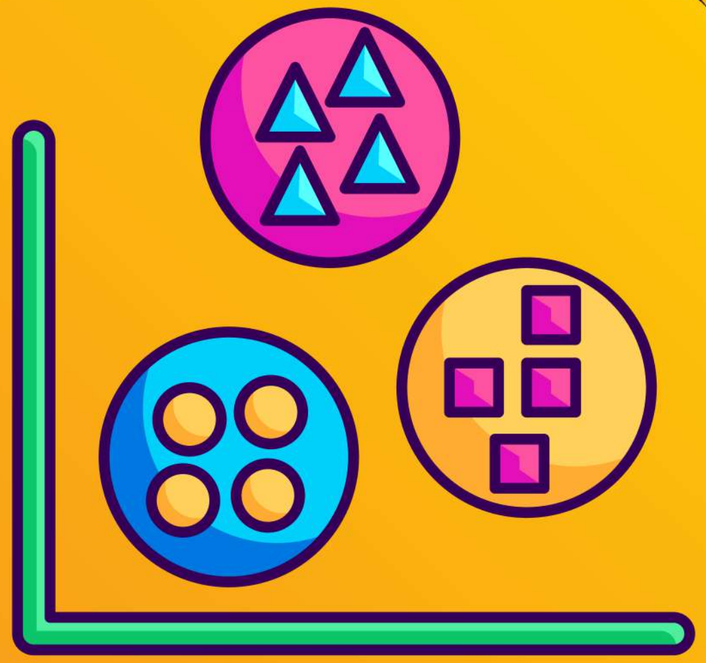
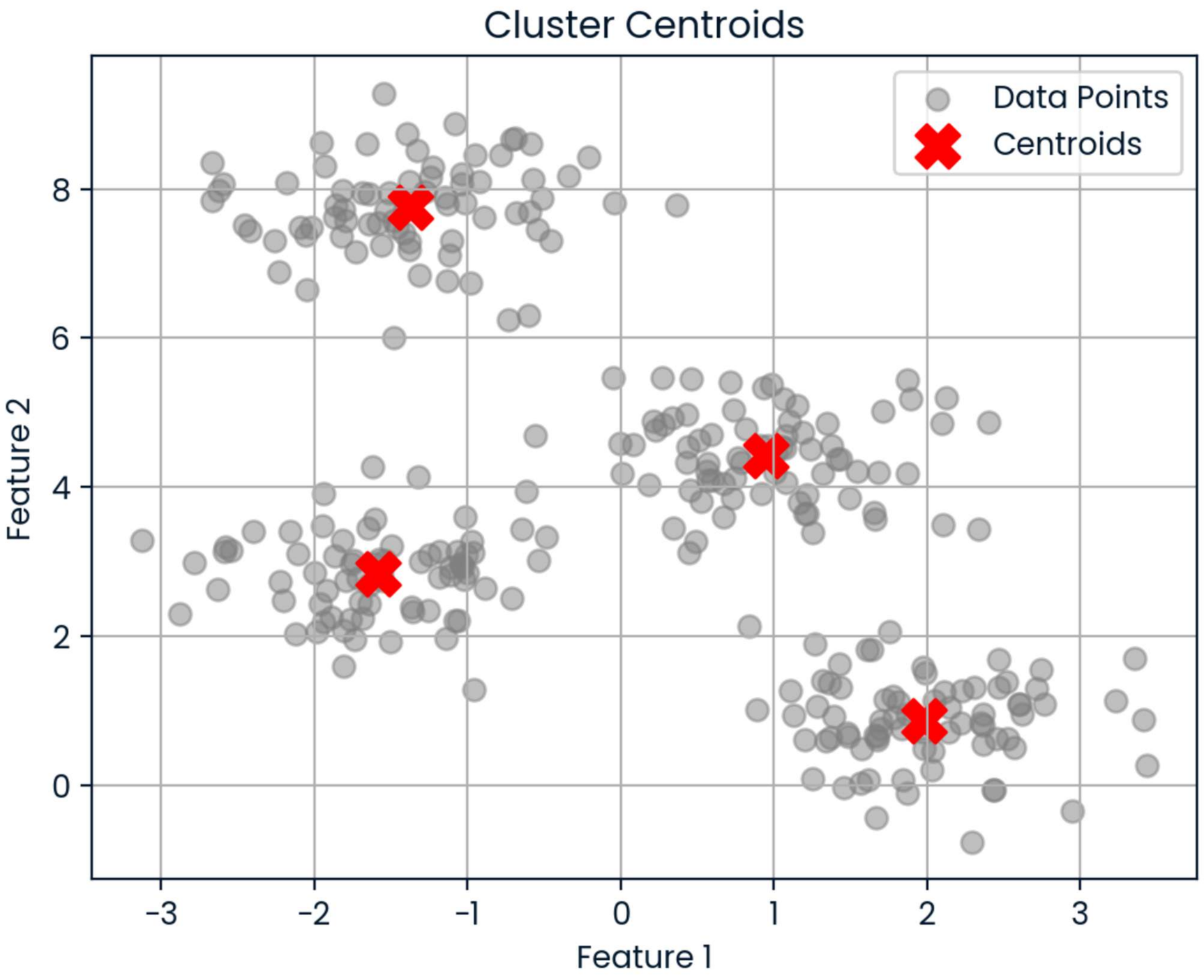


