Neuronal and Evolutionary Computation

(MIA-MEISISI) Practical Exercise

A3 Unsupervised learning

Students Name: MD ARIF ULLAH BHUIYAN & JULIO CÉSAR SIGUEÑAS PACHECO

Github Link: https://github.com/arifbhuiyan82/Unsupervised-learning-with-PCA-t-SNE-k-means-AHC-and-SOM

Objective: Apply and compare the results from five different unsupervised learning techniques:

- Principal Component Analysis (PCA)
- t-distributed Neighbor Stochastic Embedding (t-SNE)
- k-means
- Agglomerative Hierarchical Clustering (AHC)
- Self-Organizing Maps (SOM)

Part 1 Selecting and Analyzing the Datasets: In this part, we utilize the "A3-Data.txt" dataset provided on Urv Moodle. Additionally, we have downloaded the "students-knowledge-level-data.txt" dataset for predicting students' knowledge levels. You can access the dataset through the following link: Predict Student's Knowledge Level Dataset

Dataset 2 Information:

Dataset Name: Students' Knowledge Level Dataset

Data Set Characteristics: Multivariate

Number of Instances: 258

Area: Education

Attribute Characteristics: Real

Number of Attributes: 5

Associated Tasks: Classification

Attribute Information: The dataset contains 5 attributes along with a target variable for classification

STG (Study Time for Goal Object Materials): The degree of study time spent on goal object materials.

SCG (Repetition Number of User for Goal Object Materials): The degree of repetition in the number of times a user revisits goal object material.

STR (Study Time for Related Objects): The degree of study time spent on related objects associated with the goal object.

LPR (Exam Performance for Related Objects): The exam performance of the user on related objects associated with the goal object.

PEG (Exam Performance for Goal Objects): The exam performance of the user on the goal objects.

Target Variable:

UNS (Knowledge Level of User): This is the target variable for classification. It represents the knowledge level of the user and has four classes:

Very Low

Low

Middle

High

The dataset consists of 258 rows, each containing the attribute values mentioned above and the corresponding knowledge level of the user (UNS).

Preprocessing: I have performed preprocessing on the 'Students' Knowledge Level Dataset' by converting the target variable 'UNS' from string labels to numeric values. Specifically, I mapped the string labels as follows:

Very Low to 3

Low to 4

Middle to 6

High to 8

This transformation allows me to work with the dataset using code and machine learning models. It simplifies the target variable while retaining its ordinal nature, making it suitable for classification tasks.

Principal Component Analysis (PCA): PCA is a dimensionality reduction technique that aims to transform high-dimensional data into a lower-dimensional representation while preserving the most important information. It achieves this by identifying and orthogonalizing the principal components, which are linear combinations of the original features that capture the variance in the data.

PCA Approach for Dataset1 (A3-data): Breakdown of PCA approach-

Data Loading: The code loads training data from a CSV file ('A3-data.txt') into a pandas DataFrame.

Feature and Target Definitions: It defines the features (x) and the target variable ('class').

Data Preparation: The features are separated, and standardization is applied to ensure consistent scaling.

PCA: Principal Component Analysis (PCA) is performed to reduce the dimensionality of the data to two principal components.

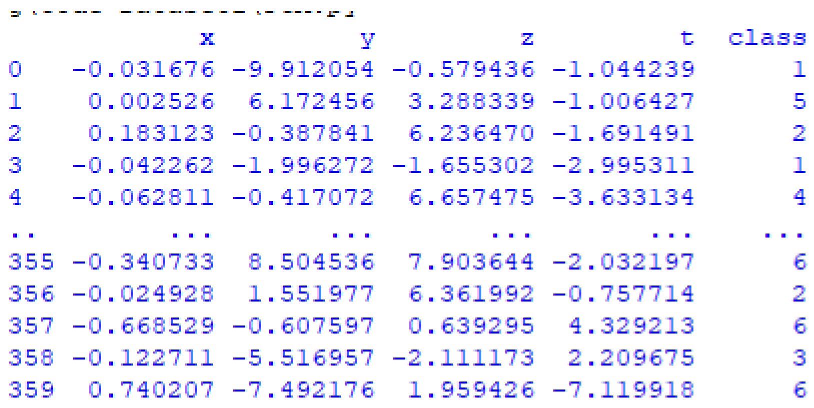
Data Transformation: A DataFrame ('principal_df') is created to store the two principal components.

Combining Principal Components with Class Labels: The principal components are combined with class labels to create a new DataFrame ('final_df').

Scatter Plot Visualization: A 2D scatter plot is generated to visualize the PCA results, with each class represented by a different color.

Displaying the Plot: The scatter plot is displayed, providing a visual representation of how the data points are distributed in the reduced 2D space.

Result from Dataset1 (A3-data):

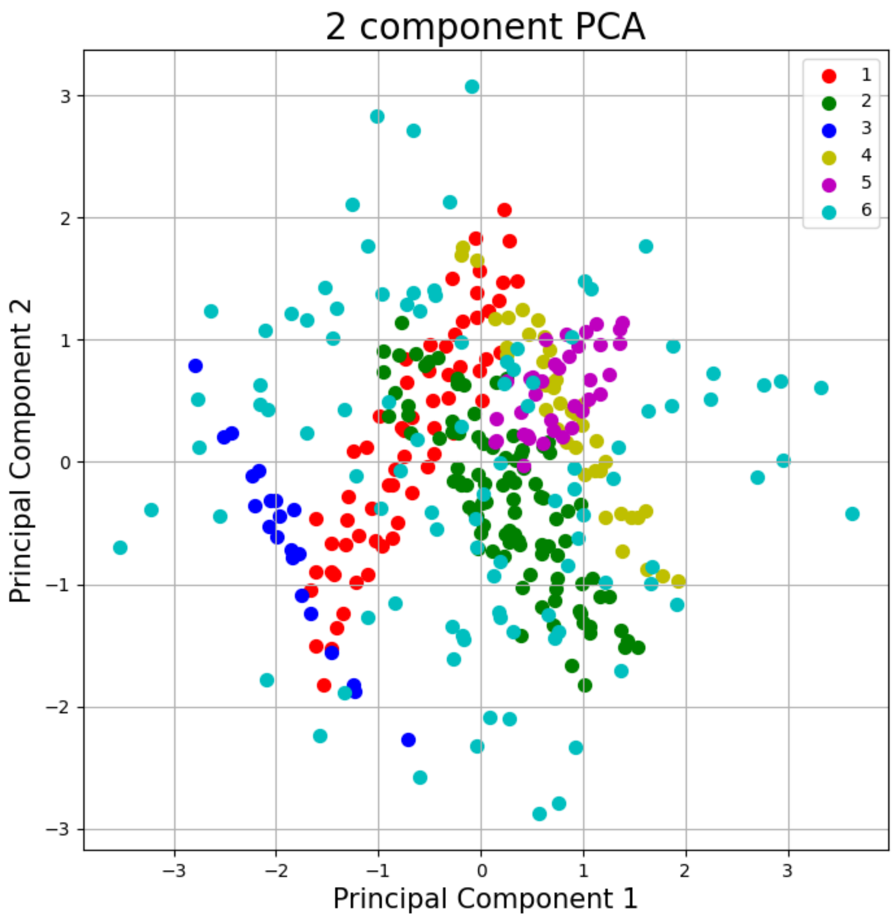


```
            x         y         z         t  class
0   -0.031676 -9.912054 -0.579436 -1.044239      1
1    0.002526  6.172456  3.288339 -1.006427      5
2    0.183123 -0.387841  6.236470 -1.691491      2
3   -0.042262 -1.996272 -1.655302 -2.995311      1
4   -0.062811 -0.417072  6.657475 -3.633134      4
..        ...       ...       ...       ...    ...
355 -0.340733  8.504536  7.903644 -2.032197      6
356 -0.024928  1.551977  6.361992 -0.757714      2
357 -0.668529 -0.607597  0.639295  4.329213      6
358 -0.122711 -5.516957 -2.111173  2.209675      3
359  0.740207 -7.492176  1.959426 -7.119918      6
```

```
[360 rows x 5 columns]
original shape:    (360, 4)
transformed shape: (360, 2)
     principal component 1  principal component 2
0                -1.615413              -0.461632
1                 0.499510               0.700627
2                 0.603876              -0.296531
3                -0.726266               0.649637
4                 0.633048               0.427483
..                     ...                    ...
355               1.071133               1.419143
356               0.404593               0.104707
357              -2.159519               0.628094
358              -2.000091              -0.313355
359               0.847054              -0.846901

[360 rows x 2 columns]
     principal component 1  principal component 2  class
0                -1.615413              -0.461632      1
1                 0.499510               0.700627      5
2                 0.603876              -0.296531      2
3                -0.726266               0.649637      1
4                 0.633048               0.427483      4
..                     ...                    ...    ...
355               1.071133               1.419143      6
356               0.404593               0.104707      2
357              -2.159519               0.628094      6
358              -2.000091              -0.313355      3
359               0.847054              -0.846901      6

[360 rows x 3 columns]
```



PCA Approach for Dataset1 (students- knowledge-level-data): Breakdown of PCA approach-

Data Loading: The code loads training data from a CSV file ('students- knowledge-level-data-numeric.txt') into a pandas DataFrame.

Feature and Target Definitions: It defines the features ('STG', 'SCG', 'STR', 'LPR', 'PEG') and the target variable ('UNS').

Data Preparation: The features are separated, and standardization is applied to ensure consistent scaling using StandardScaler().

PCA: Principal Component Analysis (PCA) is performed to reduce the dimensionality of the data to two principal components.

Data Transformation: A DataFrame ('principal_df') is created to store the two principal components.

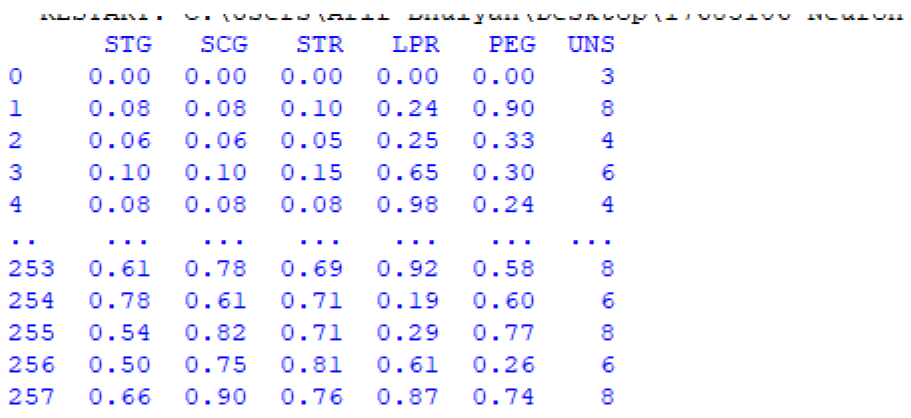
Combining Principal Components with Target Variable: The principal components are combined with the target variable ('UNS') to create a new DataFrame ('final_df').

Explained Variance and Ratios: The code prints out the explained variance and explained variance ratios of the principal components, providing insight into the amount of variance captured by each component.

Scatter Plot Visualization: A 2D scatter plot is generated to visualize the PCA results, with each class represented by a different color. Class labels are obtained from the 'UNS' column.

Displaying the Plot: The scatter plot is displayed, allowing for a visual representation of how the data points are distributed in the reduced 2D space, based on their 'UNS' class labels.

Result from Dataset2 (students- knowledge-level-data):

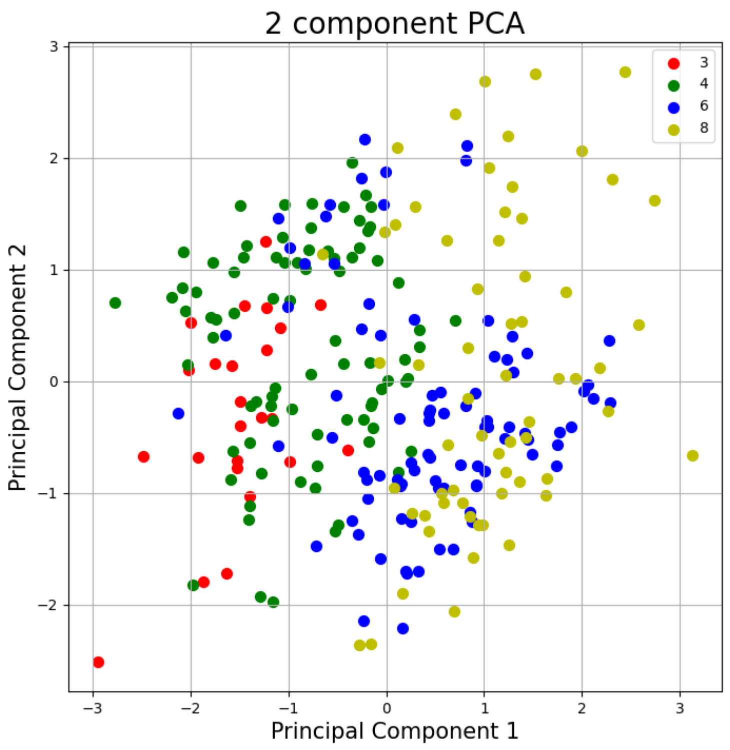


```
      STG   SCG   STR   LPR   PEG  UNS
0    0.00  0.00  0.00  0.00  0.00    3
1    0.08  0.08  0.10  0.24  0.90    8
2    0.06  0.06  0.05  0.25  0.33    4
3    0.10  0.10  0.15  0.65  0.30    6
4    0.08  0.08  0.08  0.98  0.24    4
..    ...   ...   ...   ...   ...  ...
253  0.61  0.78  0.69  0.92  0.58    8
254  0.78  0.61  0.71  0.19  0.60    6
255  0.54  0.82  0.71  0.29  0.77    8
256  0.50  0.75  0.81  0.61  0.26    6
257  0.66  0.90  0.76  0.87  0.74    8
```

```
[258 rows x 6 columns]
original shape:    (258, 5)
transformed shape: (258, 2)
     principal component 1  principal component 2
0                -2.941025              -2.511693
1                -0.272425              -2.357347
2                -1.970034              -1.821678
3                -2.124758              -0.285898
4                -2.767039               0.709394
..                     ...                    ...
253               1.517685               2.752831
254               2.279413               0.367221
255               2.577824               0.508082
256               0.814960               1.984051
257               2.433872               2.772799

[258 rows x 2 columns]
     principal component 1  principal component 2  UNS
0                -2.941025              -2.511693    3
1                -0.272425              -2.357347    8
2                -1.970034              -1.821678    4
3                -2.124758              -0.285898    6
4                -2.767039               0.709394    4
..                     ...                    ...  ...
253               1.517685               2.752831    8
254               2.279413               0.367221    6
255               2.577824               0.508082    8
256               0.814960               1.984051    6
257               2.433872               2.772799    8

[258 rows x 3 columns]
[1.39464629 1.19480422]
[0.27784814 0.23803464]
[3, 4, 6, 8]
```



t-distributed Neighbor Stochastic Embedding (t-SNE): t-SNE is a dimensionality reduction technique used for visualizing high-dimensional data in lower dimensions, typically 2D or 3D. It focuses on preserving pairwise similarities between data points, making it effective for clustering and visualization tasks, especially when dealing with complex, non-linear relationships in the data.

t-SNE Approach for Dataset1 (A3-data): Breakdown of t-SNE approach-

Data Loading: The code loads training data from a CSV file ('A3-data.txt') into a pandas DataFrame.

