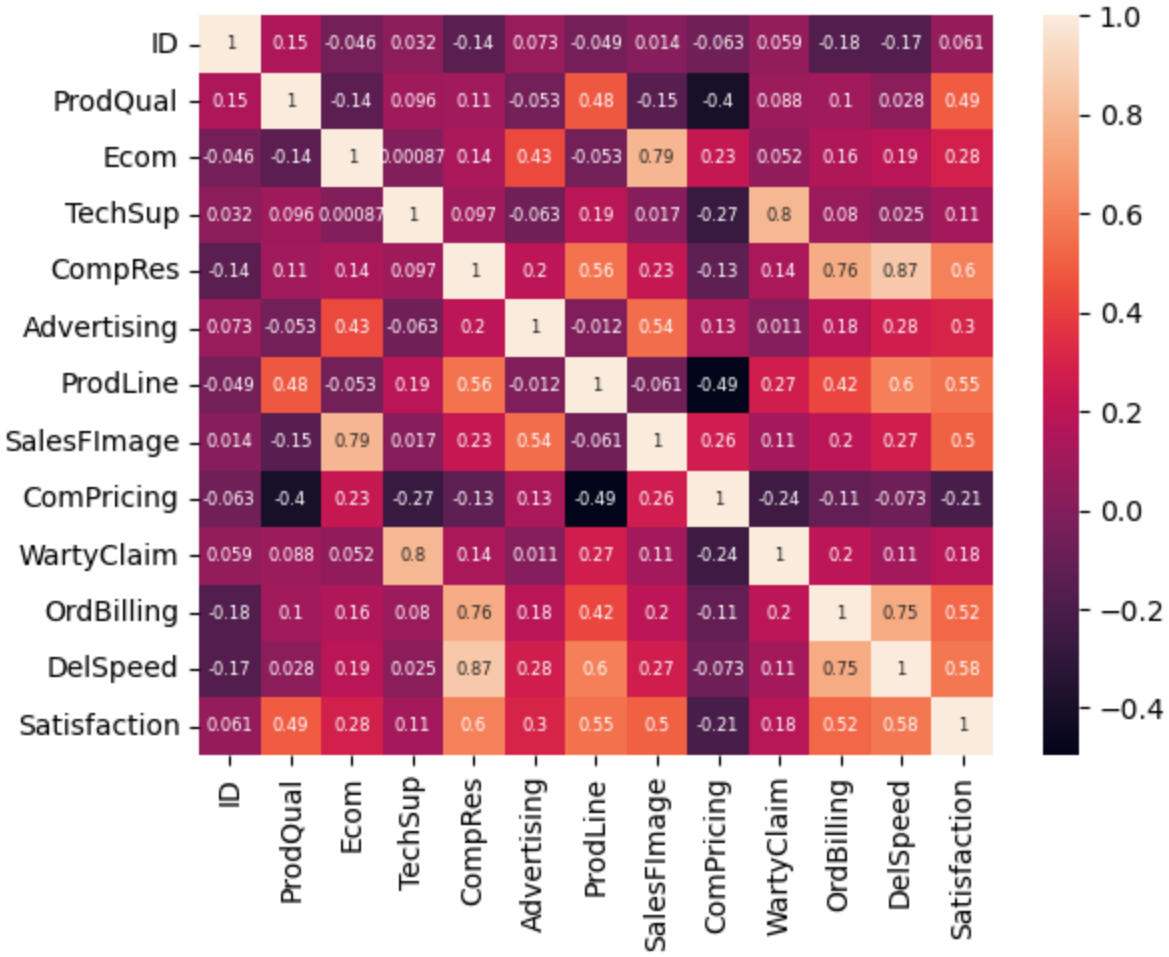
- In the histogram for Product Quality, the rating between 8-9 stands highest, for E-Commerce 3.5-4 has highest counts, for Technical Support 5-6 stands high, for Complaint Resolution 3.5-4 and for Advertising 5-6 has repeated most times.



Here, from the top, Product Line has the highest counts of 5, Salesforce Image maintaining a 5 as well, Competitive Pricing has counts of 9 as the highest, Warranty and Claims 6.5, Order & Billing 4.5, Delivery Speed 4.5 and Satisfaction has 6.5 as highest.

Next, multivariate analysis -





The above heatmap shows the correlation between the variables. The highest correlation in the heatmap plotted is for Complaint resolution to Delivery speed - 0.87.

1.2Scale the variables and write the inference for using the type of scaling function for this case study.

- To scale the data we will import zscore from scipy.stats and apply it to the existing data to obtain the scaled data.
- To get the final data to proceed further, we will drop the non-useful variables - ID and Satisfaction.

Below is the final data ready to perform PCA.

	ProdQual	Ecom	TechSup	CompRes	Advertising	ProdLine	SalesFImage	ComPricing	WartyClaim	OrdBilling	DelSpeed
0	0.496660	0.327114	-1.881421	0.380922	0.704543	-0.691530	0.821973	-0.113185	-1.646582	0.781230	-0.254531
1	0.280721	-1.394538	-0.174023	1.462141	-0.544014	1.600835	-1.896068	-1.088915	-0.665744	-0.409009	1.387605
2	1.000518	-0.390241	0.154322	0.131410	1.239639	1.218774	0.634522	-1.609304	0.192489	1.214044	0.840226
3	-1.014914	-0.533712	1.073690	-1.448834	0.615361	-0.844354	-0.583910	1.187789	1.173327	0.023805	-1.212443
4	0.856559	-0.390241	-0.108354	-0.700298	-1.614207	0.149004	-0.583910	-0.113185	0.069885	0.240212	-0.528220