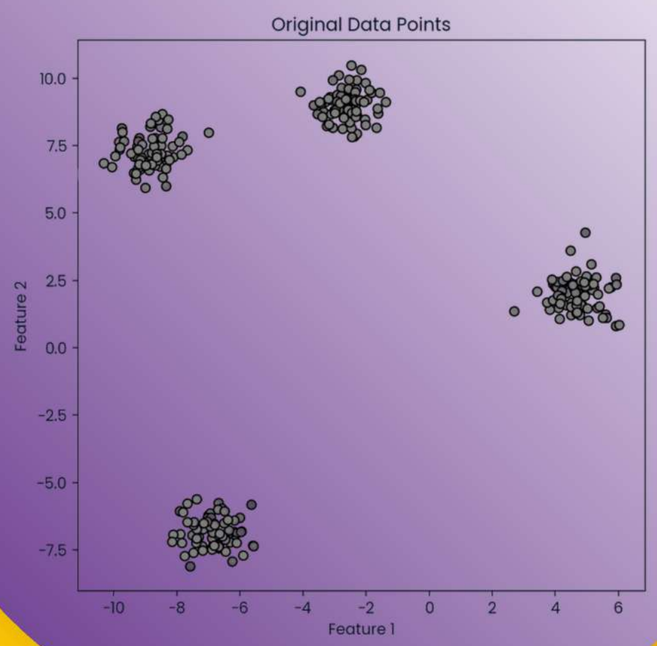
UNIVERSITY OF ALBERTA

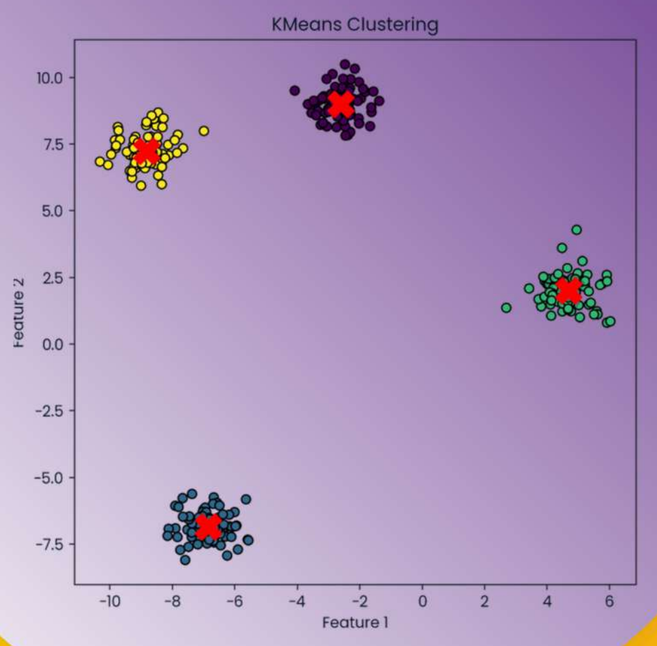
Compairing Clustering Algorithms

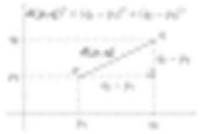
Aashish Kumar


Cluster Centroids
Data Points
Centroids
Feature 2
Feature 1
8
6
4
2
0
-3
-2
-1
0
1
2
3

KMeans Clustering







Euclidean distance

Overview | Formula

Formula

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

p, q = two points in Euclidean n-space

q_i, p_i = Euclidean vectors, starting from the origin of the space (initial point)

n = n-space

```
#Transpose the data to have observations as rows and features as columns
data = data.T
```

```
print(data.head())
```

```
Dimensions: (459, 2207)
          0         1    2         3     ...      2203      2204      2205      2206
0  0.048734  0.156177  0.0  0.504464  ...  0.068796  0.068681  0.101562  0.082707
1  0.111323  0.074072  0.0  0.571429  ...  0.194103  0.197802  0.117188  0.150376
2  0.064023  0.080148  0.0  0.473214  ...  0.181818  0.162088  0.195312  0.132832
3  0.058767  0.131366  0.0  0.410714  ...  0.031941  0.024725  0.109375  0.162907
4  0.050645  0.147004  0.0  0.446429  ...  0.140049  0.131868  0.148438  0.170426

[5 rows x 2207 columns]
```