Feature and Target Definitions: It defines the features ('x', 'y', 'z', 't') and the target variable ('class').

Data Preparation: The features are separated, and standardization is applied using StandardScaler() to ensure consistent scaling.

t-SNE Dimensionality Reduction: t-SNE (t-Distributed Stochastic Neighbor Embedding) is applied for dimensionality reduction to two dimensions (n_components=2). This technique is useful for visualizing high-dimensional data in a lower-dimensional space.

Performance Measurement: The code records the start time before applying t-SNE to measure the time elapsed during the dimensionality reduction process.

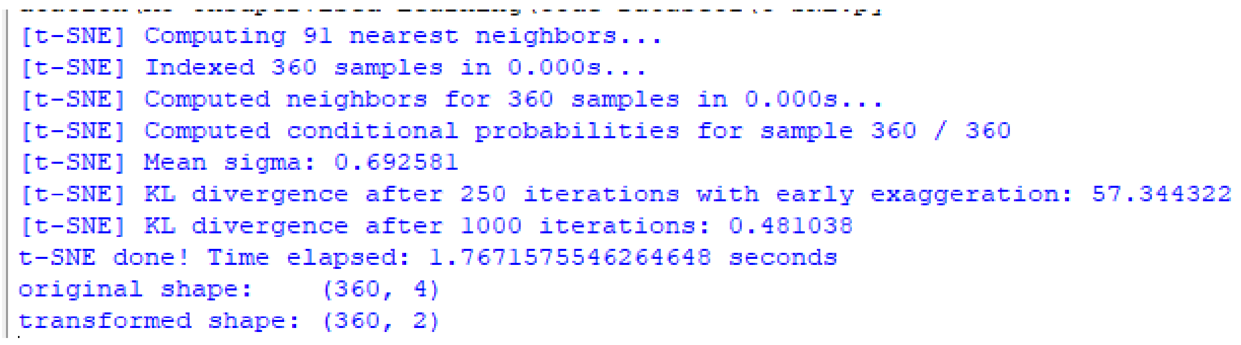
t-SNE Completion Message: After t-SNE is completed, a message is printed along with the elapsed time.

Shape Information: The code prints the shape of the original and transformed data, providing insight into the dimensionality change.

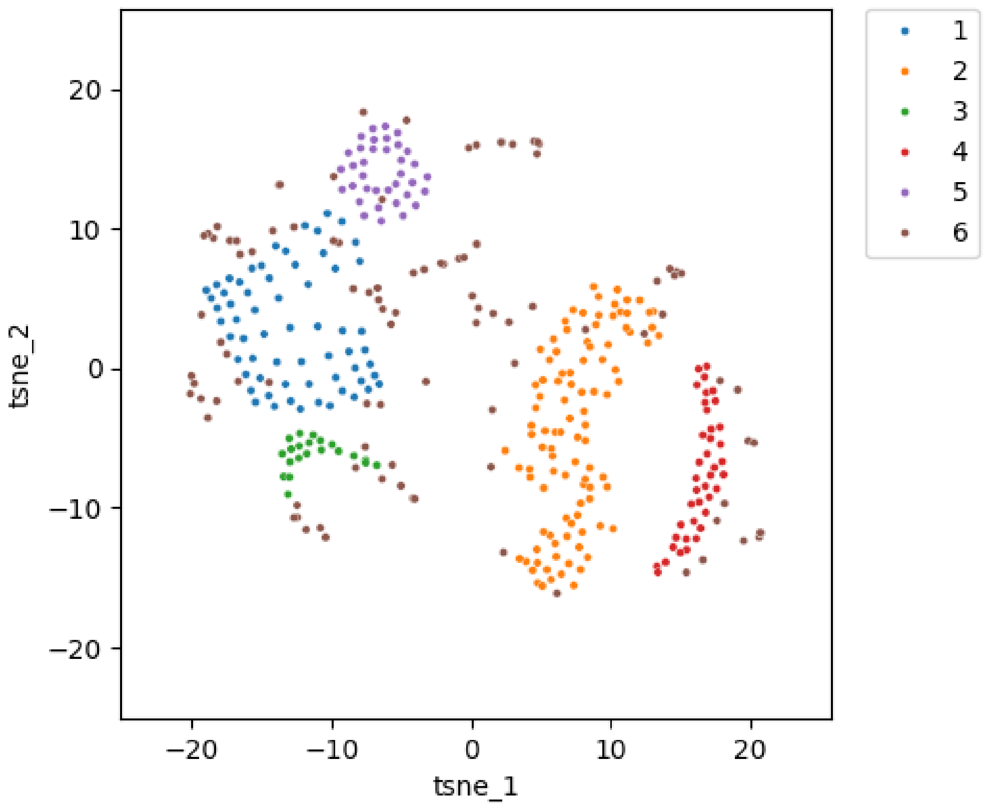
Visualization: The t-SNE results are plotted in a 2D scatter plot with color-coded labels based on the 'class' variable. The seaborn library is used for the scatter plot. Each class is represented by a different color.

Plot Customization: The code customizes the plot, ensuring equal aspect ratio, setting the legend, and adjusting the axis limits.

Result from Dataset1 (A3-data):



```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 360 samples in 0.000s...
[t-SNE] Computed neighbors for 360 samples in 0.000s...
[t-SNE] Computed conditional probabilities for sample 360 / 360
[t-SNE] Mean sigma: 0.692581
[t-SNE] KL divergence after 250 iterations with early exaggeration: 57.344322
[t-SNE] KL divergence after 1000 iterations: 0.481038
t-SNE done! Time elapsed: 1.7671575546264648 seconds
original shape:    (360, 4)
transformed shape: (360, 2)
```



t-SNE Approach for Dataset1 (students- knowledge-level-data): Breakdown of t-SNE approach-

Data Loading: The code loads training data from a CSV file ('students- knowledge-level-data-numeric.txt') into a pandas DataFrame.

Feature and Target Definitions: It defines the features ('STG', 'SCG', 'STR', 'LPR', 'PEG') and the target variable ('UNS').

Data Preparation: The features are separated, and standardization is applied using StandardScaler() to ensure consistent scaling.

t-SNE Dimensionality Reduction: t-SNE (t-Distributed Stochastic Neighbor Embedding) is applied for dimensionality reduction to two dimensions (n_components=2). This technique is used for visualizing high-dimensional data in a lower-dimensional space.

Performance Measurement: The code records the start time before applying t-SNE to measure the time elapsed during the dimensionality reduction process.

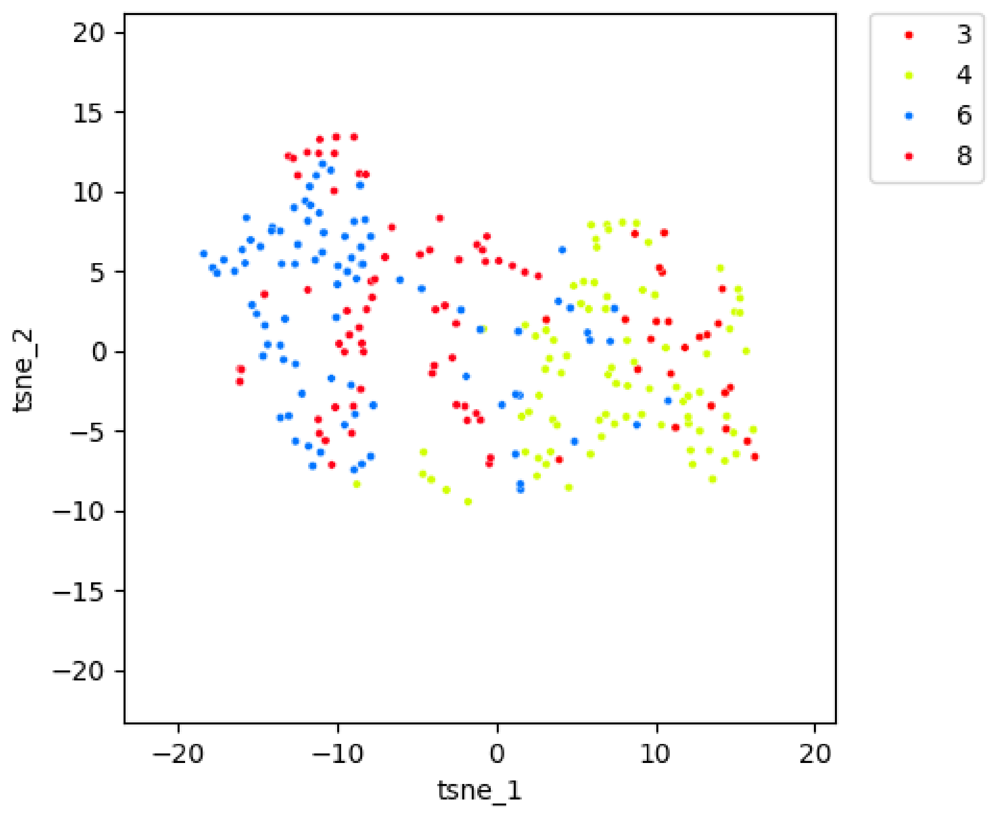
t-SNE Completion Message: After t-SNE is completed, a message is printed along with the elapsed time.

Shape Information: The code prints the shape of the original and transformed data, providing insight into the dimensionality change.

Visualization: The t-SNE results are plotted in a 2D scatter plot with color-coded labels based on the 'class' variable. The seaborn library is used for the scatter plot. Each class is represented by a different color.

Plot Customization: The code customizes the plot, ensuring equal aspect ratio, setting the legend, and adjusting the axis limits.

Result from Dataset2 (students- knowledge-level-data):



K-means: K-means is a popular clustering algorithm that partitions a dataset into 'k' clusters based on similarity. It aims to minimize the within-cluster variance by iteratively assigning data points to the nearest cluster center and updating the cluster centers based on the mean of the assigned points. K-means is sensitive to the initial choice of cluster centers.

K-means Approach for Dataset1 (A3-data): Breakdown of K-Means approach-

Data Loading: The code loads training data from a CSV file ('A3-data.txt') into a pandas DataFrame.

Feature and Target Definitions: It defines the features ('x', 'y', 'z', 't') and the target variable ('class').

Separating Features: The features are separated into the variable 'X', and summary statistics of the features are printed.

Data Visualization (Scatter Matrix): The code creates a scatter matrix plot using Plotly Express to visualize the relationships between feature variables. Each point in the matrix represents a pair of features, and points are color-coded by the 'class' variable.

Standardization: The features are standardized (scaled) using StandardScaler() to ensure consistent scaling.

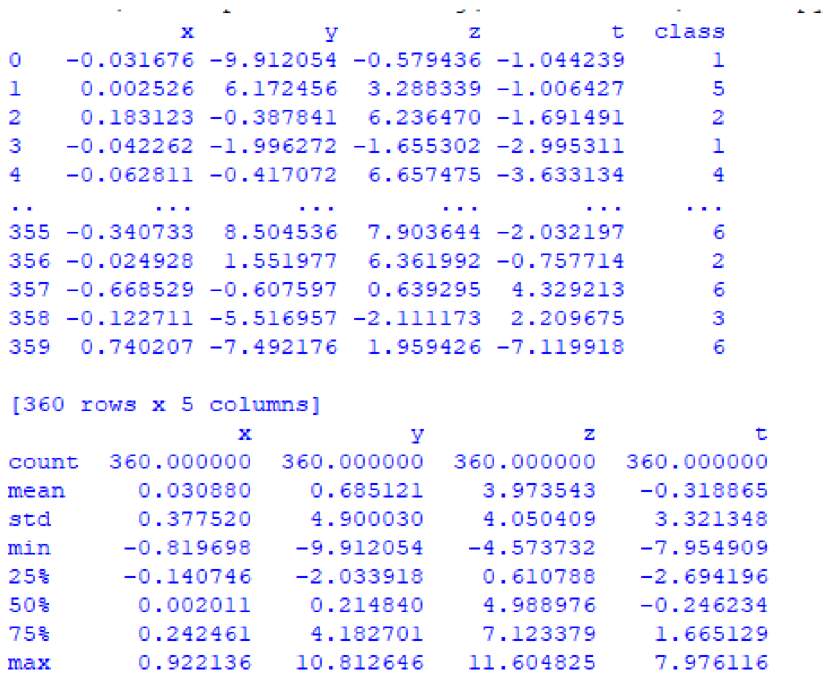
Elbow Method for K-Means: The code applies the elbow method to determine the optimal number of clusters (k) for K-means clustering. It iterates through different values of k and plots an elbow curve to help select the appropriate number of clusters.

K-Means Clustering: K-means clustering is applied with the determined number of clusters. The cluster labels are added to the DataFrame as 'KMeans_clusters'.

PCA Dimensionality Reduction (2D): Principal Component Analysis (PCA) is applied to reduce the dimensionality of the data to two components ('pca1' and 'pca2'). Ground-truth class labels and K-means cluster labels are visualized in 2D using scatter plots.