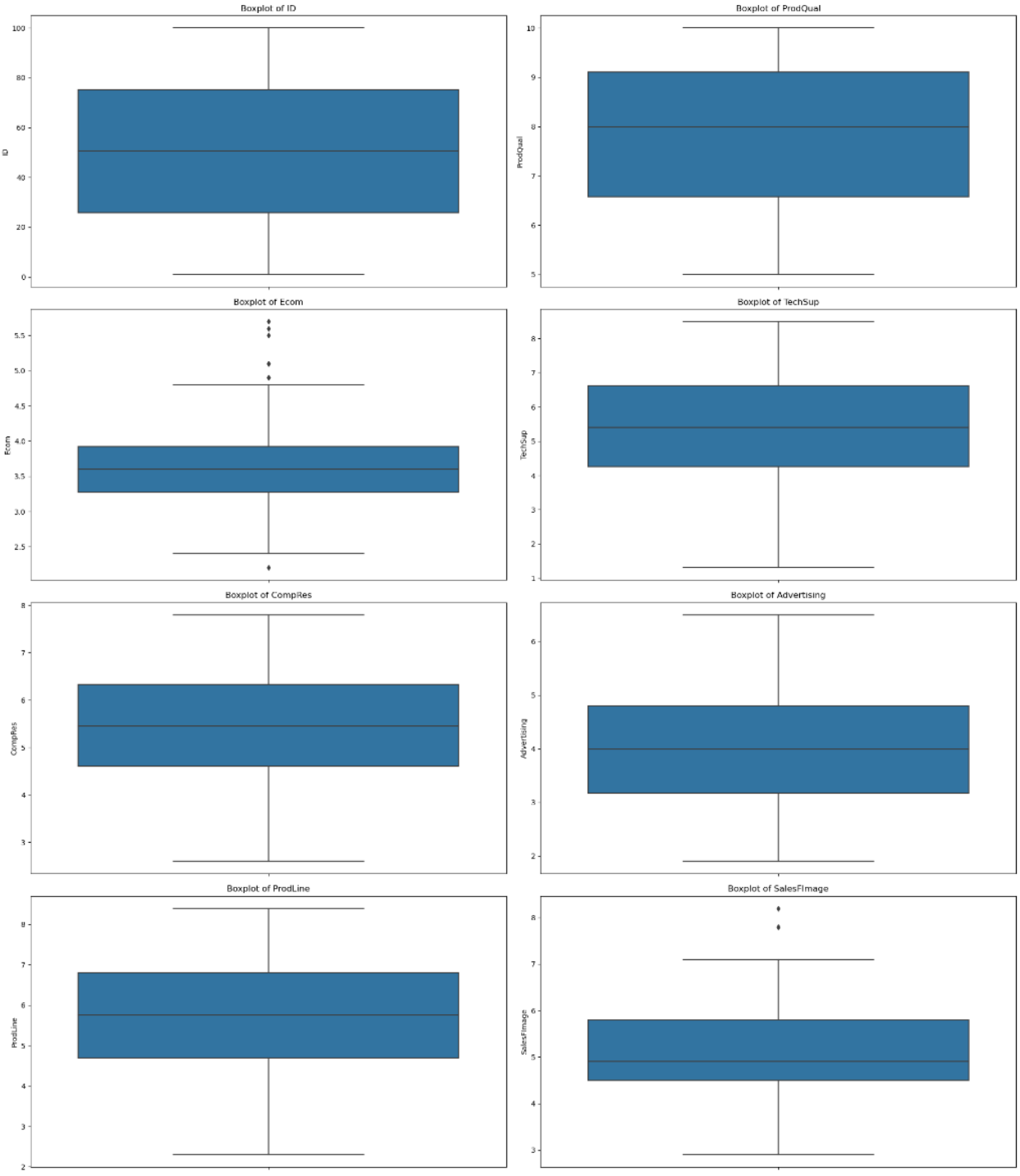
The inference for using the Z-score scaling function in this case study is that it standardizes the data by subtracting the mean and dividing by the standard deviation for each feature. This process ensures that all features have a mean of zero and a standard deviation of one. It helps in comparing and interpreting the features on a common scale, mitigating the influence of outliers, and improving the performance of various statistical and machine learning algorithms that are sensitive to feature scales. Moreover, Z-score scaling is particularly useful when dealing with data that may have different units or scales, promoting more robust and accurate analyses in the case study.

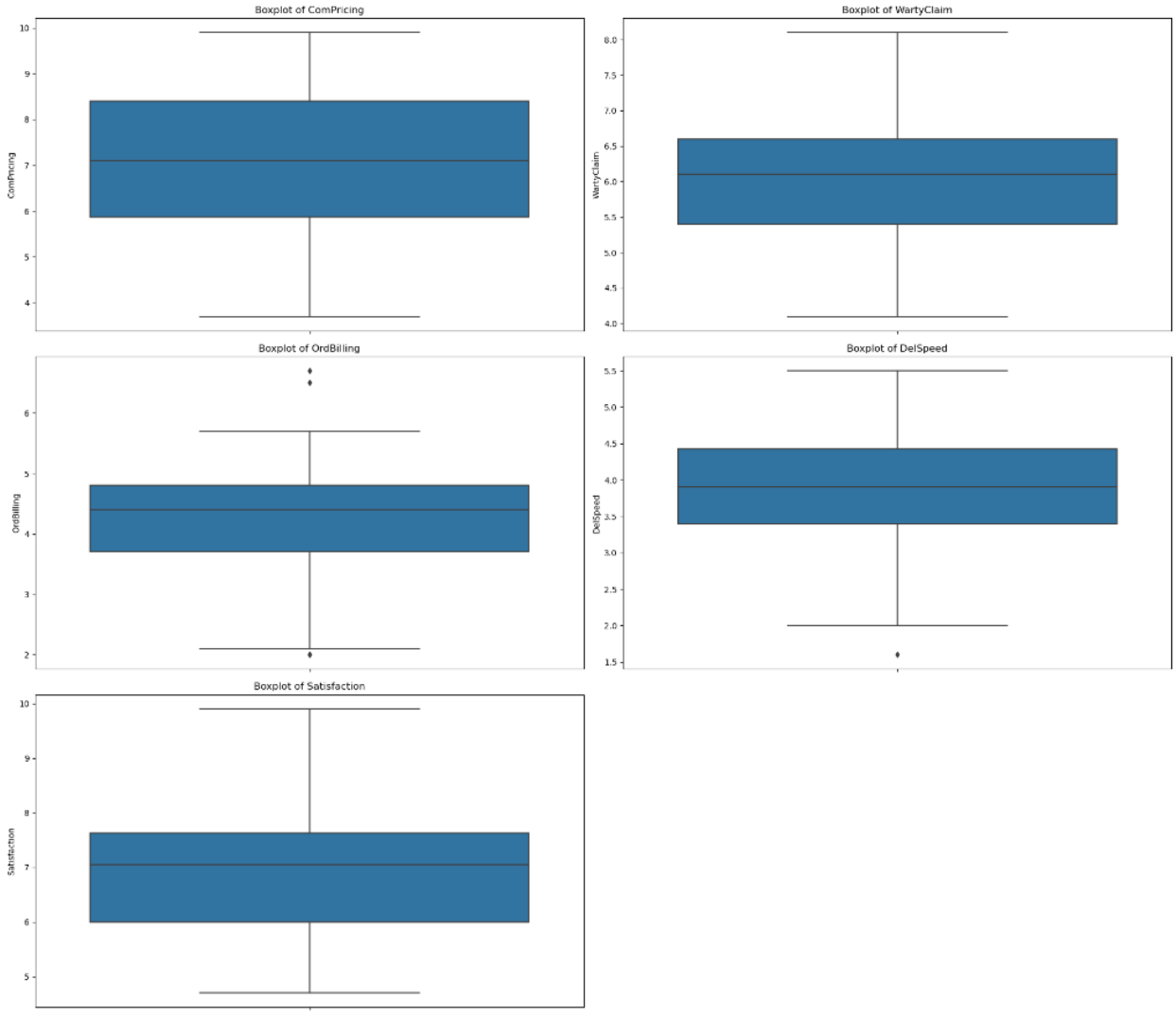
1.3 Comment on the comparison between covariance and the correlation matrix after scaling.