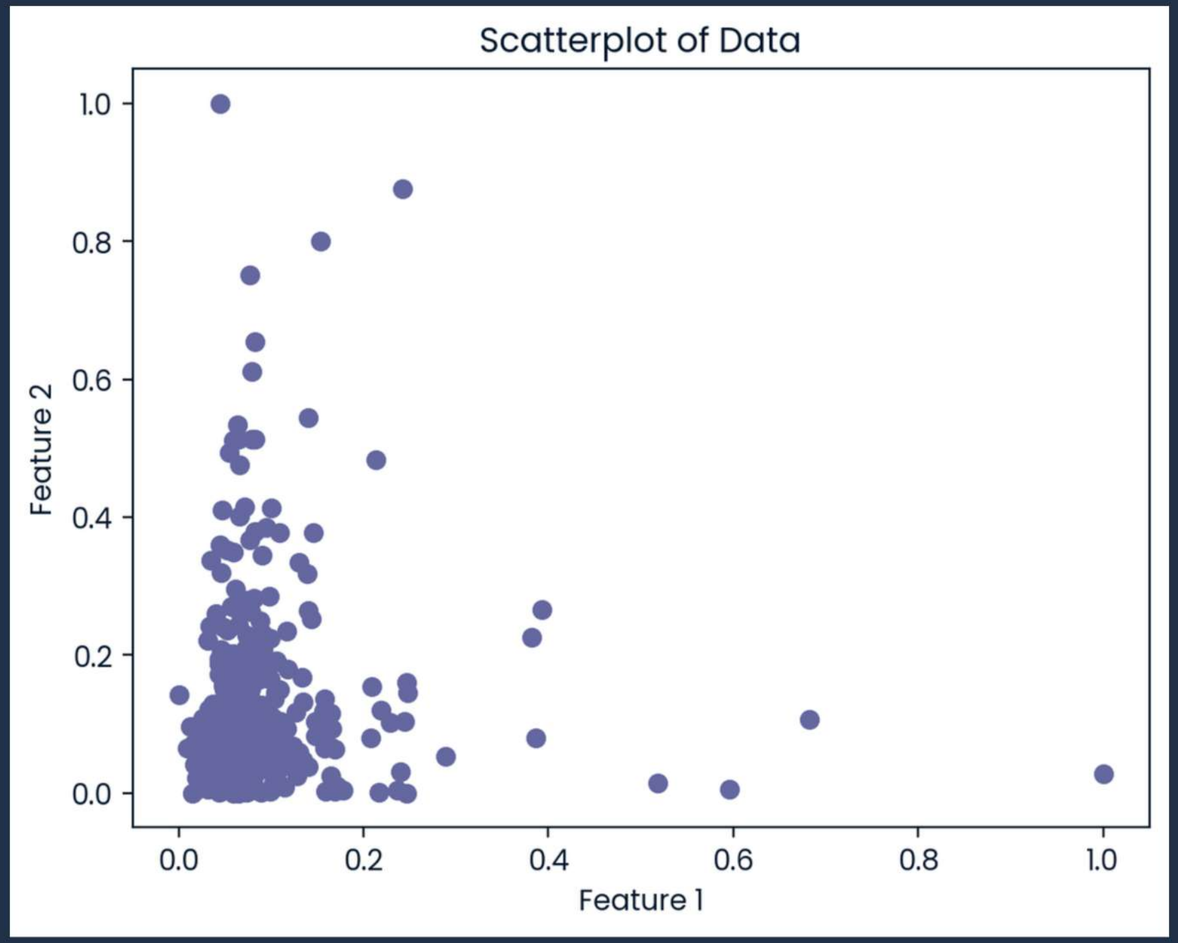
```
print(data.describe())
```

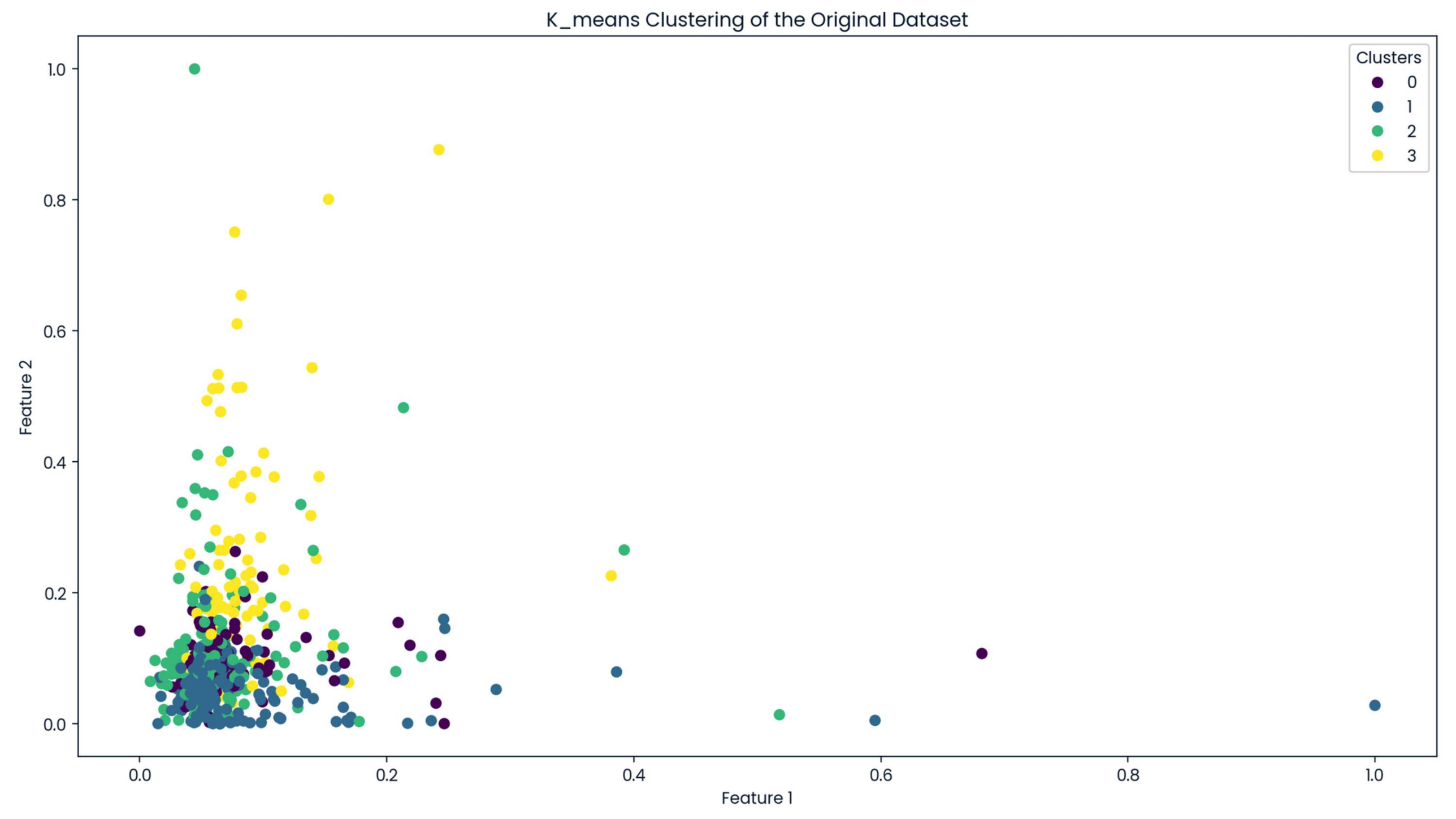
```
                0           1    2     ...         2204         2205         2206
count  459.000000  459.000000  459.0  ...  459.000000  459.000000  459.000000
mean     0.080778    0.120589    0.0  ...    0.251107    0.204572    0.196368
std      0.077852    0.125591    0.0  ...    0.159509    0.157142    0.120650
min      0.000000    0.000000    0.0  ...    0.000000    0.000000    0.000000
25%      0.049212    0.050928    0.0  ...    0.127747    0.101562    0.110276
50%      0.061634    0.086378    0.0  ...    0.225275    0.156250    0.177945
75%      0.084090    0.145699    0.0  ...    0.329670    0.242188    0.250627
max      1.000000    1.000000    0.0  ...    1.000000    1.000000    1.000000

[8 rows x 2207 columns]
```