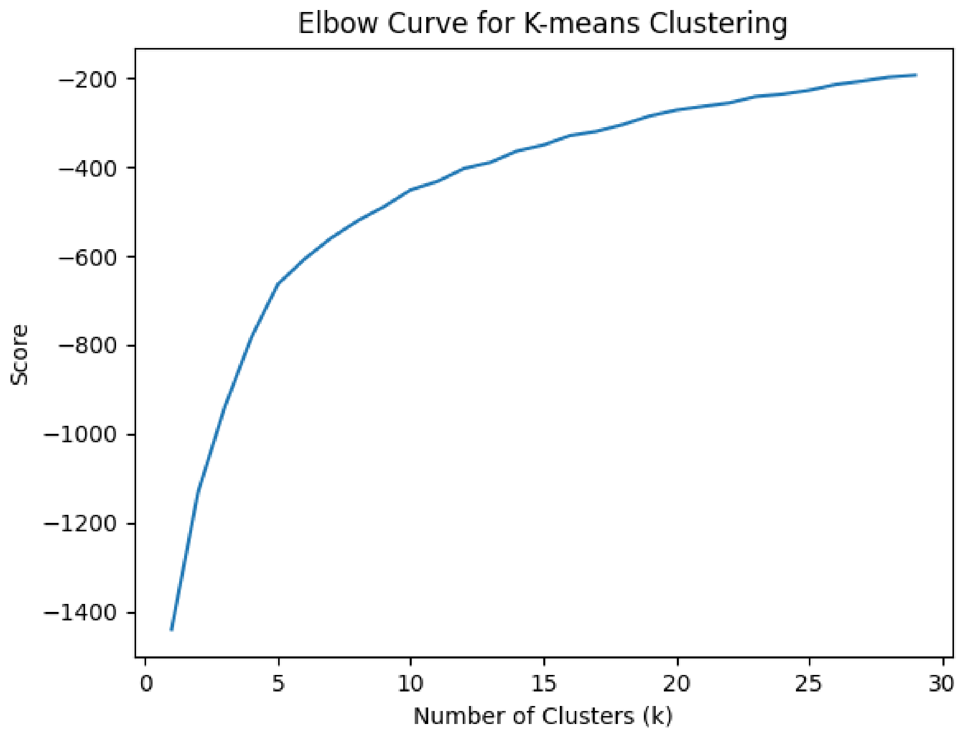
PCA Dimensionality Reduction (3D): PCA is applied again to reduce the dimensionality of the data to three components ('pca1', 'pca2', and 'pca3'). A 3D scatter plot is created to visualize the data points in a three-dimensional space.

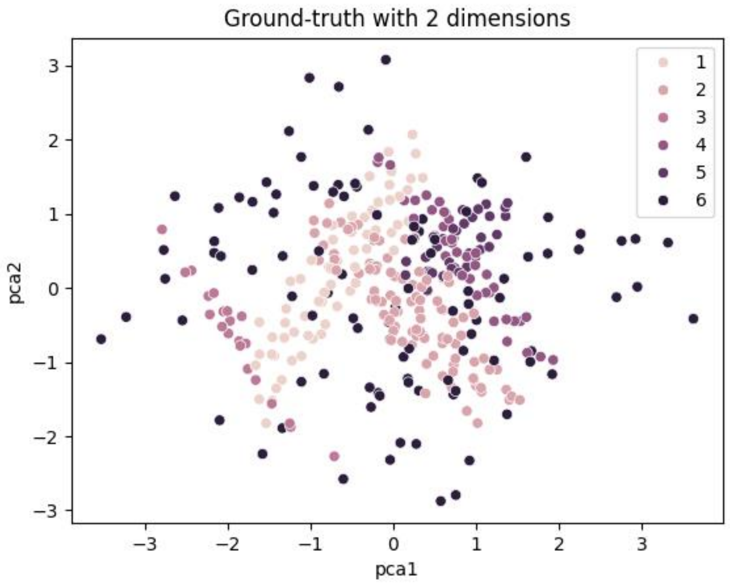
Result from Dataset1 (A3-data):

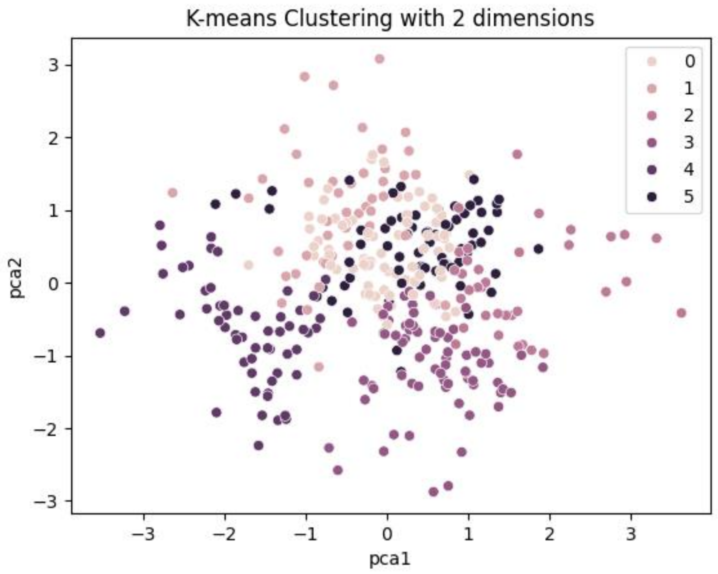


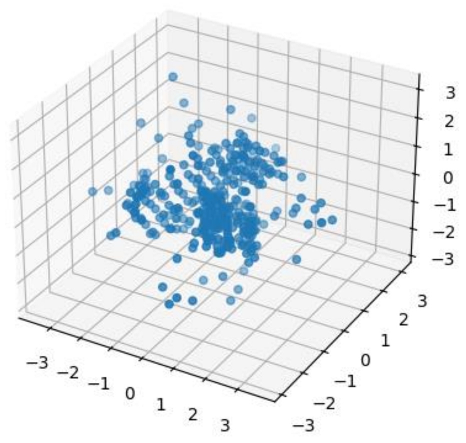
```
             x         y         z         t  class
0    -0.031676 -9.912054 -0.579436 -1.044239      1
1     0.002526  6.172456  3.288339 -1.006427      5
2     0.183123 -0.387841  6.236470 -1.691491      2
3    -0.042262 -1.996272 -1.655302 -2.995311      1
4    -0.062811 -0.417072  6.657475 -3.633134      4
..         ...       ...       ...       ...    ...
355 -0.340733  8.504536  7.903644 -2.032197      6
356 -0.024928  1.551977  6.361992 -0.757714      2
357 -0.668529 -0.607597  0.639295  4.329213      6
358 -0.122711 -5.516957 -2.111173  2.209675      3
359  0.740207 -7.492176  1.959426 -7.119918      6

[360 rows x 5 columns]
                x           y           z           t
count  360.000000  360.000000  360.000000  360.000000
mean     0.030880    0.685121    3.973543   -0.318865
std      0.377520    4.900030    4.050409    3.321348
min     -0.819698   -9.912054   -4.573732   -7.954909
25%     -0.140746   -2.033918    0.610788   -2.694196
50%      0.002011    0.214840    4.988976   -0.246234
75%      0.242461    4.182701    7.123379    1.665129
max      0.922136   10.812646   11.604825    7.976116
```




Ground-truth with 2 dimensions
pca1
pca2
1
2
3
4
5
6


K-means Clustering with 2 dimensions
pca1
pca2
0
1
2
3
4
5



K-means Approach for Dataset2 (students- knowledge-level-data): Breakdown of K-means approach-

Data Loading: The code loads training data from a CSV file ('students- knowledge-level-data-numeric.txt') into a pandas DataFrame.

Feature and Target Definitions: It defines the features ('STG', 'SCG', 'STR', 'LPR', 'PEG') and the target variable ('UNS').

Separating Features: The features are separated into the variable 'X', and summary statistics of the features are printed.

Data Visualization (Scatter Matrix): The code creates a scatter matrix plot using Plotly Express to visualize the relationships between feature variables. Each point in the matrix represents a pair of features, and points are color-coded by the 'UNS' variable.

Standardization: The features are standardized (scaled) using StandardScaler() to ensure consistent scaling.

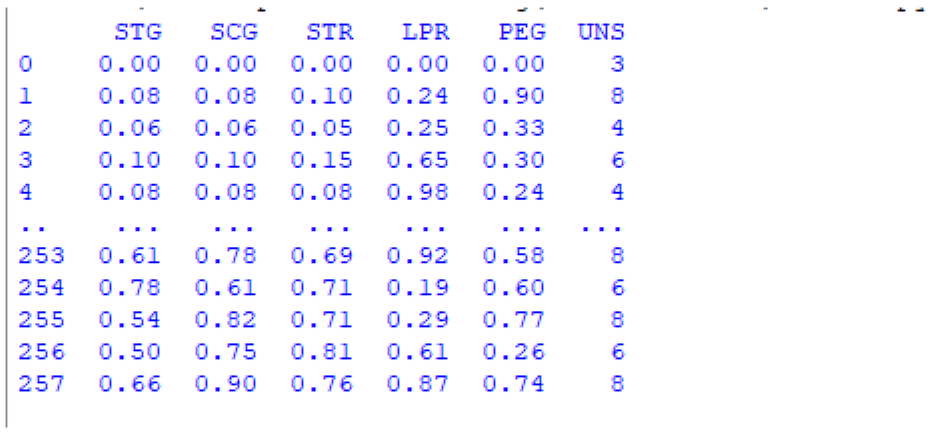
Elbow Method for K-Means: The code applies the elbow method to determine the optimal number of clusters (k) for K-means clustering. It iterates through different values of k and plots an elbow curve to help select the appropriate number of clusters.

K-Means Clustering: K-means clustering is applied with the determined number of clusters. The cluster labels are added to the DataFrame as 'KMeans_clusters'.

PCA Dimensionality Reduction (2D): Principal Component Analysis (PCA) is applied to reduce the dimensionality of the data to two components ('pca1' and 'pca2'). Ground-truth class labels and K-means cluster labels are visualized in 2D using scatter plots.

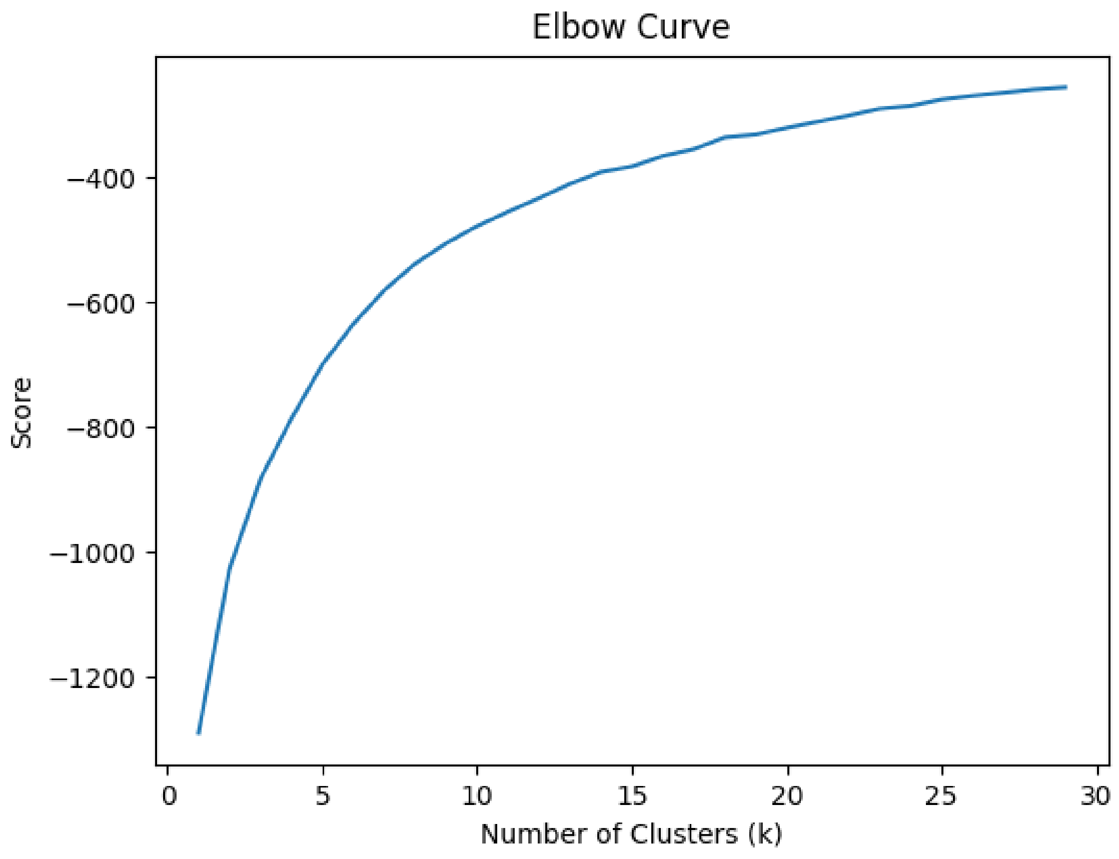
Classification Report: A classification report is printed, providing information on precision, recall, F1-score, and support for each class.