After scaling the data, both the covariance matrix and the correlation matrix provide insights into the relationships between variables. However, the key difference lies in their units of measurement. The covariance matrix displays the raw covariance values, reflecting the direction of relationships without standardization. On the other hand, the correlation matrix presents standardized values, ranging from -1 to +1, offering a more interpretable and comparable measure of the strength and direction of linear relationships. The correlation matrix is preferred when dealing with variables of different scales, as it eliminates scale discrepancies and facilitates a more meaningful analysis of the data.

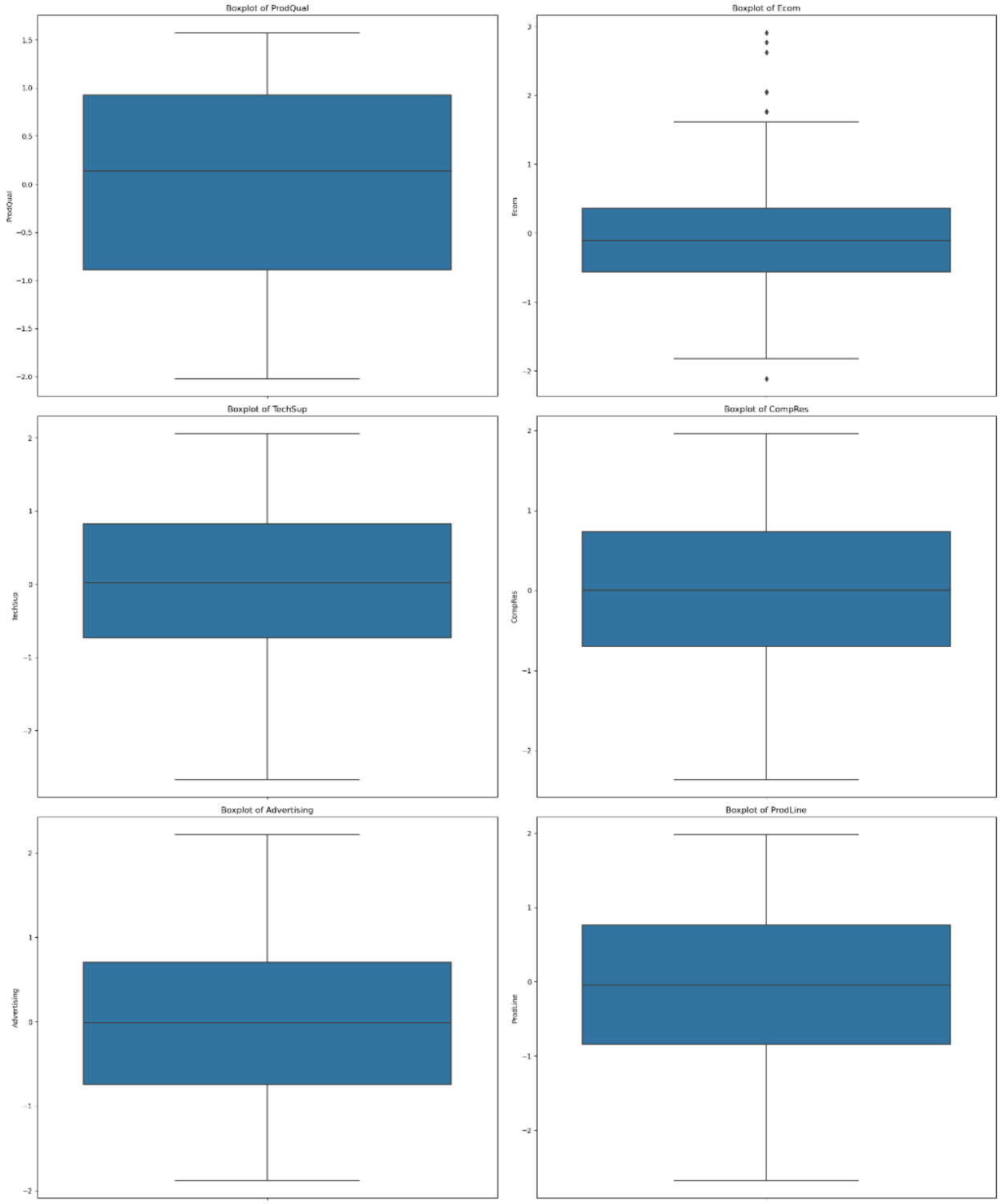
1.4 Check the dataset for outliers before and after scaling. Draw your inferences from this exercise.