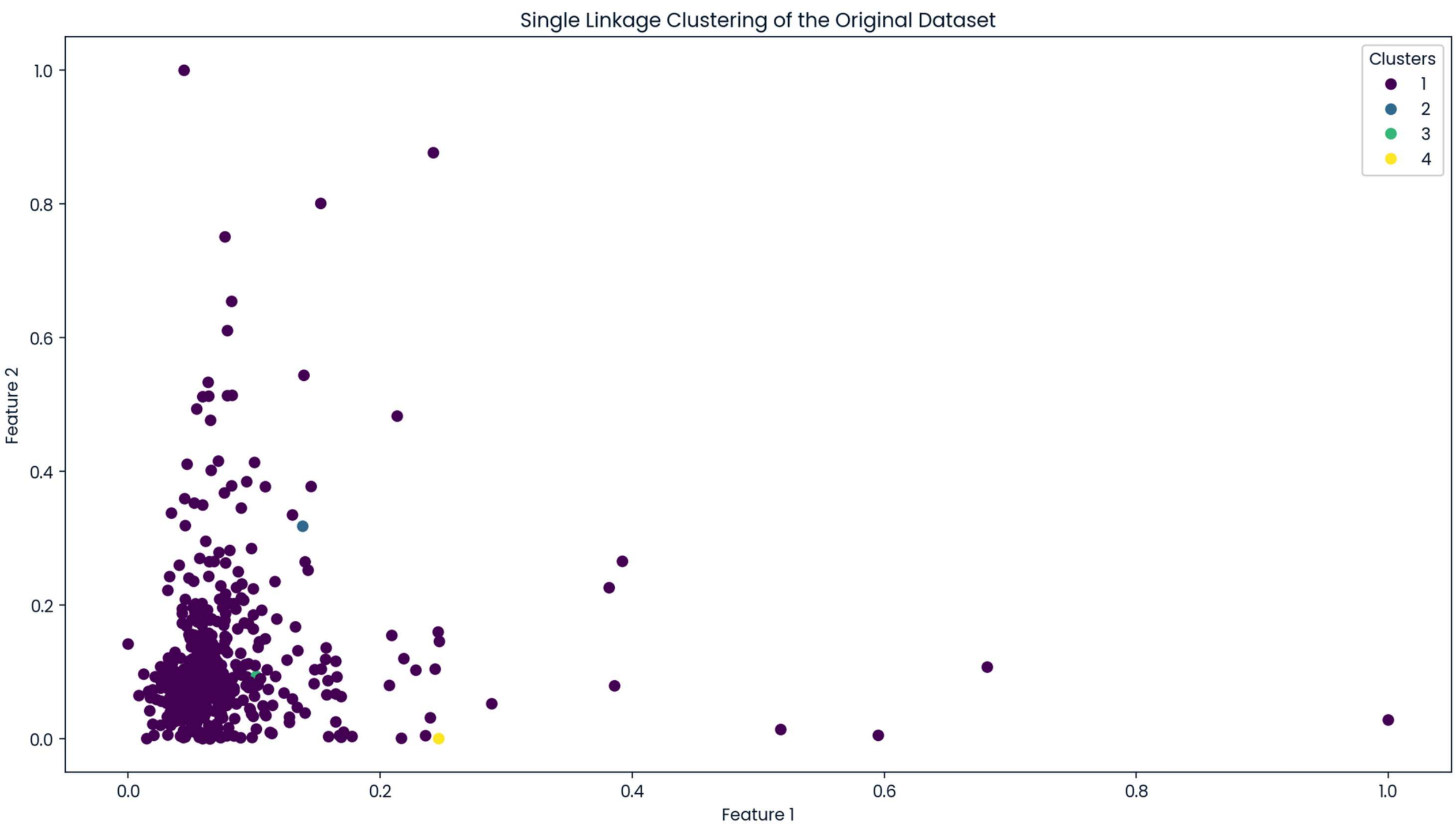
```
print("Emptyvalues:", data.isna().sum().sum())
print("Duplicates:", data.duplicated().sum())
```

```
Emptyvalues: 0
Duplicates: 0
```

- No need to scale or standardize the data.
- No empty or duplicate values are present.
- Some rows or columns may contain the same values (e.g., zeros).
- Removing uniform values will not impact clustering results.
- Uniform values do not influence overall clustering outcomes.
- Scaling is unnecessary.




K_means Clustering of the Original Dataset
Clusters
0
1
2
3
Feature 2
Feature 1
0.0
0.2
0.4
0.6
0.8
1.0


Single Linkage Clustering of the Original Dataset
Clusters
1
2
3
4
Feature 1
Feature 2
0.0
0.2
0.4
0.6
0.8
1.0

Silhouette Score: Understanding the Metric

The silhouette score is a metric used to evaluate the quality of clustering. It measures how well each data point fits within its assigned cluster compared to other clusters. A higher silhouette score indicates that the clusters are well-defined and distinct.

For a data point i, the silhouette score is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