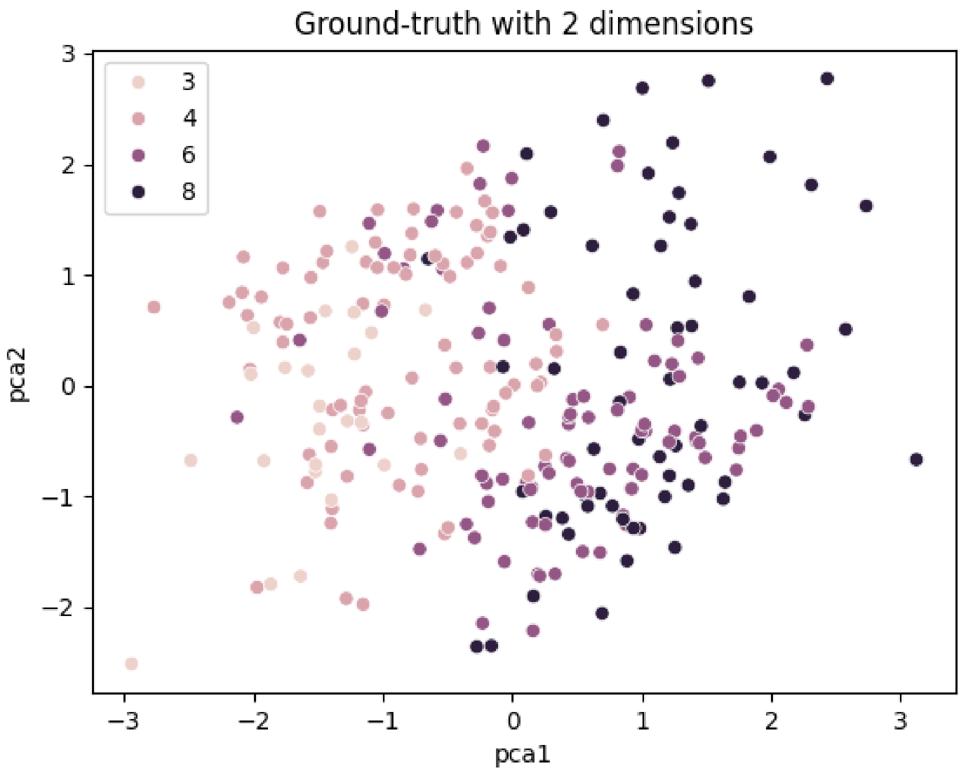
Result from Dataset2 (students- knowledge-level-data):

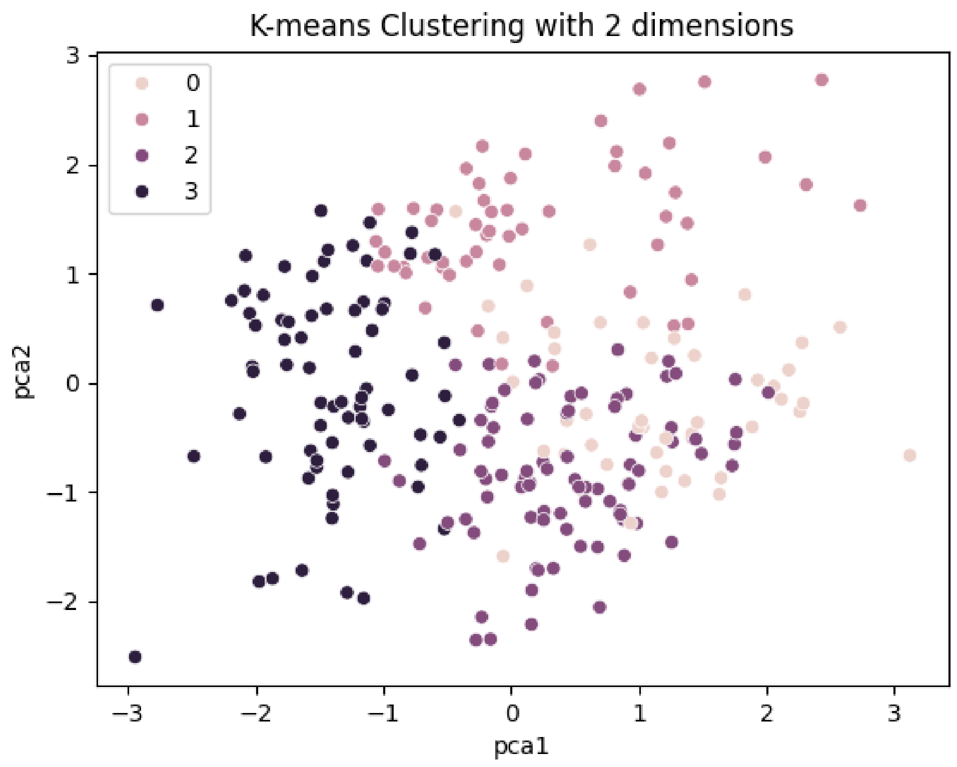


```
      STG   SCG   STR   LPR   PEG  UNS
0    0.00  0.00  0.00  0.00  0.00    3
1    0.08  0.08  0.10  0.24  0.90    8
2    0.06  0.06  0.05  0.25  0.33    4
3    0.10  0.10  0.15  0.65  0.30    6
4    0.08  0.08  0.08  0.98  0.24    4
..    ...   ...   ...   ...   ...  ...
253  0.61  0.78  0.69  0.92  0.58    8
254  0.78  0.61  0.71  0.19  0.60    6
255  0.54  0.82  0.71  0.29  0.77    8
256  0.50  0.75  0.81  0.61  0.26    6
257  0.66  0.90  0.76  0.87  0.74    8
```

```
[258 rows x 6 columns]
              STG         SCG         STR         LPR         PEG
count  258.000000  258.000000  258.000000  258.000000  258.000000
mean     0.371147    0.355674    0.468004    0.432713    0.458539
std      0.210271    0.211962    0.245940    0.248108    0.255211
min      0.000000    0.000000    0.000000    0.000000    0.000000
25%      0.240750    0.210000    0.291250    0.250000    0.250000
50%      0.327000    0.302500    0.490000    0.330000    0.500000
75%      0.495000    0.497500    0.690000    0.647500    0.660000
max      0.990000    0.900000    0.950000    0.990000    0.930000
[[-1.76851922 -1.68126919 -1.90661929 -1.74744134 -1.80019743]
 [-1.38731871 -1.30311004 -1.49922538 -0.77824063  1.73315205]
 [-1.48261884 -1.39764983 -1.70292233 -0.73785726 -0.50463595]
 ...
 [ 0.80458427  2.19486207  0.98587747 -0.57632381  1.22277934]
 [ 0.61398401  1.86397282  1.39327138  0.71594381 -0.77945202]
 [ 1.37638505  2.57302122  1.18957442  1.76591125  1.10500103]]
```




Ground-truth with 2 dimensions
3
4
6
8
pca1
pca2


K-means Clustering with 2 dimensions
0
1
2
3
pca1
pca2

Agglomerative Hierarchical Clustering (AHC): AHC is a hierarchical clustering technique that starts with each data point as its own cluster and then merges clusters iteratively based on a chosen linkage criterion (e.g., single, complete, average linkage). It produces a hierarchical tree-like structure called a dendrogram, which can be cut at different levels to obtain different numbers of clusters.

AHC Approach for Dataset2 (A3-data): Breakdown of AHC approach-

Data Loading: The code loads training data from a CSV file ('A3-data.txt') into a pandas DataFrame.

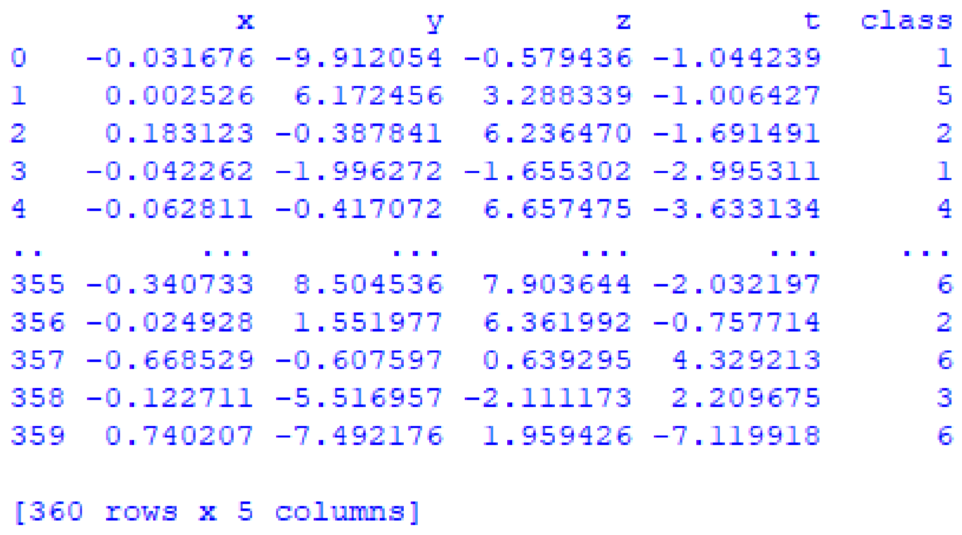
Feature and Target Definitions: It defines the features ('x', 'y', 'z', 't') and the target variable ('class').

Separating Features: The features are separated into the variable 'X'.

Dendrogram Plot Function: The code defines a function plot_dendrogram to create a dendrogram for hierarchical clustering.

Hierarchical Clustering (Average Linkage): It performs hierarchical clustering using the Agglomerative Clustering algorithm with 'average' linkage. The dendrogram is plotted to visualize the hierarchy of clusters. It shows the top three levels of the dendrogram.

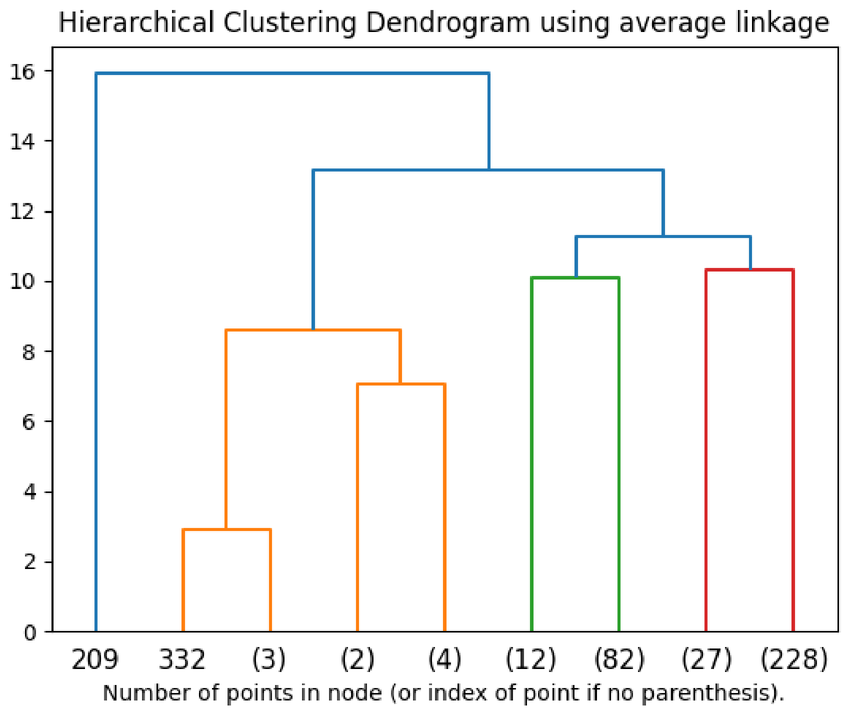
Hierarchical Clustering (Complete Linkage): It performs hierarchical clustering again using the Agglomerative Clustering algorithm with 'complete' linkage. Similar to the previous step, the dendrogram is plotted to visualize the hierarchy of clusters, displaying the top three levels.

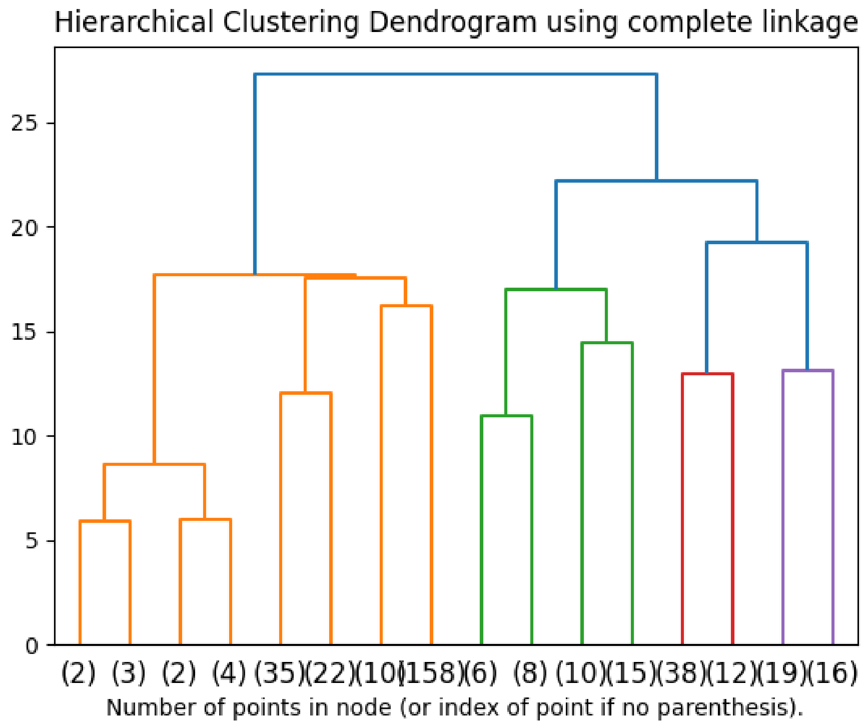


```
            x         y         z         t  class
0   -0.031676 -9.912054 -0.579436 -1.044239      1
1    0.002526  6.172456  3.288339 -1.006427      5
2    0.183123 -0.387841  6.236470 -1.691491      2
3   -0.042262 -1.996272 -1.655302 -2.995311      1
4   -0.062811 -0.417072  6.657475 -3.633134      4
..        ...       ...       ...       ...    ...
355 -0.340733  8.504536  7.903644 -2.032197      6
356 -0.024928  1.551977  6.361992 -0.757714      2
357 -0.668529 -0.607597  0.639295  4.329213      6
358 -0.122711 -5.516957 -2.111173  2.209675      3
359  0.740207 -7.492176  1.959426 -7.119918      6

[360 rows x 5 columns]
```

Result from Dataset1 (A3-data):





AHC Approach for Dataset2 (students- knowledge-level-data): Breakdown of AHC approach-

Data Loading: The code loads training data from a CSV file ('students- knowledge-level-data-numeric.txt') into a pandas DataFrame.