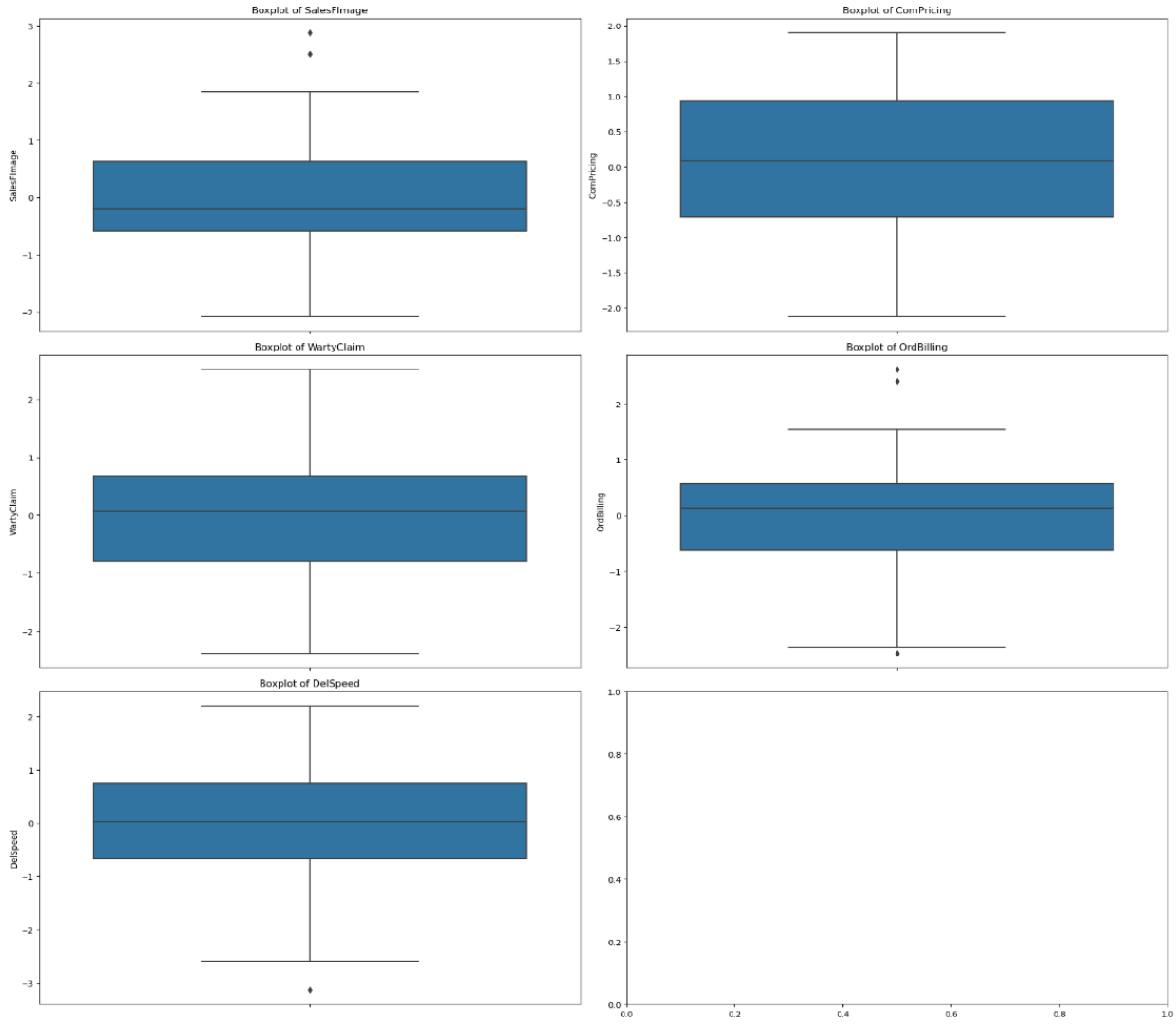
The following boxplots are from the data before scaling the data.




Boxplot of ComPricing
ComPricing
Boxplot of WartyClaim
WartyClaim
Boxplot of OrdBilling
OrdBilling
Boxplot of DelSpeed
DelSpeed
Boxplot of Satisfaction
Satisfaction

The following boxplots are from the data after scaling the data which seems to be the same as before.





After scaling the data, outliers remain unchanged in terms of their relative positions and values in the dataset. Scaling does not alter the presence or absence of outliers; it only affects the distribution and spread of the data. Outliers that existed before scaling will still exist after scaling, and their impact on the dataset, such as influencing means and standard deviations, will persist. Scaling helps standardize the data to a common scale but does not remove or modify the outliers themselves.

1.5 Build the covariance matrix, eigenvalues and eigenvector.

The following is the covariance matrix -

```
array([[ 1.01010101e+00, -1.38548704e-01,  9.65661154e-02,
         1.07444445e-01, -5.40132667e-02,  4.82316579e-01,
        -1.53346338e-01, -4.05335236e-01,  8.92043497e-02,
         1.05356640e-01,  2.79979825e-02],
       [-1.38548704e-01,  1.01010101e+00,  8.75544162e-04,
         1.41595213e-01,  4.34233041e-01, -5.32200387e-02,
         7.99539102e-01,  2.31780203e-01,  5.24224157e-02,
         1.57724577e-01,  1.93571786e-01],
       [ 9.65661154e-02,  8.75544162e-04,  1.01010101e+00,
         9.76329270e-02, -6.35051180e-02,  1.94571168e-01,
         1.71621612e-02, -2.73521901e-01,  8.05220127e-01,
         8.09109340e-02,  2.56976702e-02],
       [ 1.07444445e-01,  1.41595213e-01,  9.76329270e-02,
         1.01010101e+00,  1.98905906e-01,  5.67087831e-01,
         2.32072486e-01, -1.29246720e-01,  1.41826562e-01,
         7.64513729e-01,  8.73829997e-01],
       [-5.40132667e-02,  4.34233041e-01, -6.35051180e-02,
         1.98905906e-01,  1.01010101e+00, -1.16674936e-02,
         5.47680463e-01,  1.35572620e-01,  1.09010852e-02,
         1.86096560e-01,  2.78649579e-01],
       [ 4.82316579e-01, -5.32200387e-02,  1.94571168e-01,
         5.67087831e-01, -1.16674936e-02,  1.01010101e+00,
        -6.19348764e-02, -4.99947880e-01,  2.75835887e-01,
         4.28695202e-01,  6.07929503e-01],
       [-1.53346338e-01,  7.99539102e-01,  1.71621612e-02,
         2.32072486e-01,  5.47680463e-01, -6.19348764e-02,
         1.01010101e+00,  2.67269246e-01,  1.08540752e-01,
         1.97098390e-01,  2.74294201e-01],
       [-4.05335236e-01,  2.31780203e-01, -2.73521901e-01,
```

-1.29246720e-01, 1.35572620e-01, -4.99947880e-01,
2.67269246e-01, 1.01010101e+00, -2.47460661e-01,
-1.15724268e-01, -7.36078070e-02],
[8.92043497e-02, 5.24224157e-02, 8.05220127e-01,
1.41826562e-01, 1.09010852e-02, 2.75835887e-01,
1.08540752e-01, -2.47460661e-01, 1.01010101e+00,
1.99055678e-01, 1.10499598e-01],
[1.05356640e-01, 1.57724577e-01, 8.09109340e-02,
7.64513729e-01, 1.86096560e-01, 4.28695202e-01,
1.97098390e-01, -1.15724268e-01, 1.99055678e-01,
1.01010101e+00, 7.58588957e-01],
[2.79979825e-02, 1.93571786e-01, 2.56976702e-02,
8.73829997e-01, 2.78649579e-01, 6.07929503e-01,
2.74294201e-01, -7.36078070e-02, 1.10499598e-01,
7.58588957e-01, 1.01010101e+00]])