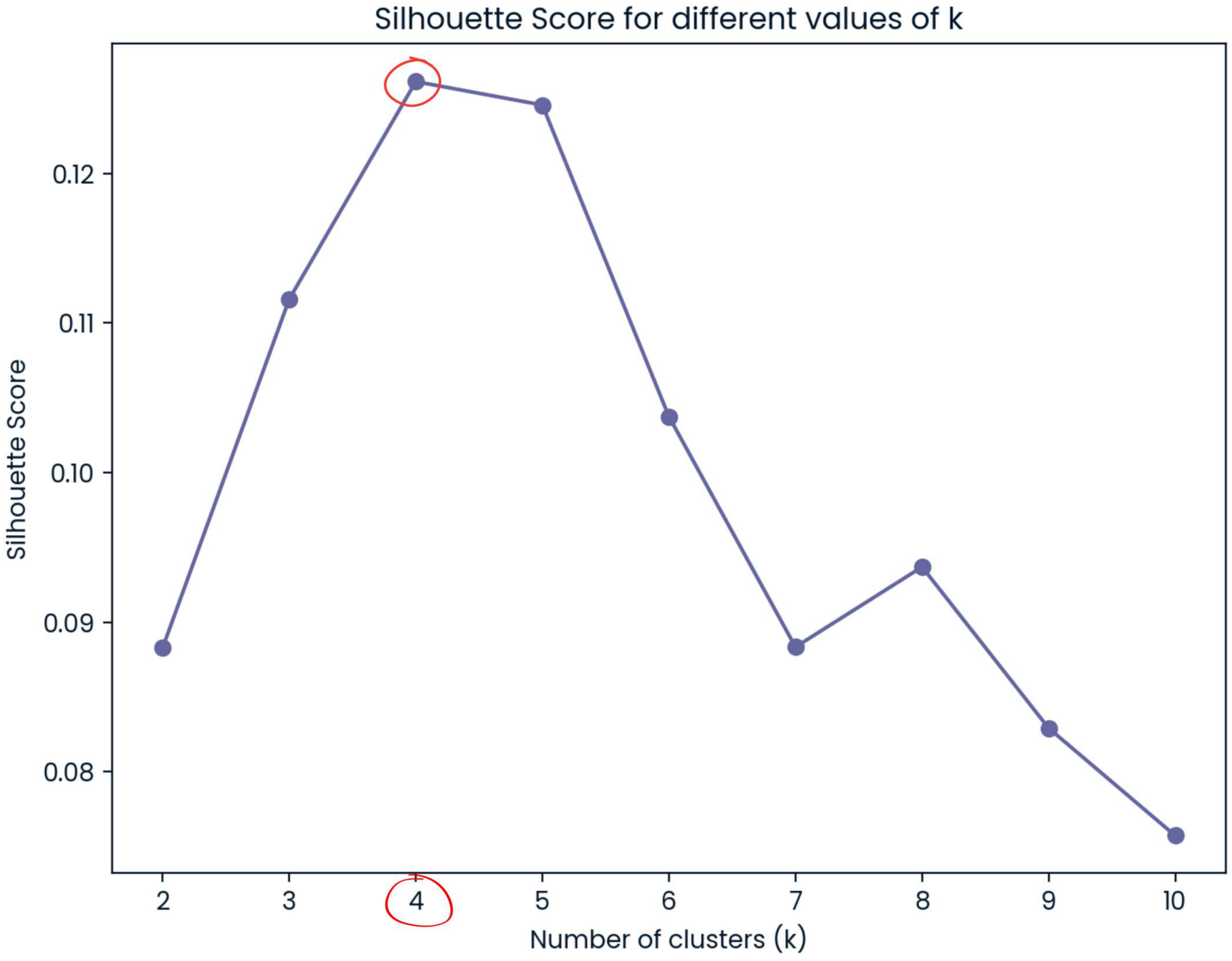
Where:

- $a(i)$: The **average distance** between i and all other points in the **same cluster**.
- $b(i)$: The **average distance** between i and all points in the **nearest cluster** (the next best cluster to which i could belong).
- $s(i)$: Silhouette score for point i, ranges from -1 to $+1$.

The overall silhouette score is the mean of s(i) across all points in the dataset:

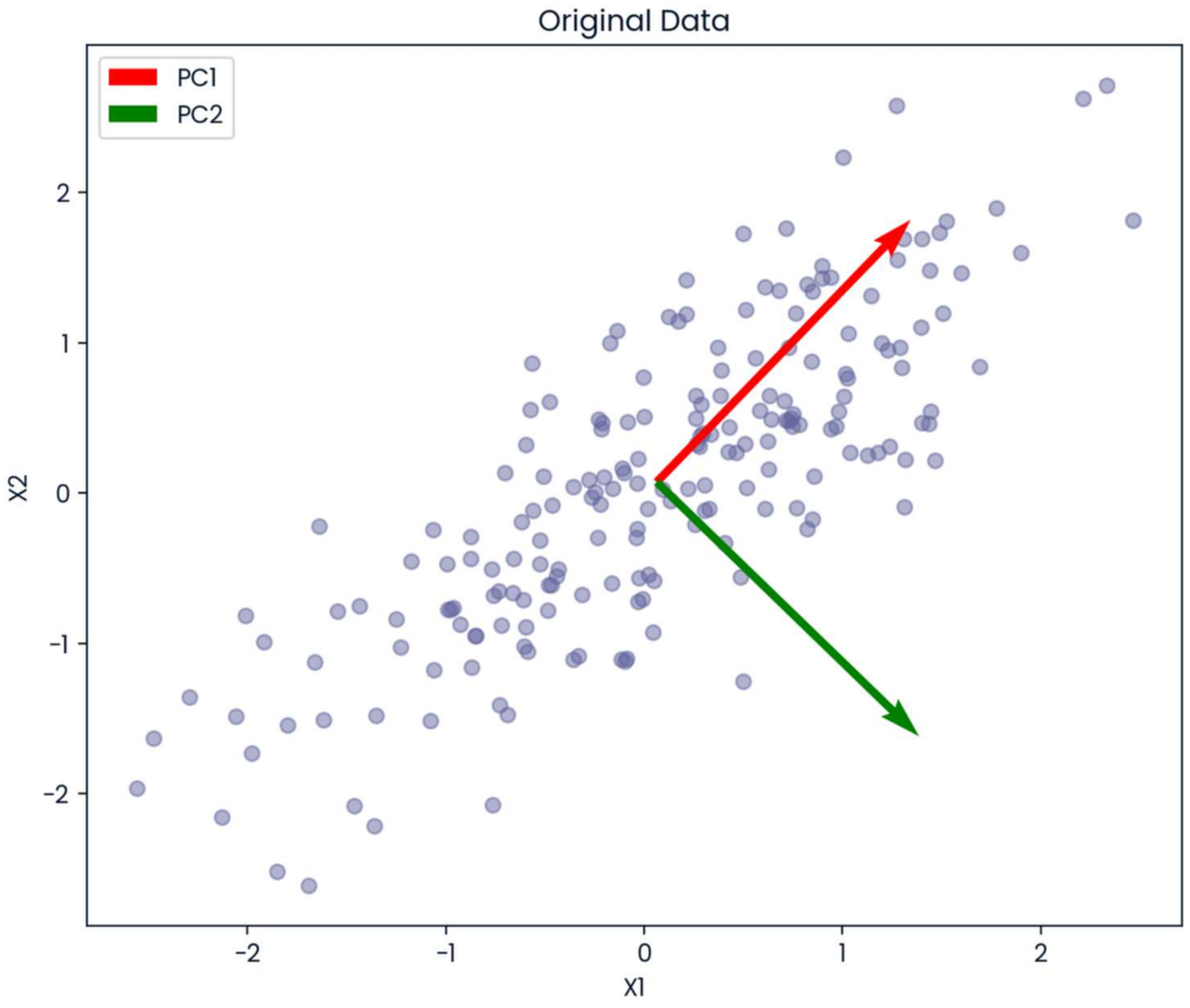
$$S = \frac{1}{n} \sum_{i=1}^{n} s(i)$$

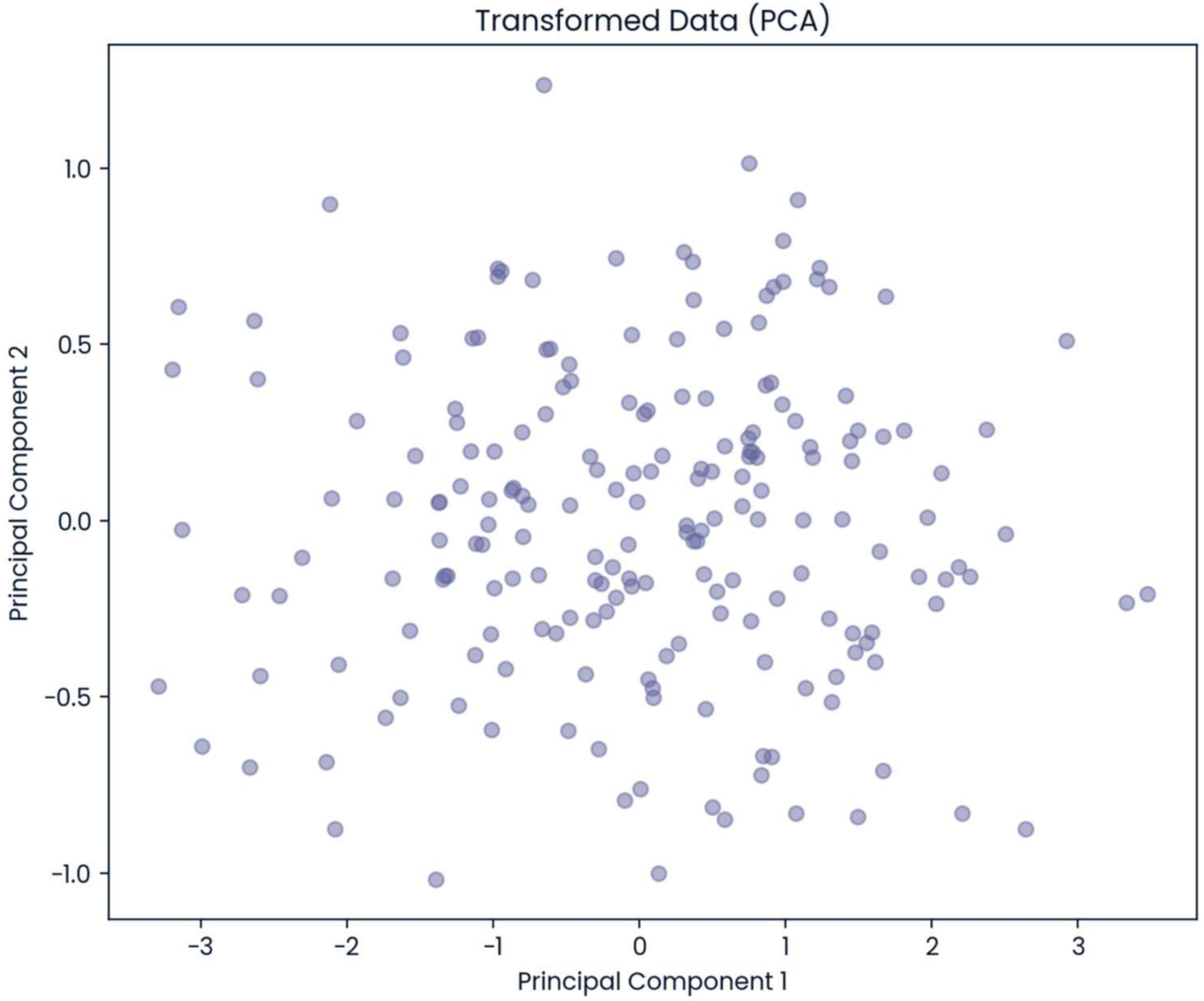
Where n is the number of data points.