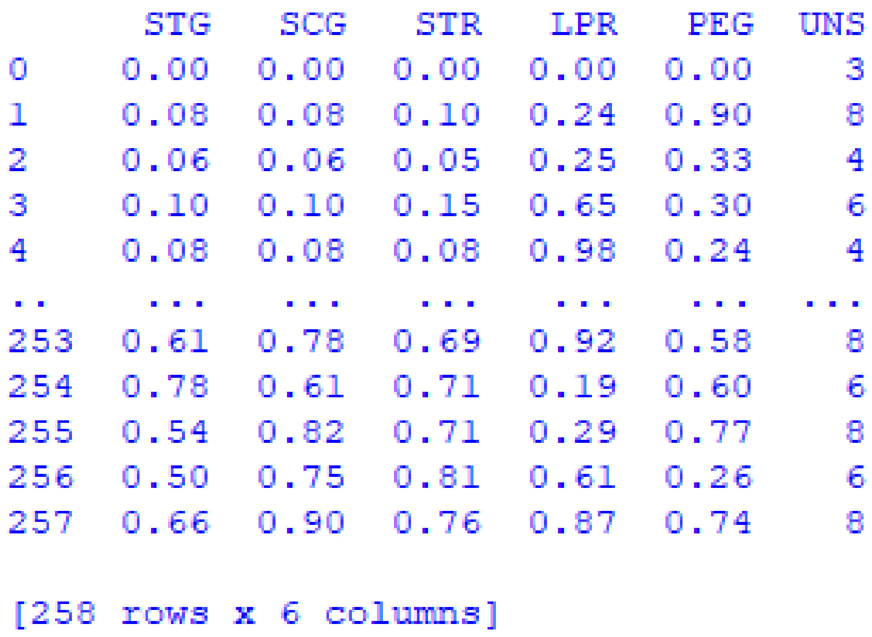
Feature and Target Definitions: It defines the features ('STG', 'SCG', 'STR', 'LPR', 'PEG') and the target variable ('UNS').

Separating Features: The features are separated into the variable 'X'.

Dendrogram Plot Function: The code defines a function plot_dendrogram to create a dendrogram for hierarchical clustering.

Hierarchical Clustering (Average Linkage): It performs hierarchical clustering using the Agglomerative Clustering algorithm with 'average' linkage. The dendrogram is plotted to visualize the hierarchy of clusters. It shows the top three levels of the dendrogram.

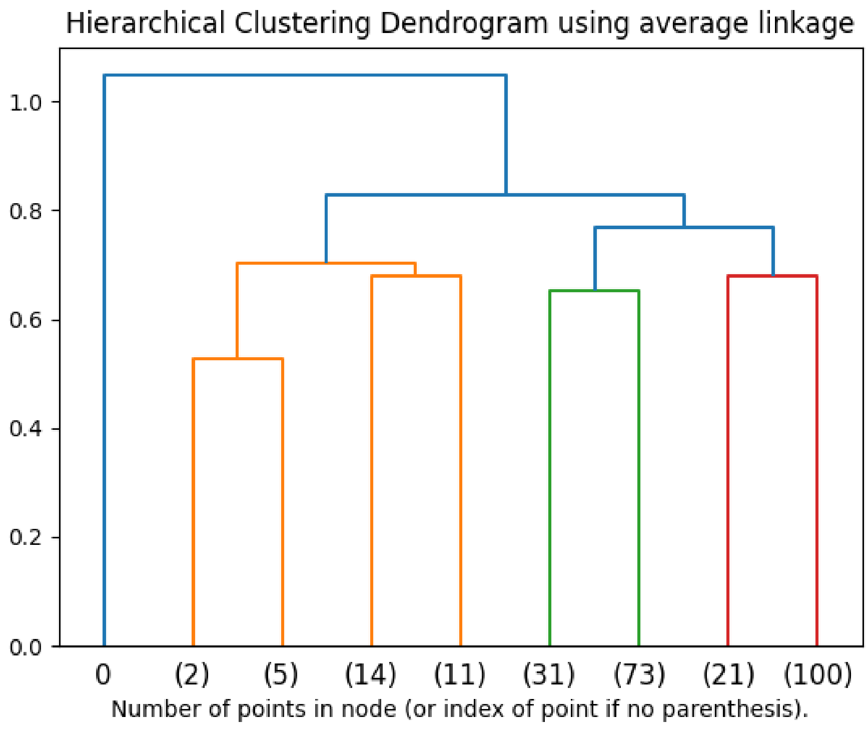
Hierarchical Clustering (Complete Linkage): It performs hierarchical clustering again using the Agglomerative Clustering algorithm with 'complete' linkage. Similar to the previous step, the dendrogram is plotted to visualize the hierarchy of clusters, displaying the top three levels.

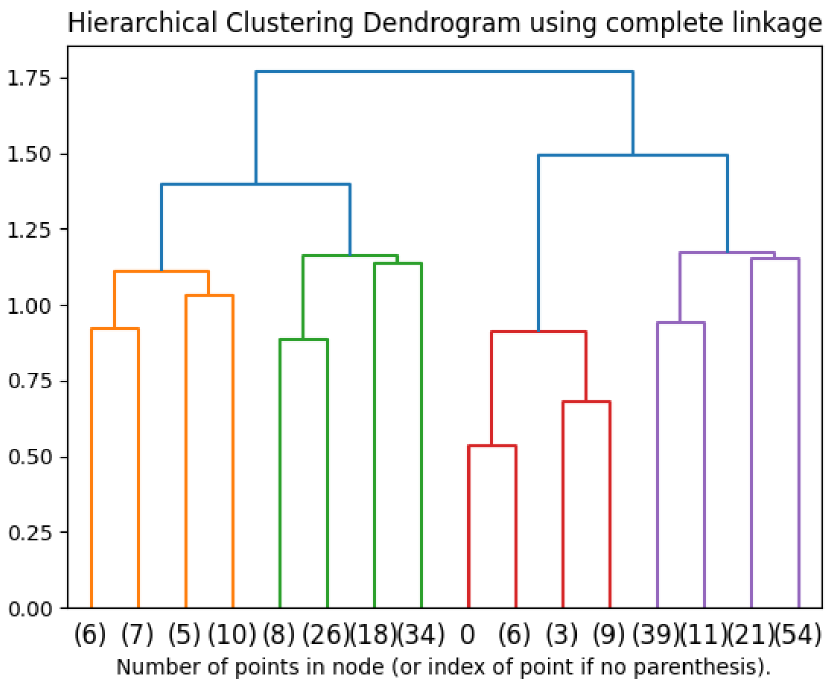


```
      STG   SCG   STR   LPR   PEG  UNS
0    0.00  0.00  0.00  0.00  0.00    3
1    0.08  0.08  0.10  0.24  0.90    8
2    0.06  0.06  0.05  0.25  0.33    4
3    0.10  0.10  0.15  0.65  0.30    6
4    0.08  0.08  0.08  0.98  0.24    4
..    ...   ...   ...   ...   ...  ...
253  0.61  0.78  0.69  0.92  0.58    8
254  0.78  0.61  0.71  0.19  0.60    6
255  0.54  0.82  0.71  0.29  0.77    8
256  0.50  0.75  0.81  0.61  0.26    6
257  0.66  0.90  0.76  0.87  0.74    8

[258 rows x 6 columns]
```

Result from Dataset2 (students- knowledge-level-data):





Self-Organizing Maps (SOM): SOM is a type of artificial neural network used for dimensionality reduction and visualization. It maps high-dimensional data onto a lower-dimensional grid or lattice, preserving the topological relationships between data points. SOMs are often used for tasks like clustering and visualization, particularly in cases where the data exhibits complex structures.

SOM Approach for Dataset1 (A3-data): Breakdown of SOM approach-

Data Loading & Preprocessing:

- Load data from 'A3-data.txt' into a DataFrame.
- Separate features (X) and the target (y).
- Apply Min-Max scaling to features.

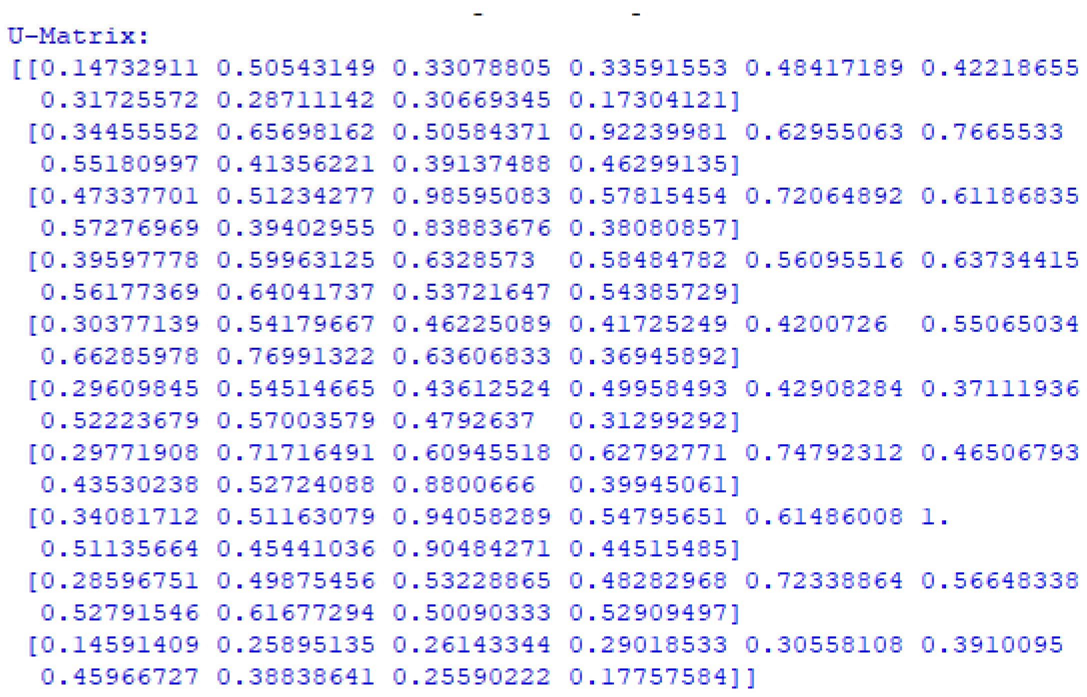
Self-Organizing Map (SOM):

- Create a 10x10 SOM.
- Initialize with random weights and train for 100 iterations.

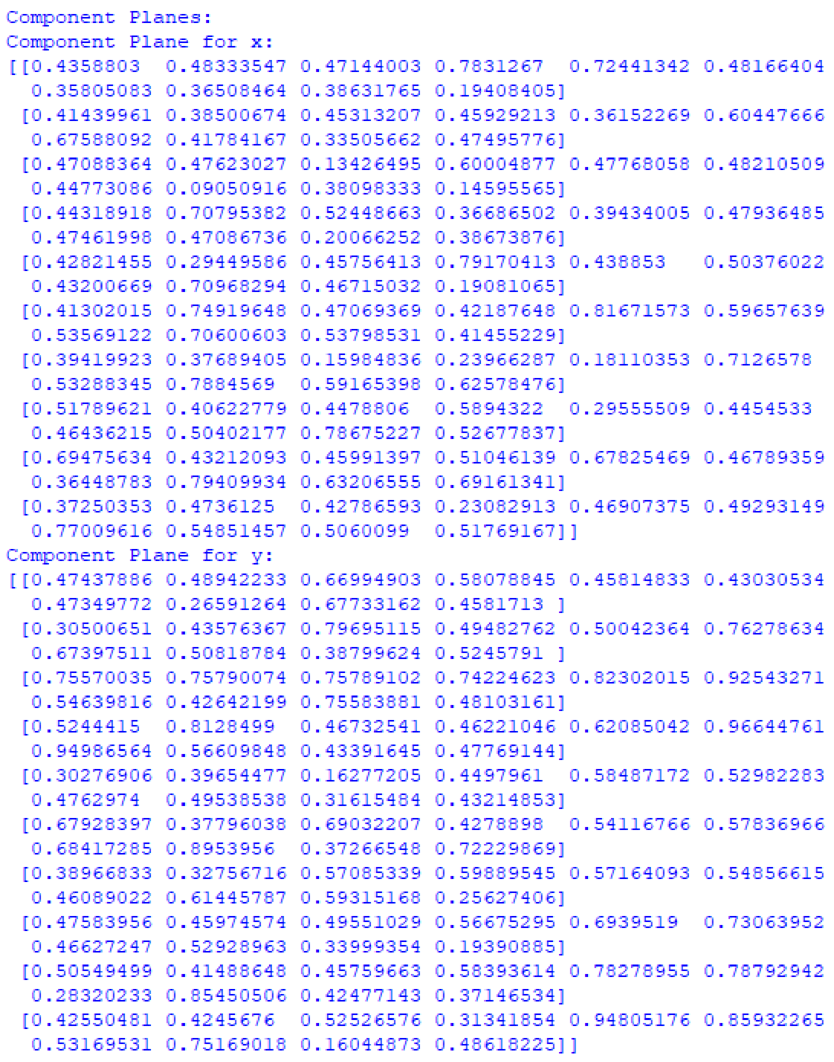
Visualizations:

- Display U-Matrix to show neighborhood relationships.
- Create component planes to visualize feature responses.
- Visualize data points on the SOM using markers and colors based on target variable.