The following is the EigenVector -

array([[-0.13378962, -0.16595278, -0.15769263, -0.47068359, -0.18373495,
-0.38676517, -0.2036696 , 0.15168864, -0.21293363, -0.43721774,
-0.47308914],
[-0.31349802, 0.44650918, -0.23096734, 0.01944394, 0.36366471,
-0.28478056, 0.47069599, 0.4134565 , -0.19167191, 0.02639905,
0.07305172],
[0.06227164, -0.23524791, -0.61095105, 0.21035078, -0.08809705,
0.11627864, -0.2413421 , 0.05304529, -0.59856398, 0.16892981,
0.23262477],
[0.6431362 , 0.27238033, -0.19339314, -0.20632037, 0.31789448,
0.20290226, 0.22217722, -0.33354348, -0.18530205, -0.23685365,
-0.1973299],
[0.2316662 , 0.42228844, -0.02395667, 0.02865743, -0.80387024,
0.11667416, 0.20437283, 0.24892601, -0.03292706, 0.02675377,
-0.03543294],
[-0.56456996, 0.26325703, -0.10876896, -0.02815231, -0.20056937,

0.09819533, 0.10497225, -0.70973595, -0.13983966, -0.11947974,
0.02979992],
[0.19164132, 0.05962621, -0.01719992, -0.0084996 , -0.06306962,
-0.60814755, 0.00143735, -0.30824887, -0.03064024, 0.65931989,
-0.23423927],
[0.13547311, -0.12202642, 0.46470964, 0.51339754, -0.05347713,
-0.3332071 , 0.16910665, -0.09883227, -0.4435404 , -0.36601754,
0.06539059],
[0.0313281 , -0.54251104, -0.35929961, 0.09324751, -0.15468169,
-0.08415534, 0.64489911, -0.09414389, 0.31756604, -0.09907265,
-0.02188514],
[0.06659717, 0.28155772, -0.3881709 , 0.53467243, 0.03715799,
-0.23479794, -0.35341191, -0.04518224, 0.43534752, -0.30386545,
-0.12010386],
[0.18279209, 0.06233863, -0.05192956, -0.36253352, -0.08118684,
-0.38507778, -0.08469869, -0.10295751, 0.12893245, -0.19415064,
0.77563222]])

The following is the EigenValues -

array([3.4615872 , 2.57666335, 1.70805705, 1.09753137, 0.61557989,
0.55745836, 0.40557389, 0.249446 , 0.20560936, 0.13418341,
0.09942123])

1.6 Write the explicit form of the first PC (in terms of EigenVectors).

PC1 = -0.13378962 * 3.4615872 - 0.16595278 * 2.57666335 - 0.15769263 * 1.70805705 - 0.47068359 * 1.09753137 - 0.18373495 * 0.61557989 - 0.38676517 * 0.55745836 - 0.2036696 * 0.40557389 + 0.15168864 * 0.249446 - 0.21293363 * 0.20560936 - 0.43721774 * 0.13418341 - 0.47308914 * 0.09942123

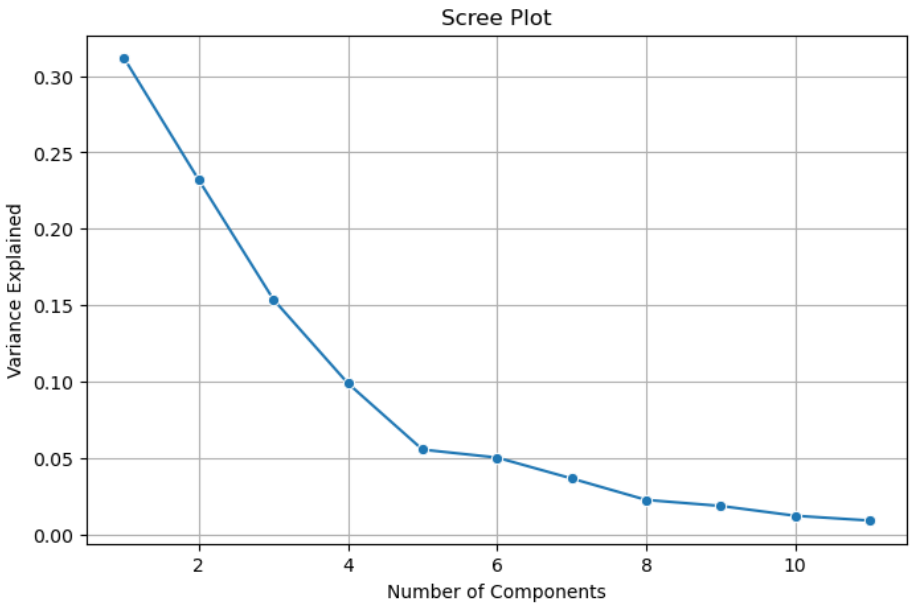
1.7 Discuss the cumulative values of the eigenvalues. How does it help you to decide on the optimum number of principal components? What do the eigenvectors indicate? Perform PCA and export the data of the Principal Component scores into a data frame.

Here, by using np.cumsum(), we have obtained the following cumulative values of eigenvalues -

array([0.31154285, 0.54344255, 0.69716768, 0.79594551, 0.8513477 , 0.90151895, 0.9380206 , 0.96047074, 0.97897558, 0.99105209, 1.])

The cumulative values of the eigenvalues in PCA represent the total amount of variance explained by each principal component and its predecessors. The cumulative values keep increasing, indicating how much variance is explained by considering an increasing number of principal components. **It helps us decide on the optimum number of principal components to retain**. By looking at the cumulative values, we can determine the proportion of total variance explained by considering the first 'k' principal components.

In Principal Component Analysis, **eigenvalues** signify the variance explained by each principal component. Larger eigenvalues correspond to principal components that capture more significant variability in the data. Selecting principal components with the highest eigenvalues allows us to retain crucial patterns, reduce data dimensionality, and preserve the most meaningful information, leading to a more concise and insightful representation of the dataset.