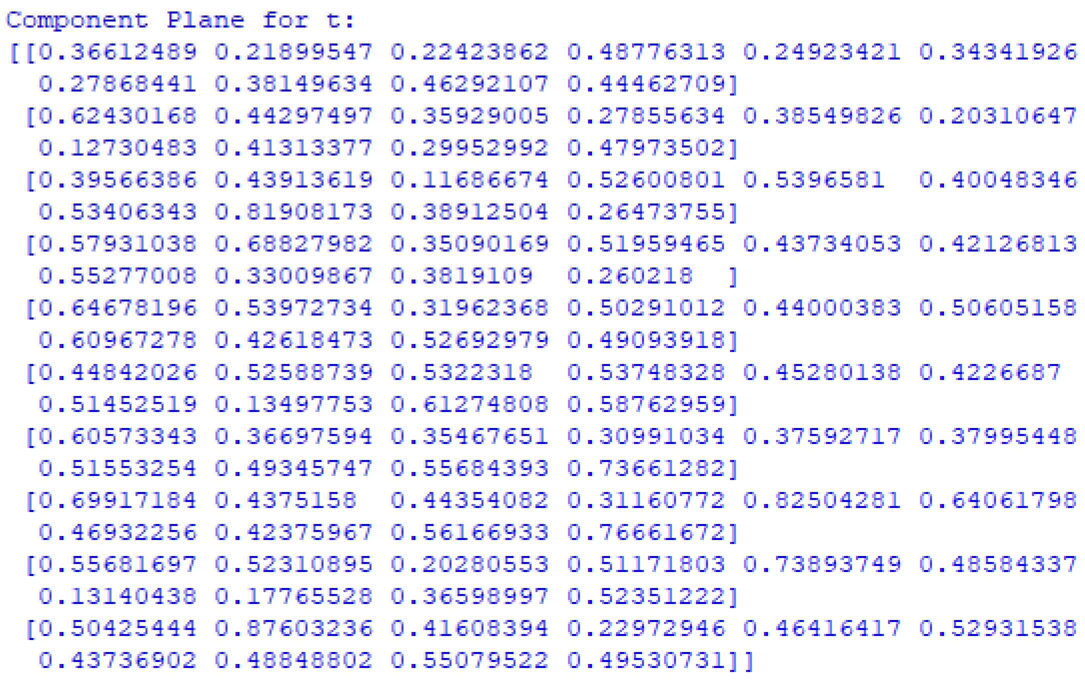
Result from Dataset1 (A3-data):



```
U-Matrix:
[[0.14732911 0.50543149 0.33078805 0.33591553 0.48417189 0.42218655
  0.31725572 0.28711142 0.30669345 0.17304121]
 [0.34455552 0.65698162 0.50584371 0.92239981 0.62955063 0.7665533
  0.55180997 0.41356221 0.39137488 0.46299135]
 [0.47337701 0.51234277 0.98595083 0.57815454 0.72064892 0.61186835
  0.57276969 0.39402955 0.83883676 0.38080857]
 [0.39597778 0.59963125 0.6328573  0.58484782 0.56095516 0.63734415
  0.56177369 0.64041737 0.53721647 0.54385729]
 [0.30377139 0.54179667 0.46225089 0.41725249 0.4200726  0.55065034
  0.66285978 0.76991322 0.63606833 0.36945892]
 [0.29609845 0.54514665 0.43612524 0.49958493 0.42908284 0.37111936
  0.52223679 0.57003579 0.4792637  0.31299292]
 [0.29771908 0.71716491 0.60945518 0.62792771 0.74792312 0.46506793
  0.43530238 0.52724088 0.8800666  0.39945061]
 [0.34081712 0.51163079 0.94058289 0.54795651 0.61486008 1.
  0.51135664 0.45441036 0.90484271 0.44515485]
 [0.28596751 0.49875456 0.53228865 0.48282968 0.72338864 0.56648338
  0.52791546 0.61677294 0.50090333 0.52909497]
 [0.14591409 0.25895135 0.26143344 0.29018533 0.30558108 0.3910095
  0.45966727 0.38838641 0.25590222 0.17757584]]
```



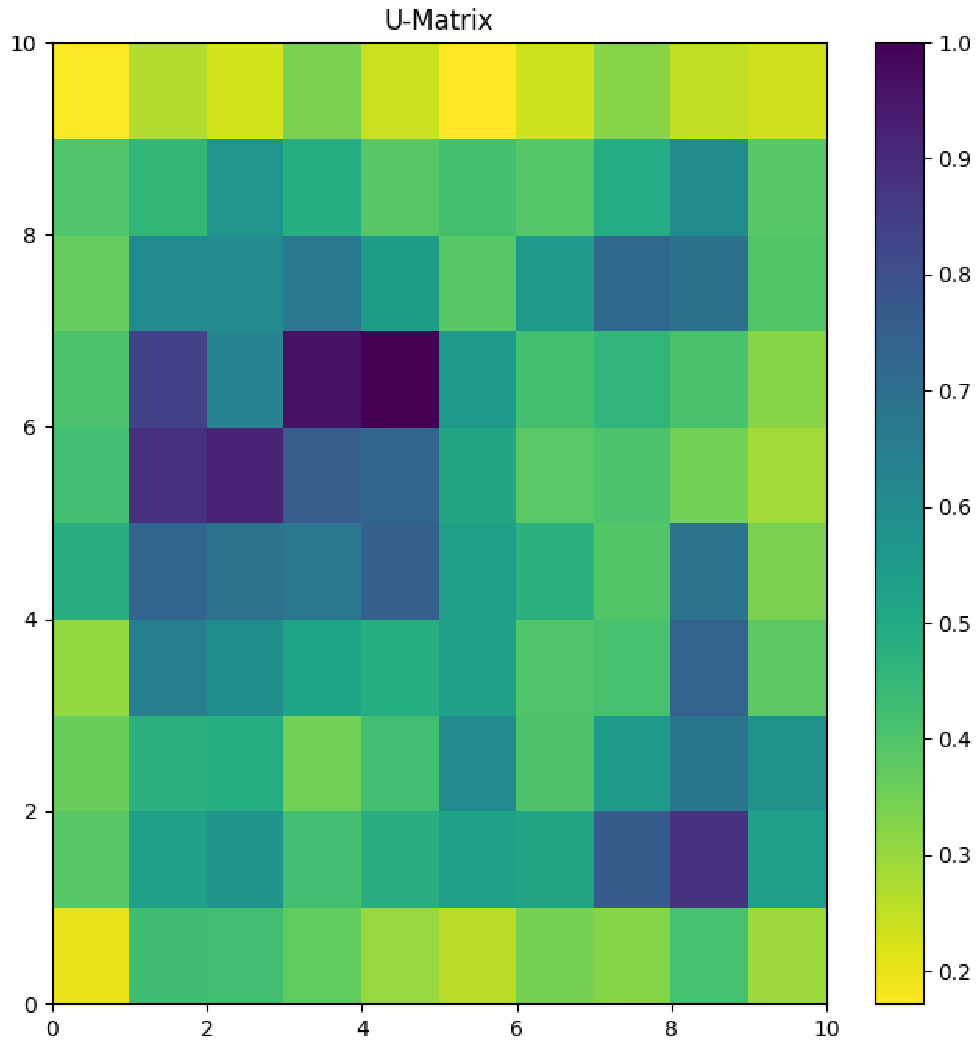
```
Component Planes:
Component Plane for x:
[[0.4358803  0.48333547 0.47144003 0.7831267  0.72441342 0.48166404
  0.35805083 0.36508464 0.38631765 0.19408405]
 [0.41439961 0.38500674 0.45313207 0.45929213 0.36152269 0.60447666
  0.67588092 0.41784167 0.33505662 0.47495776]
 [0.47088364 0.47623027 0.13426495 0.60004877 0.47768058 0.48210509
  0.44773086 0.09050916 0.38098333 0.14595565]
 [0.44318918 0.70795382 0.52448663 0.36686502 0.39434005 0.47936485
  0.47461998 0.47086736 0.20066252 0.38673876]
 [0.42821455 0.29449586 0.45756413 0.79170413 0.438853   0.50376022
  0.43200669 0.70968294 0.46715032 0.19081065]
 [0.41302015 0.74919648 0.47069369 0.42187648 0.81671573 0.59657639
  0.53569122 0.70600603 0.53798531 0.41455229]
 [0.39419923 0.37689405 0.15984836 0.23966287 0.18110353 0.7126578
  0.53288345 0.7884569  0.59165398 0.62578476]
 [0.51789621 0.40622779 0.4478806  0.5894322  0.29555509 0.4454533
  0.46436215 0.50402177 0.78675227 0.52677837]
 [0.69475634 0.43212093 0.45991397 0.51046139 0.67825469 0.46789359
  0.36448783 0.79409934 0.63206555 0.69161341]
 [0.37250353 0.4736125  0.42786593 0.23082913 0.46907375 0.49293149
  0.77009616 0.54851457 0.5060099  0.51769167]]
Component Plane for y:
[[0.47437886 0.48942233 0.66994903 0.58078845 0.45814833 0.43030534
  0.47349772 0.26591264 0.67733162 0.4581713 ]
 [0.30500651 0.43576367 0.79695115 0.49482762 0.50042364 0.76278634
  0.67397511 0.50818784 0.38799624 0.5245791 ]
 [0.75570035 0.75790074 0.75789102 0.74224623 0.82302015 0.92543271
  0.54639816 0.42642199 0.75583881 0.48103161]
 [0.5244415  0.8128499  0.46732541 0.46221046 0.62085042 0.96644761
  0.94986564 0.56609848 0.43391645 0.47769144]
 [0.30276906 0.39654477 0.16277205 0.4497961  0.58487172 0.52982283
  0.4762974  0.49538538 0.31615484 0.43214853]
 [0.67928397 0.37796038 0.69032207 0.4278898  0.54116766 0.57836966
  0.68417285 0.8953956  0.37266548 0.72229869]
 [0.38966833 0.32756716 0.57085339 0.59889545 0.57164093 0.54856615
  0.46089022 0.61445787 0.59315168 0.25627406]
 [0.47583956 0.45974574 0.49551029 0.56675295 0.6939519  0.73063952
  0.46627247 0.52928963 0.33999354 0.19390885]
 [0.50549499 0.41488648 0.45759663 0.58393614 0.78278955 0.78792942
  0.28320233 0.85450506 0.42477143 0.37146534]
 [0.42550481 0.4245676  0.52526576 0.31341854 0.94805176 0.85932265
  0.53169531 0.75169018 0.16044873 0.48618225]]
```

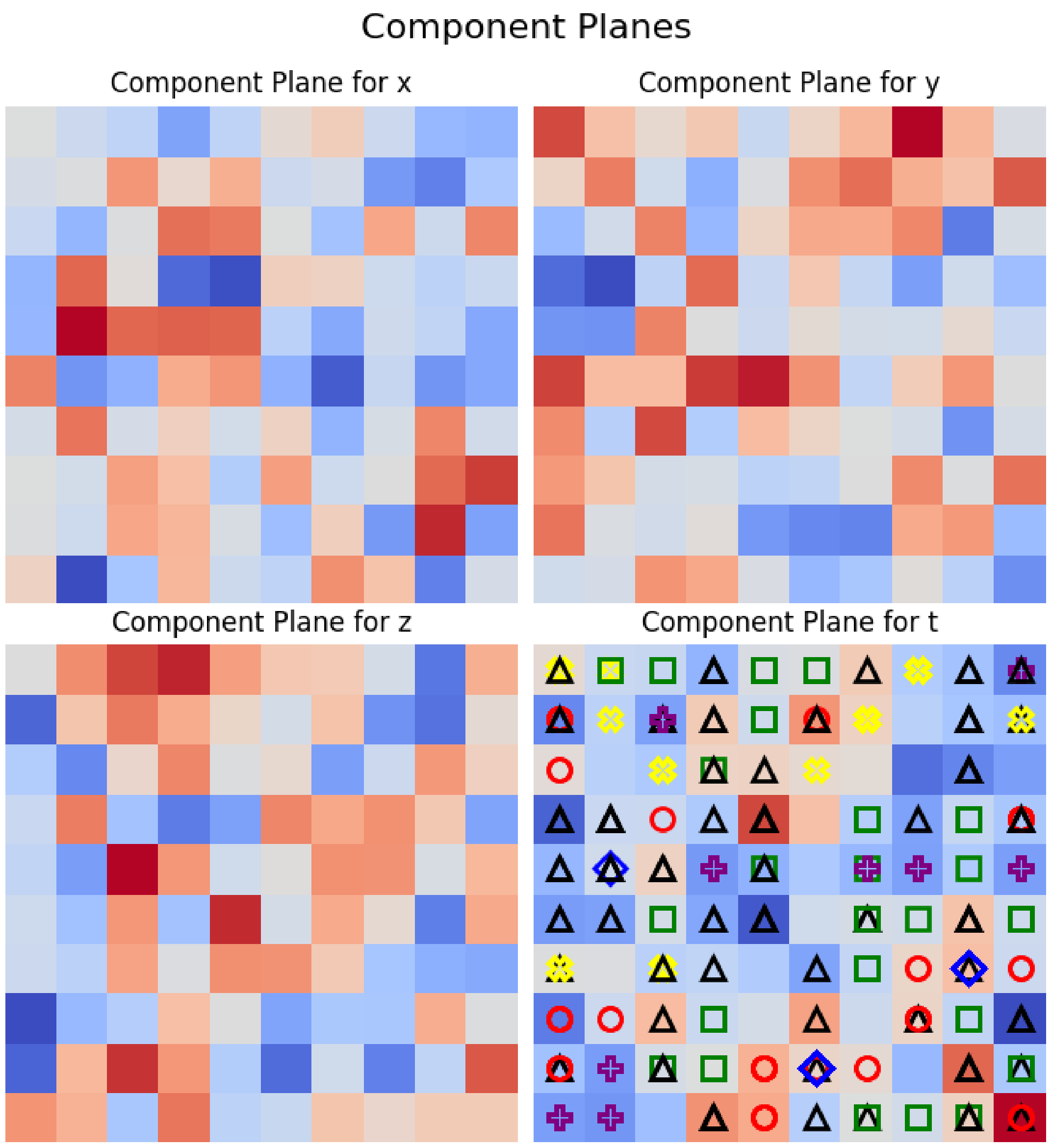


```
Component Plane for t:
[[0.36612489 0.21899547 0.22423862 0.48776313 0.24923421 0.34341926
  0.27868441 0.38149634 0.46292107 0.44462709]
 [0.62430168 0.44297497 0.35929005 0.27855634 0.38549826 0.20310647
  0.12730483 0.41313377 0.29952992 0.47973502]
 [0.39566386 0.43913619 0.11686674 0.52600801 0.5396581  0.40048346
  0.53406343 0.81908173 0.38912504 0.26473755]
 [0.57931038 0.68827982 0.35090169 0.51959465 0.43734053 0.42126813
  0.55277008 0.33009867 0.3819109  0.260218  ]
 [0.64678196 0.53972734 0.31962368 0.50291012 0.44000383 0.50605158
  0.60967278 0.42618473 0.52692979 0.49093918]
 [0.44842026 0.52588739 0.5322318  0.53748328 0.45280138 0.4226687
  0.51452519 0.13497753 0.61274808 0.58762959]
 [0.60573343 0.36697594 0.35467651 0.30991034 0.37592717 0.37995448
  0.51553254 0.49345747 0.55684393 0.73661282]
 [0.69917184 0.4375158  0.44354082 0.31160772 0.82504281 0.64061798
  0.46932256 0.42375967 0.56166933 0.76661672]
 [0.55681697 0.52310895 0.20280553 0.51171803 0.73893749 0.48584337
  0.13140438 0.17765528 0.36598997 0.52351222]
 [0.50425444 0.87603236 0.41608394 0.22972946 0.46416417 0.52931538
  0.43736902 0.48848802 0.55079522 0.49530731]]
```

```
Updated DataFrame with Winning Neurons:
            x         y         z         t  class  Winner_X  Winner_Y
0   -0.031676 -9.912054 -0.579436 -1.044239      1         9         8
1    0.002526  6.172456  3.288339 -1.006427      5         2         1
2    0.183123 -0.387841  6.236470 -1.691491      2         3         2
3   -0.042262 -1.996272 -1.655302 -2.995311      1         8         2
4   -0.062811 -0.417072  6.657475 -3.633134      4         3         9
..        ...       ...       ...       ...    ...       ...       ...
355 -0.340733  8.504536  7.903644 -2.032197      6         9         4
356 -0.024928  1.551977  6.361992 -0.757714      2         7         2
357 -0.668529 -0.607597  0.639295  4.329213      6         2         7
358 -0.122711 -5.516957 -2.111173  2.209675      3         4         0
359  0.740207 -7.492176  1.959426 -7.119918      6         0         4

[360 rows x 7 columns]
```





SOM Approach for Dataset2 (students- knowledge-level-data): Breakdown of SOM approach-

Data Loading & Preprocessing:

- Load data from 'students- knowledge-level-data-numeric.txt' into a DataFrame.
- Separate features (X) and the target (y).
- Apply Min-Max scaling to features.