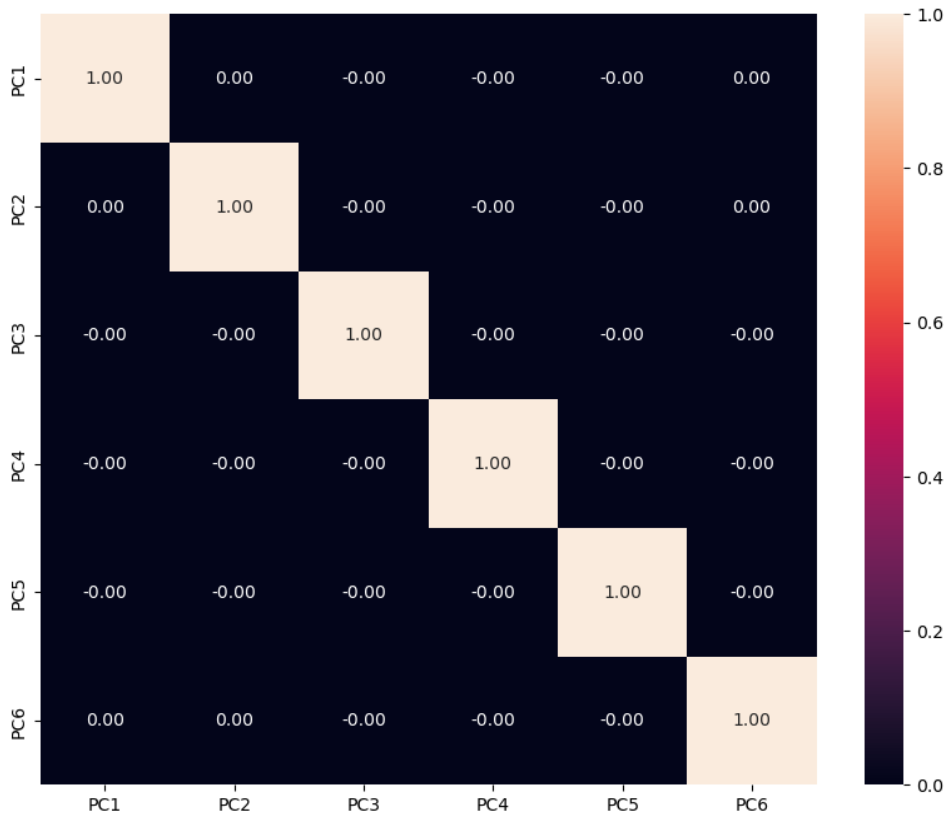
With the help of the above scree plot, we have decided to go ahead with **6 principal components.**

After which we have calculated the PC Scores and have loaded it into a new dataframe.

The following is the dataframe -

	PC1	PC2	PC3	PC4	PC5	PC6
0	0.079551	1.543198	1.895046	1.168119	-0.113909	0.086364
1	-1.100966	-2.420298	2.045521	-0.427083	-0.550453	0.475588
2	-2.197067	-0.727440	0.166800	1.310312	-1.061797	0.244819
3	1.562933	0.171366	-1.827179	-1.192240	-0.939629	-0.957207
4	0.767570	-1.428111	0.234356	0.069525	1.206498	-0.251606
5	2.908622	0.309387	1.532706	-0.746605	-0.848741	-0.334014
6	5.293191	1.057481	-0.644861	0.028470	1.303547	0.107495
7	1.476591	1.111083	0.705905	-0.567127	-1.086442	0.494166
8	-0.613948	1.379473	0.575067	-1.769264	0.367067	-0.127711
9	-0.423660	1.981541	0.336888	-0.365410	0.119662	0.310581

To cross check if we have satisfied the objective, we will plot a heatmap and check for the correlation of the variables.



The correlation is 0, which means we have achieved the goal of PCA.

1.8 Mention the business implication of using the Principal Component Analysis for this case study.

- Dimensionality Reduction: PCA reduces the number of variables in the data, simplifying computations and improving efficiency.
- Improved Insights: By focusing on the most significant principal components, businesses gain valuable insights into the key factors driving data patterns.
- Enhanced Visualization: PCA enables visualization of high-dimensional data in lower dimensions, aiding in better data interpretation.
- Data Preprocessing: PCA helps in data preprocessing, reducing noise and improving the performance of machine learning models.
- Anomaly Detection: PCA assists in detecting outliers and anomalies, facilitating fraud detection and error identification.
- Better Decision-making: PCA provides clearer understanding and interpretable results, leading to more informed business decisions.

Overall, using PCA benefits businesses with increased efficiency, improved decision-making, and a better understanding of complex datasets.

CLUSTERING

2.1 Clustering: Read the data and do exploratory data analysis. Describe the data briefly. (Check the null values, Data types, shape, EDA, etc)

- Firstly, we read the dataset and print the head to understand the data from the top 5 rows.

	Unnamed: 0	States	Health_indeces1	Health_indices2	Per_capita_income	GDP
0	0	Bachevo	417	66	564	1823
1	1	Balgarchevo	1485	646	2710	73662
2	2	Belasitsa	654	299	1104	27318
3	3	Belo_Pole	192	25	573	250
4	4	Beslen	43	8	528	22

- Secondly, we check for the data types, shape, null values using info().