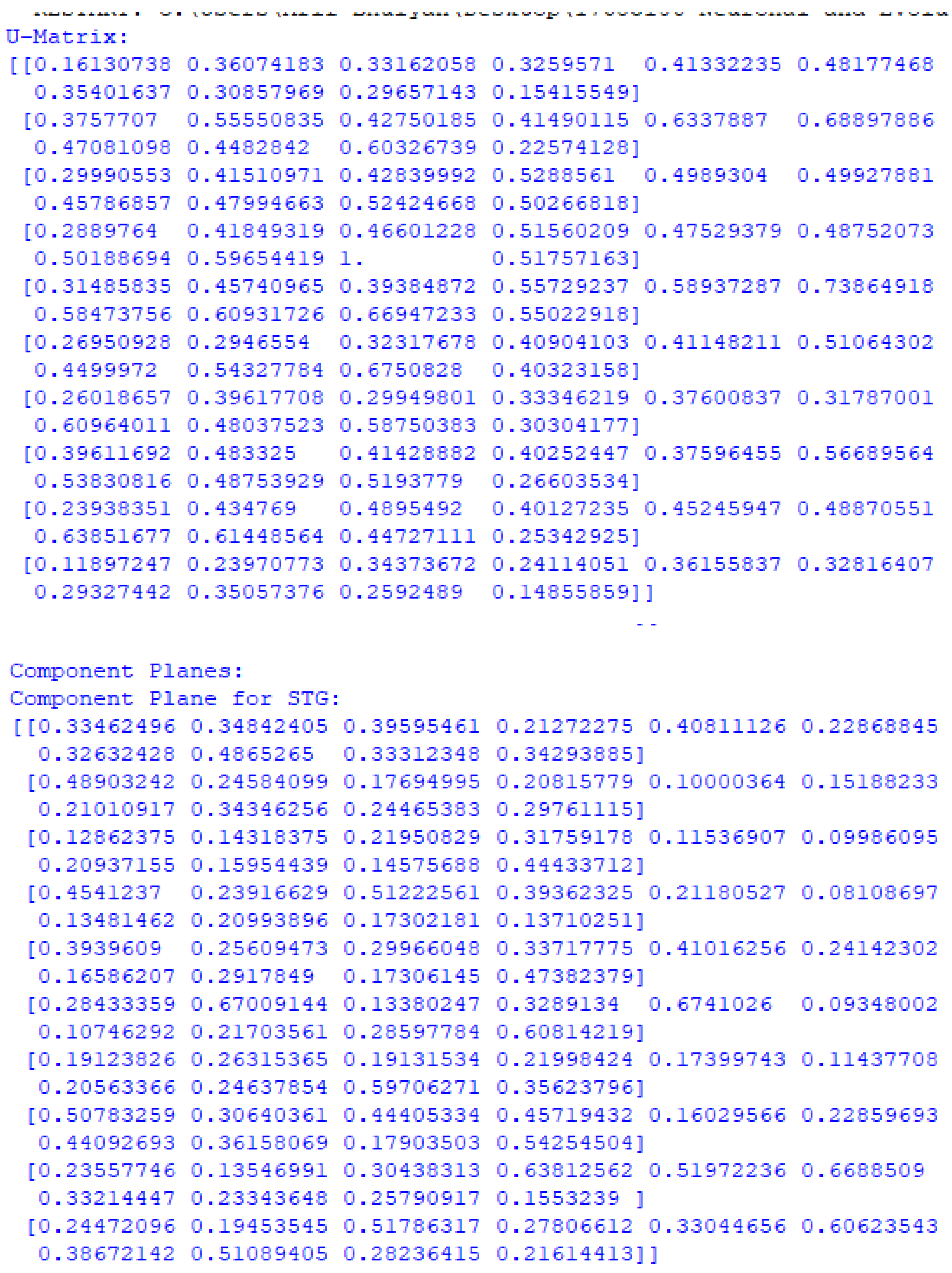
Self-Organizing Map (SOM):

- Create a 10x10 SOM.
- Initialize with random weights and train for 100 iterations.

Visualizations:

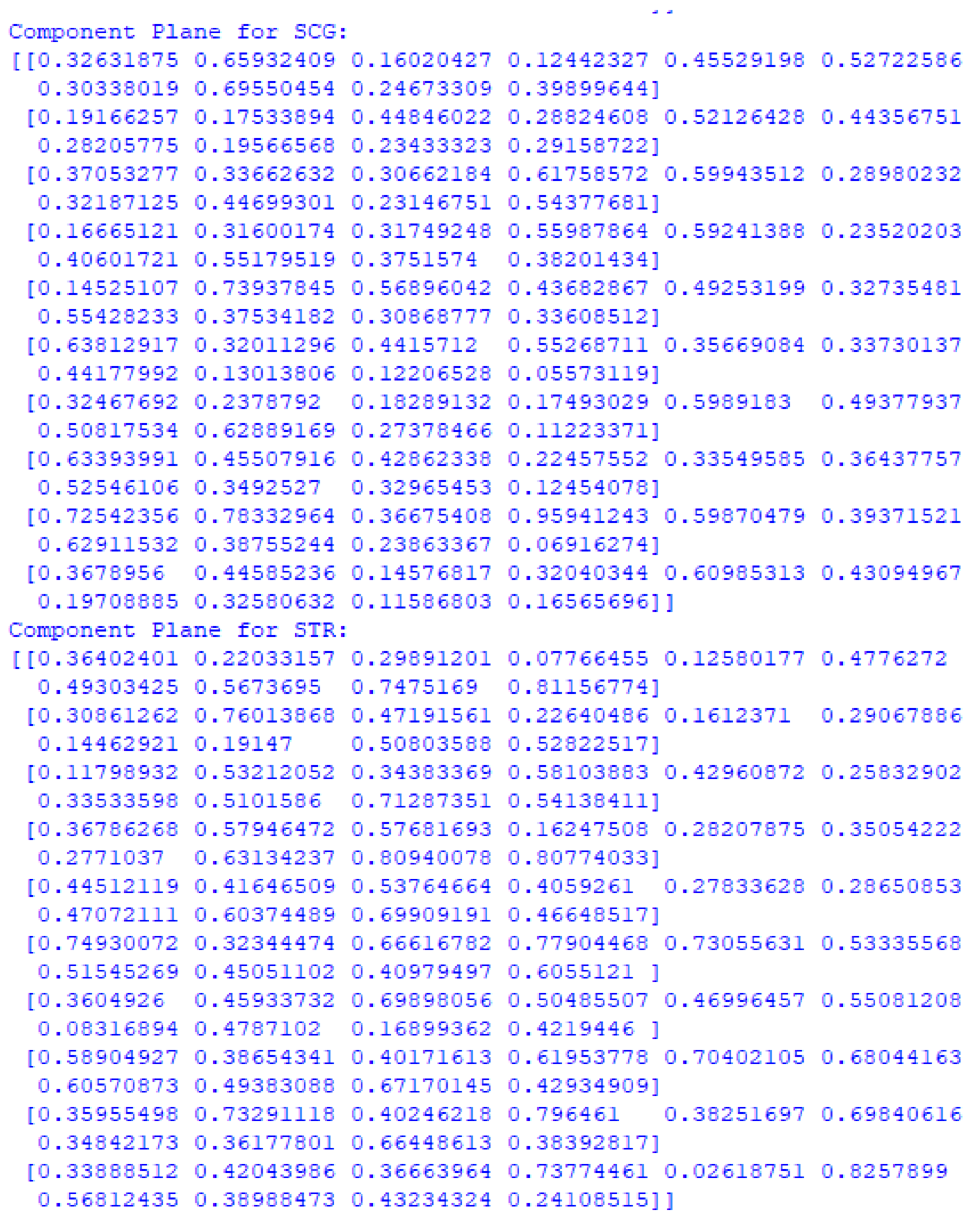
- Display U-Matrix to visualize neighborhood relationships.
- Create component planes to visualize feature responses.
- Visualize data points on the SOM using markers and colors based on the target variable.

Result from Dataset2 (students- knowledge-level-data):

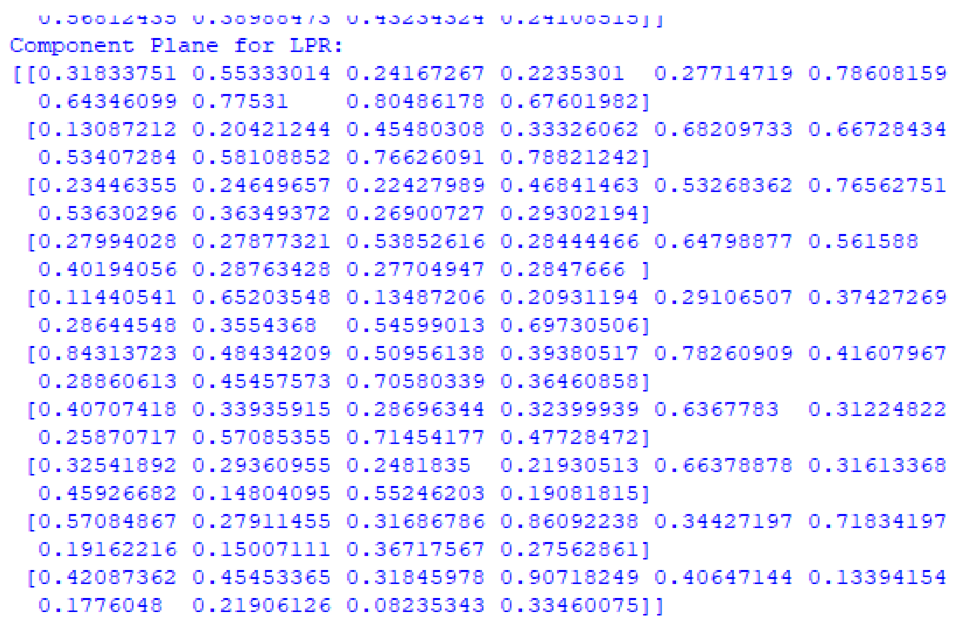


```
U-Matrix:
[[0.16130738 0.36074183 0.33162058 0.3259571  0.41332235 0.48177468
  0.35401637 0.30857969 0.29657143 0.15415549]
 [0.3757707  0.55550835 0.42750185 0.41490115 0.6337887  0.68897886
  0.47081098 0.4482842  0.60326739 0.22574128]
 [0.29990553 0.41510971 0.42839992 0.5288561  0.4989304  0.49927881
  0.45786857 0.47994663 0.52424668 0.50266818]
 [0.2889764  0.41849319 0.46601228 0.51560209 0.47529379 0.48752073
  0.50188694 0.59654419 1.         0.51757163]
 [0.31485835 0.45740965 0.39384872 0.55729237 0.58937287 0.73864918
  0.58473756 0.60931726 0.66947233 0.55022918]
 [0.26950928 0.2946554  0.32317678 0.40904103 0.41148211 0.51064302
  0.4499972  0.54327784 0.6750828  0.40323158]
 [0.26018657 0.39617708 0.29949801 0.33346219 0.37600837 0.31787001
  0.60964011 0.48037523 0.58750383 0.30304177]
 [0.39611692 0.483325   0.41428882 0.40252447 0.37596455 0.56689564
  0.53830816 0.48753929 0.5193779  0.26603534]
 [0.23938351 0.434769   0.4895492  0.40127235 0.45245947 0.48870551
  0.63851677 0.61448564 0.44727111 0.25342925]
 [0.11897247 0.23970773 0.34373672 0.24114051 0.36155837 0.32816407
  0.29327442 0.35057376 0.2592489  0.14855859]]

Component Planes:
Component Plane for STG:
[[0.33462496 0.34842405 0.39595461 0.21272275 0.40811126 0.22868845
  0.32632428 0.4865265  0.33312348 0.34293885]
 [0.48903242 0.24584099 0.17694995 0.20815779 0.10000364 0.15188233
  0.21010917 0.34346256 0.24465383 0.29761115]
 [0.12862375 0.14318375 0.21950829 0.31759178 0.11536907 0.09986095
  0.20937155 0.15954439 0.14575688 0.44433712]
 [0.4541237  0.23916629 0.51222561 0.39362325 0.21180527 0.08108697
  0.13481462 0.20993896 0.17302181 0.13710251]
 [0.3939609  0.25609473 0.29966048 0.33717775 0.41016256 0.24142302
  0.16586207 0.2917849  0.17306145 0.47382379]
 [0.28433359 0.67009144 0.13380247 0.3289134  0.6741026  0.09348002
  0.10746292 0.21703561 0.28597784 0.60814219]
 [0.19123826 0.26315365 0.19131534 0.21998424 0.17399743 0.11437708
  0.20563366 0.24637854 0.59706271 0.35623796]
 [0.50783259 0.30640361 0.44405334 0.45719432 0.16029566 0.22859693
  0.44092693 0.36158069 0.17903503 0.54254504]
 [0.23557746 0.13546991 0.30438313 0.63812562 0.51972236 0.6688509
  0.33214447 0.23343648 0.25790917 0.1553239 ]
 [0.24472096 0.19453545 0.51786317 0.27806612 0.33044656 0.60623543
  0.38672142 0.51089405 0.28236415 0.21614413]]
```



```
Component Plane for SCG:
[[0.32631875 0.65932409 0.16020427 0.12442327 0.45529198 0.52722586
  0.30338019 0.69550454 0.24673309 0.39899644]
 [0.19166257 0.17533894 0.44846022 0.28824608 0.52126428 0.44356751
  0.28205775 0.19566568 0.23433323 0.29158722]
 [0.37053277 0.33662632 0.30662184 0.61758572 0.59943512 0.28980232
  0.32187125 0.44699301 0.23146751 0.54377681]
 [0.16665121 0.31600174 0.31749248 0.55987864 0.59241388 0.23520203
  0.40601721 0.55179519 0.3751574  0.38201434]
 [0.14525107 0.73937845 0.56896042 0.43682867 0.49253199 0.32735481
  0.55428233 0.37534182 0.30868777 0.33608512]
 [0.63812917 0.32011296 0.4415712  0.55268711 0.35669084 0.33730137
  0.44177992 0.13013806 0.12206528 0.05573119]
 [0.32467692 0.2378792  0.18289132 0.17493029 0.5989183  0.49377937
  0.50817534 0.62889169 0.27378466 0.11223371]
 [0.63393991 0.45507916 0.42862338 0.22457552 0.33549585 0.36437757
  0.52546106 0.3492527  0.32965453 0.12454078]
 [0.72542356 0.78332964 0.36675408 0.95941243 0.59870479 0.39371521
  0.62911532 0.38755244 0.23863367 0.06916274]
 [0.3678956  0.44585236 0.14576817 0.32040344 0.60985313 0.43094967
  0.19708885 0.32580632 0.11586803 0.16565696]]
Component Plane for STR:
[[0.36402401 0.22033157 0.29891201 0.07766455 0.12580177 0.4776272
  0.49303425 0.5673695  0.7475169  0.81156774]
 [0.30861262 0.76013868 0.47191561 0.22640486 0.1612371  0.29067886
  0.14462921 0.19147    0.50803588 0.52822517]
 [0.11798932 0.53212052 0.34383369 0.58103883 0.42960872 0.25832902
  0.33533598 0.5101586  0.71287351 0.54138411]
 [0.36786268 0.57946472 0.57681693 0.16247508 0.28207875 0.35054222
  0.2771037  0.63134237 0.80940078 0.80774033]
 [0.44512119 0.41646509 0.53764664 0.4059261  0.27833628 0.28650853
  0.47072111 0.60374489 0.69909191 0.46648517]
 [0.74930072 0.32344474 0.66616782 0.77904468 0.73055631 0.53335568
  0.51545269 0.45051102 0.40979497 0.6055121 ]
 [0.3604926  0.45933732 0.69898056 0.50485507 0.46996457 0.55081208
  0.08316894 0.4787102  0.16899362 0.4219446 ]
 [0.58904927 0.38654341 0.40171613 0.61953778 0.70402105 0.68044163
  0.60570873 0.49383088 0.67170145 0.42934909]
 [0.35955498 0.73291118 0.40246218 0.796461   0.38251697 0.69840616
  0.34842173 0.36177801 0.66448613 0.38392817]
 [0.33888512 0.42043986 0.36663964 0.73774461 0.02618751 0.8257899
  0.56812435 0.38988473 0.43234324 0.24108515]]
```