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 297 entries, 0 to 296
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0         297 non-null    int64
 1   States             297 non-null    object
 2   Health_indeces1    297 non-null    int64
 3   Health_indices2    297 non-null    int64
 4   Per_capita_income  297 non-null    int64
 5   GDP                297 non-null    int64
dtypes: int64(5), object(1)
memory usage: 14.0+ KB
```

- We have created a new dataframe dropping the 2 non-useful variables Unnamed: 0 and States.

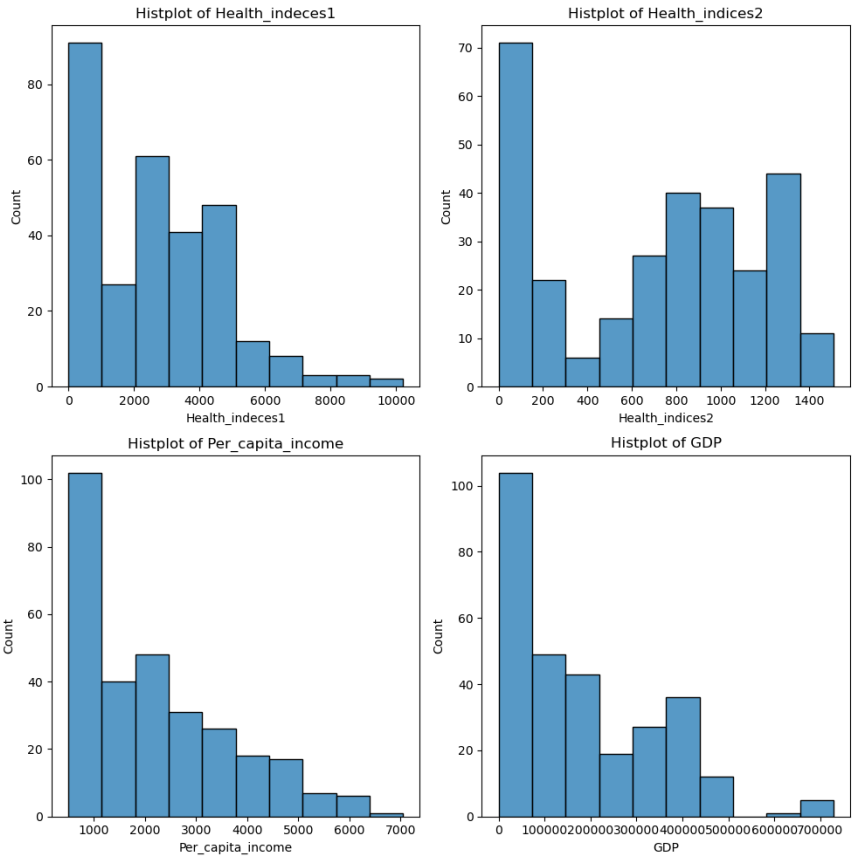
	Health_indeces1	Health_indices2	Per_capita_income	GDP
0	417	66	564	1823
1	1485	646	2710	73662
2	654	299	1104	27318
3	192	25	573	250
4	43	8	528	22

- Further, using isnull() and duplicated() we can check for null values and duplicated data. We see that there is 1 duplicated row present.
- We will drop this by using data.duplicated().

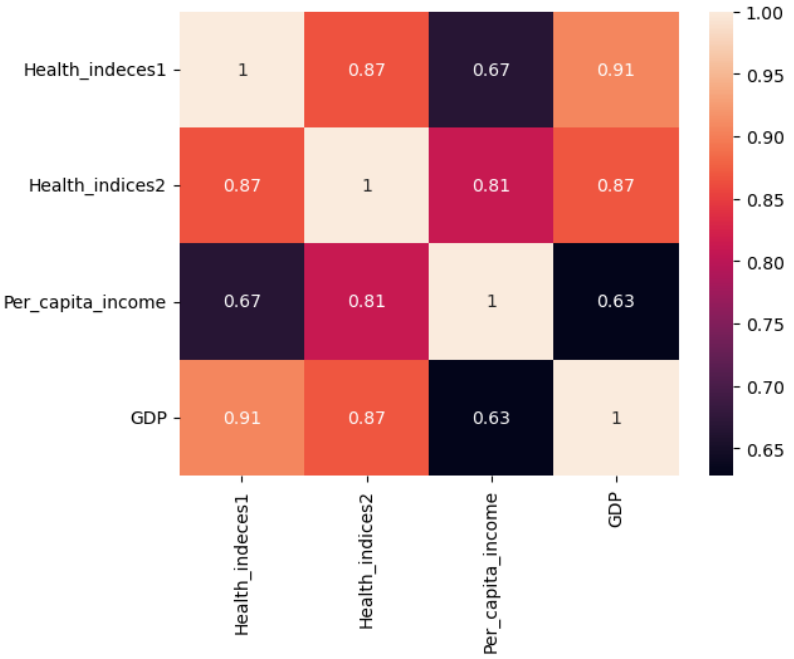
The following table is the statistical summary -

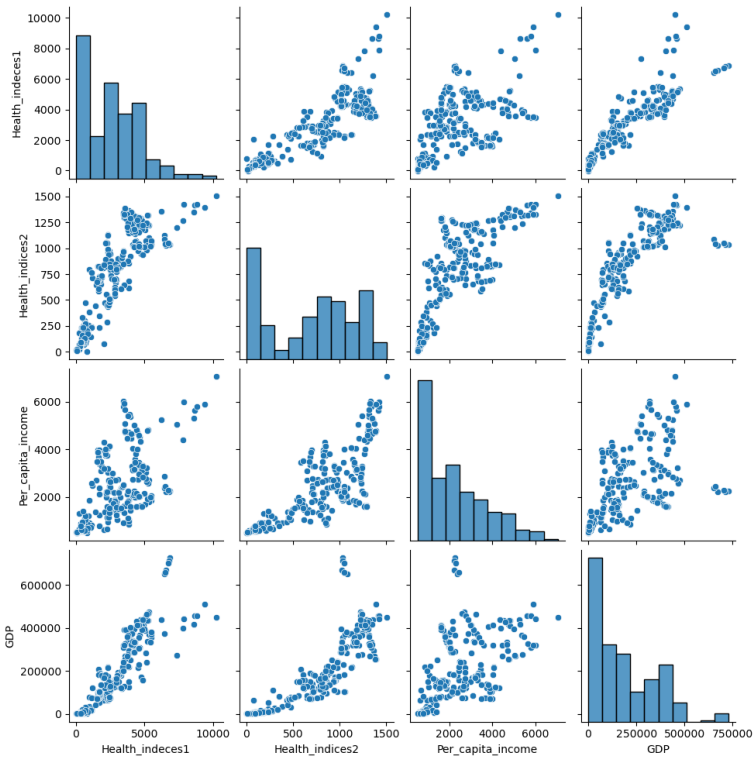
	Health_indeces1	Health_indices2	Per_capita_income	GDP
count	296.000000	296.000000	296.000000	296.000000
mean	2638.804054	695.929054	2162.422297	175190.739865
std	2036.487207	468.063249	1491.354116	167141.448576
min	-10.000000	0.000000	500.000000	22.000000
25%	650.750000	178.000000	767.500000	8748.750000
50%	2454.500000	810.500000	1869.000000	137192.500000
75%	4102.750000	1076.000000	3138.750000	314751.250000
max	10219.000000	1508.000000	7049.000000	728575.000000

Next, univariate analysis -



Finally, multivariate analysis -





2.2 Do you think scaling is necessary for clustering in this case? Justify.

In this case, scaling (zscore w.r.t this case study) is necessary for clustering. The reason is that clustering algorithms, such as K-means or hierarchical clustering, use distance-based metrics to determine similarities between data points. When features have significantly different scales, it can lead to biased cluster formation.

In the provided data, we can observe that the ranges of the features vary significantly. For example, "Per_capita_income" and "GDP" have much larger values compared to "Health_indices1" and "Health_indices2." Without scaling, the clustering algorithm may give more weight to features with larger values, which can result in suboptimal clusters.

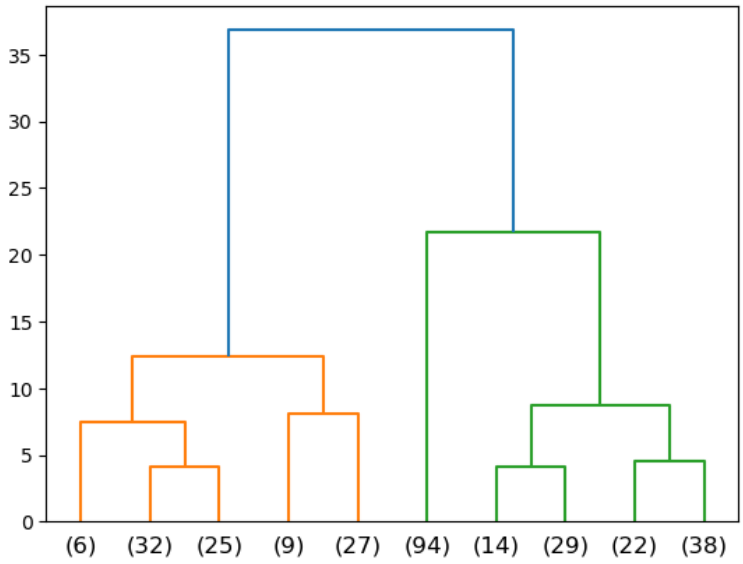
Scaling ensures that all features are brought to a similar scale, usually with a mean of zero and a standard deviation of one (e.g., using Z-score scaling). This process equalizes the importance of each feature and prevents one dominating the clustering process due to its larger scale.

In conclusion, scaling is necessary for clustering in this case to ensure fair and meaningful comparisons between the features and improve the accuracy and interpretability of the clustering results.

2.3 Apply hierarchical clustering to scaled data. Identify the number of optimum clusters using Dendrogram and briefly describe them.

- To perform Hierarchical Clustering, we will import dendrogram & linkage from scipy.cluster.hierarchy.
- We have used the wardlink method.

And after we prune the tree, the following is the dendrogram.



To identify the number of optimum clusters, we look for the longest vertical line in the dendrogram that does not intersect any horizontal lines. This line represents a significant distance jump between clusters, and cutting the dendrogram at that point suggests the optimal number of clusters. In this case, **the number of optimum clusters is 3.**

Finally, by using fcluster we have clustered the data into 3 clusters and created a new variable in the dataframe to represent which cluster a value belongs to.

	Health_indeces1	Health_indices2	Per_capita_income	GDP	Clusters
0	417	66	564	1823	2
1	1485	646	2710	73662	3
2	654	299	1104	27318	2
3	192	25	573	250	2
4	43	8	528	22	2

2.4 Apply K-Means clustering on scaled data and determine optimum clusters. Apply elbow curve and find the silhouette score.

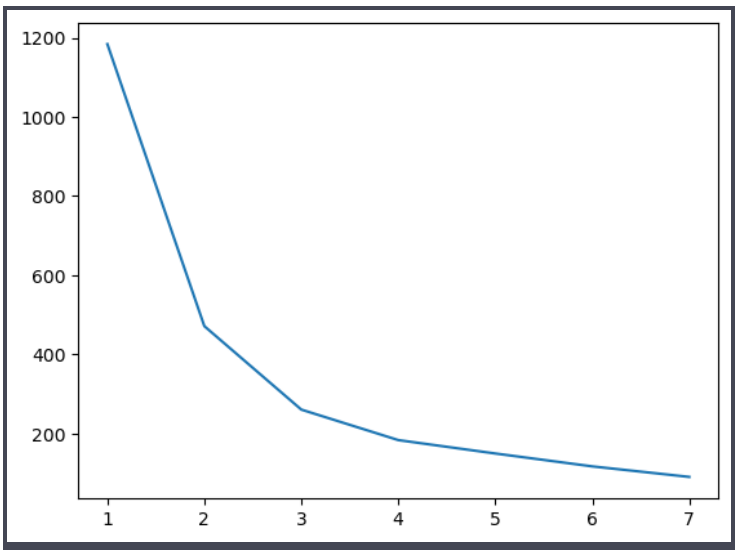
To perform K-Means clustering, we will import **KMeans from sklearn.cluster**.

Further we will fit this into our scaled data.