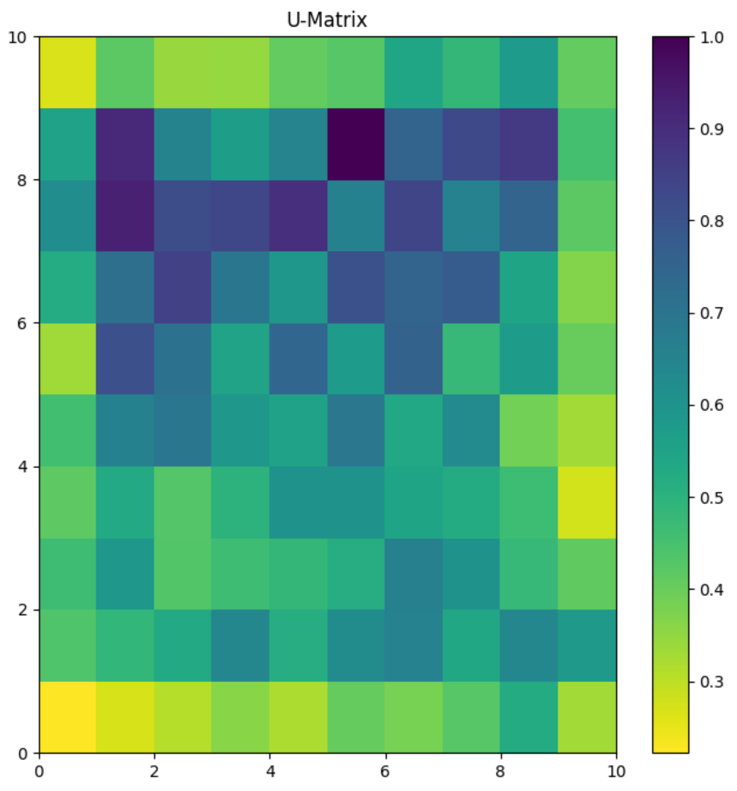
0.56812435 0.38988473 0.43234324 0.24108515]]

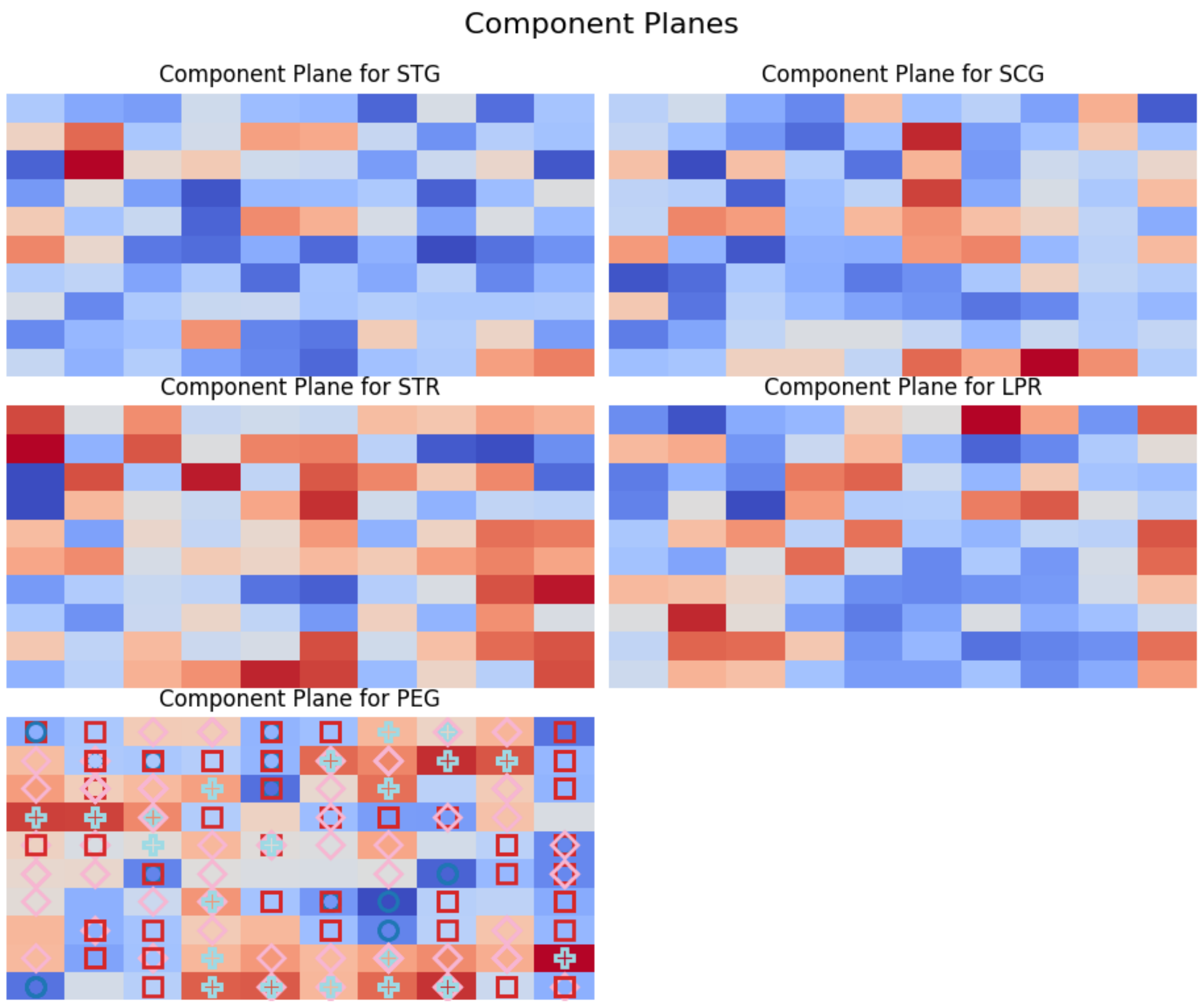
```
Component Plane for LPR:
[[0.31833751 0.55333014 0.24167267 0.2235301  0.27714719 0.78608159
  0.64346099 0.77531    0.80486178 0.67601982]
 [0.13087212 0.20421244 0.45480308 0.33326062 0.68209733 0.66728434
  0.53407284 0.58108852 0.76626091 0.78821242]
 [0.23446355 0.24649657 0.22427989 0.46841463 0.53268362 0.76562751
  0.53630296 0.36349372 0.26900727 0.29302194]
 [0.27994028 0.27877321 0.53852616 0.28444466 0.64798877 0.561588
  0.40194056 0.28763428 0.27704947 0.2847666 ]
 [0.11440541 0.65203548 0.13487206 0.20931194 0.29106507 0.37427269
  0.28644548 0.3554368  0.54599013 0.69730506]
 [0.84313723 0.48434209 0.50956138 0.39380517 0.78260909 0.41607967
  0.28860613 0.45457573 0.70580339 0.36460858]
 [0.40707418 0.33935915 0.28696344 0.32399939 0.6367783  0.31224822
  0.25870717 0.57085355 0.71454177 0.47728472]
 [0.32541892 0.29360955 0.2481835  0.21930513 0.66378878 0.31613368
  0.45926682 0.14804095 0.55246203 0.19081815]
 [0.57084867 0.27911455 0.31686786 0.86092238 0.34427197 0.71834197
  0.19162216 0.15007111 0.36717567 0.27562861]
 [0.42087362 0.45453365 0.31845978 0.90718249 0.40647144 0.13394154
  0.1776048  0.21906126 0.08235343 0.33460075]]
```

```
Component Plane for PEG:
[[0.32060565 0.20550137 0.10878496 0.25489131 0.42623748 0.27581608
  0.50737566 0.61131399 0.29498647 0.2182044 ]
 [0.58727793 0.24511212 0.26790851 0.52050466 0.12353153 0.23707847
  0.29451541 0.26032994 0.34530884 0.30312114]
 [0.30457108 0.21784812 0.64104452 0.39427345 0.33054768 0.21191771
  0.38498788 0.59260528 0.44457043 0.6431085 ]
 [0.19288602 0.53586014 0.67448127 0.80659091 0.13911287 0.30258801
  0.4673753  0.814991   0.84679476 0.70153649]
 [0.69154419 0.33052116 0.39834916 0.65665458 0.91340627 0.54561215
  0.68291173 0.61324179 0.44891335 0.6440824 ]
 [0.3312566  0.83408671 0.41352567 0.58457347 0.66890976 0.34435384
  0.64967921 0.19013035 0.16744405 0.29553547]
 [0.40521124 0.85218842 0.78471444 0.6159447  0.14969953 0.63065265
  0.79688583 0.58184917 0.37939641 0.30667017]
 [0.5519028  0.86856868 0.73456362 0.25715462 0.21395957 0.42176062
  0.84972369 0.71555358 0.91231361 0.74821253]
 [0.36053623 0.56578063 0.3723714  0.77929615 0.30693932 0.74068513
  0.93334933 0.35060183 0.50264289 0.87161619]
 [0.31796859 0.4632899  0.14373375 0.6192045  0.84734285 0.69919936
  0.66571986 0.43442572 0.72527846 0.54407411]]

Updated DataFrame with Winning Neurons:
      STG   SCG   STR   LPR   PEG  UNS  Winner_X  Winner_Y
0    0.00  0.00  0.00  0.00  0.00    3         0         3
1    0.08  0.08  0.10  0.24  0.90    8         8         9
2    0.06  0.06  0.05  0.25  0.33    4         0         3
3    0.10  0.10  0.15  0.65  0.30    6         1         6
4    0.08  0.08  0.08  0.98  0.24    4         2         5
..    ...   ...   ...   ...   ...  ...       ...       ...
253  0.61  0.78  0.69  0.92  0.58    8         8         3
254  0.78  0.61  0.71  0.19  0.60    6         9         5
255  0.54  0.82  0.71  0.29  0.77    8         7         0
256  0.50  0.75  0.81  0.61  0.26    6         5         0
257  0.66  0.90  0.76  0.87  0.74    8         8         3

[258 rows x 8 columns]
```





Conclusion: We have evaluated five different unsupervised learning techniques (AHC, K-Means, PCA, t-SNE, and SOM) on our dataset1 and Datase2 and obtained various results. Here are key points to consider:

Cluster Quality: AHC, K-Means, and SOM produced similar cluster assignments, indicating potential consistency in their clustering results. t-SNE does not directly provide cluster assignments but offers a lower-dimensional representation for visualization.

Silhouette Score: Silhouette scores, which measure cluster quality, were not provided. Calculating and comparing these scores could offer additional insights.

Dimensionality Reduction: PCA and t-SNE reduced the dataset's dimensionality for visualization purposes, with PCA preserving variance and t-SNE focusing on local similarities.

SOM U-Matrix and Component Planes: SOM's U-Matrix and Component Planes offer insights into data topology and representation.

Computational Complexity: Consider the computational resources required by each algorithm, as they vary in terms of runtime and memory usage.

AHC, K-Means, and SOM seem promising for clustering, while PCA and t-SNE are useful for dimensionality reduction and visualization.

The choice of the most appropriate algorithm depends on our specific goals, data characteristics, and computational constraints.