After which, we check the k-mean inertia where the "inertia" (also known as within-cluster sum of squares) is a measure of how compact and well-separated the clusters are. It quantifies the sum of squared distances between each data point and its assigned cluster centroid. Lower inertia indicates that the data points are tightly grouped around their respective centroids, resulting in more well-defined and distinct clusters.

The goal of the K-means algorithm is to minimize the inertia by iteratively adjusting the cluster centroids and reassigning data points until convergence.

By comparing the inertia across different values of K (number of clusters), we can use the "elbow method" to find the optimal number of clusters. The elbow method looks for a point in the inertia plot where the decrease in inertia starts to slow down, forming an elbow-like curve. This point indicates the best trade-off between reducing inertia and preventing overfitting, representing the optimal number of clusters for the given dataset.



As the above line graph (elbow method) indicates, we will consider 3 clusters as there is a significant drop in k-mean inertia till **n = 3.**

	Health_indeces1	Health_indices2	Per_capita_income	GDP	Clusters	Clus_kmeans
0	417	66	564	1823	2	2
1	1485	646	2710	73662	3	0
2	654	299	1104	27318	2	2
3	192	25	573	250	2	2
4	43	8	528	22	2	2

Here, we have created a new variable to indicate the k-means clusters.

For the Silhouette Score, we will import silhouette_score from sklearn.metrics.

This is the final Silhouette Score - 0.5322430767761168

2.5 Describe cluster profiles for the clusters defined. Recommend different priority based actions that need to be taken for different clusters on the bases of their vulnerability situations according to their Economic and Health Conditions.

To provide cluster profiles and recommend priority-based actions for different clusters based on their vulnerability situations concerning economic and health conditions, we need the actual cluster results from the clustering analysis. Without the specific clustering output and the features used for clustering, it's challenging to give precise recommendations.

However, I can provide a general approach to developing cluster profiles and suggesting actions:

Step 1: Cluster Profiling

Analyze the clustering output to understand the characteristics of each cluster. Examine the economic and health conditions of each cluster, such as per capita income, GDP, and health indices.
Identify key differences between clusters, focusing on vulnerable groups with lower economic and health indicators.

Step 2: Priority-based Actions

Based on the cluster profiles, prioritize actions as follows:

Cluster(s) with Low Economic and Health Conditions:
- Implement targeted social welfare programs to support vulnerable communities.
- Provide access to affordable healthcare services and nutrition assistance.
- Offer skill development and employment opportunities to improve economic conditions.

Cluster(s) with Low Economic but Better Health Conditions:
- Focus on programs to boost economic growth and income generation.
- Enhance access to education and job training to improve employment prospects.

Cluster(s) with Higher Economic but Poor Health Conditions:
- Strengthen healthcare facilities and services in these areas.
- Promote awareness of health-related issues and encourage preventive healthcare practices.

Cluster(s) with Higher Economic and Health Conditions:
- Focus on sustainable development and economic growth initiatives.
- Invest in infrastructure and public services to maintain the positive trajectory.

The specific actions and policies will depend on the unique characteristics of the clusters identified through the clustering analysis. It is essential to involve relevant stakeholders, including government agencies, healthcare organizations, NGOs, and community representatives, to ensure a comprehensive and effective implementation of the recommended actions.

Finally, clustering results are data-driven, and any actions taken should be evidence-based and regularly reviewed to address the changing needs of the vulnerable populations.

Thank you!