Silhouette Score for different values of k
Silhouette Score
0.12
0.11
0.10
0.09
0.08
2
3
4
5
6
7
8
9
10
Number of clusters (k)

Principal Component Analysis

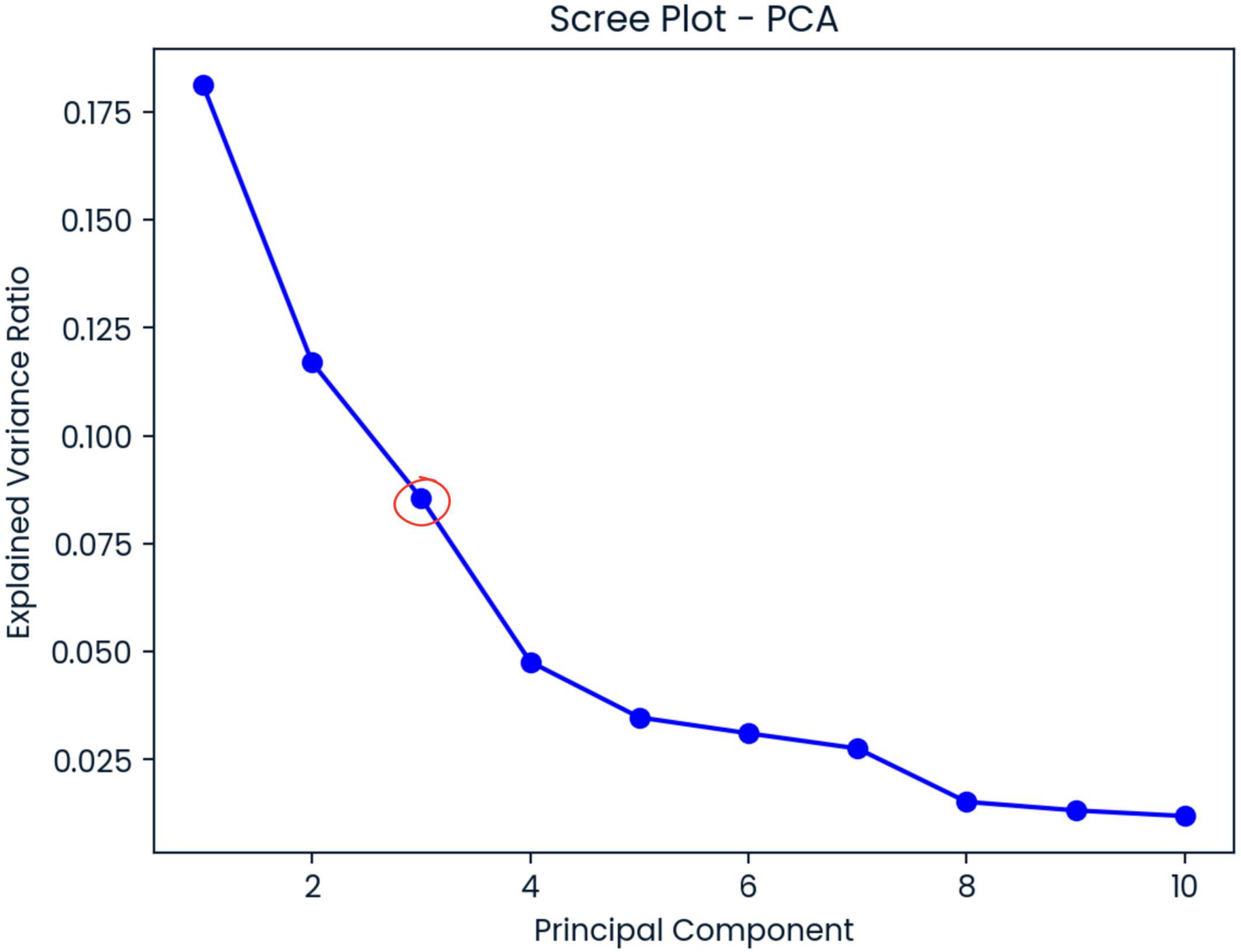
- PCA helps to simplify complex data by reducing the number of variables (or features) while keeping the important information.
- Imagine you have a dataset with many measurements (like height, weight, age, etc.). These measurements can be hard to visualize and analyze.
- PCA looks for patterns in the data and identifies the main factors that explain the most variation (differences) among the data points.
- It transforms the original measurements into new dimensions (called principal components) that combine the original data in a way that highlights these patterns.
- Instead of dealing with all the original measurements, PCA allows you to work with just a few new dimensions that capture most of the important information.



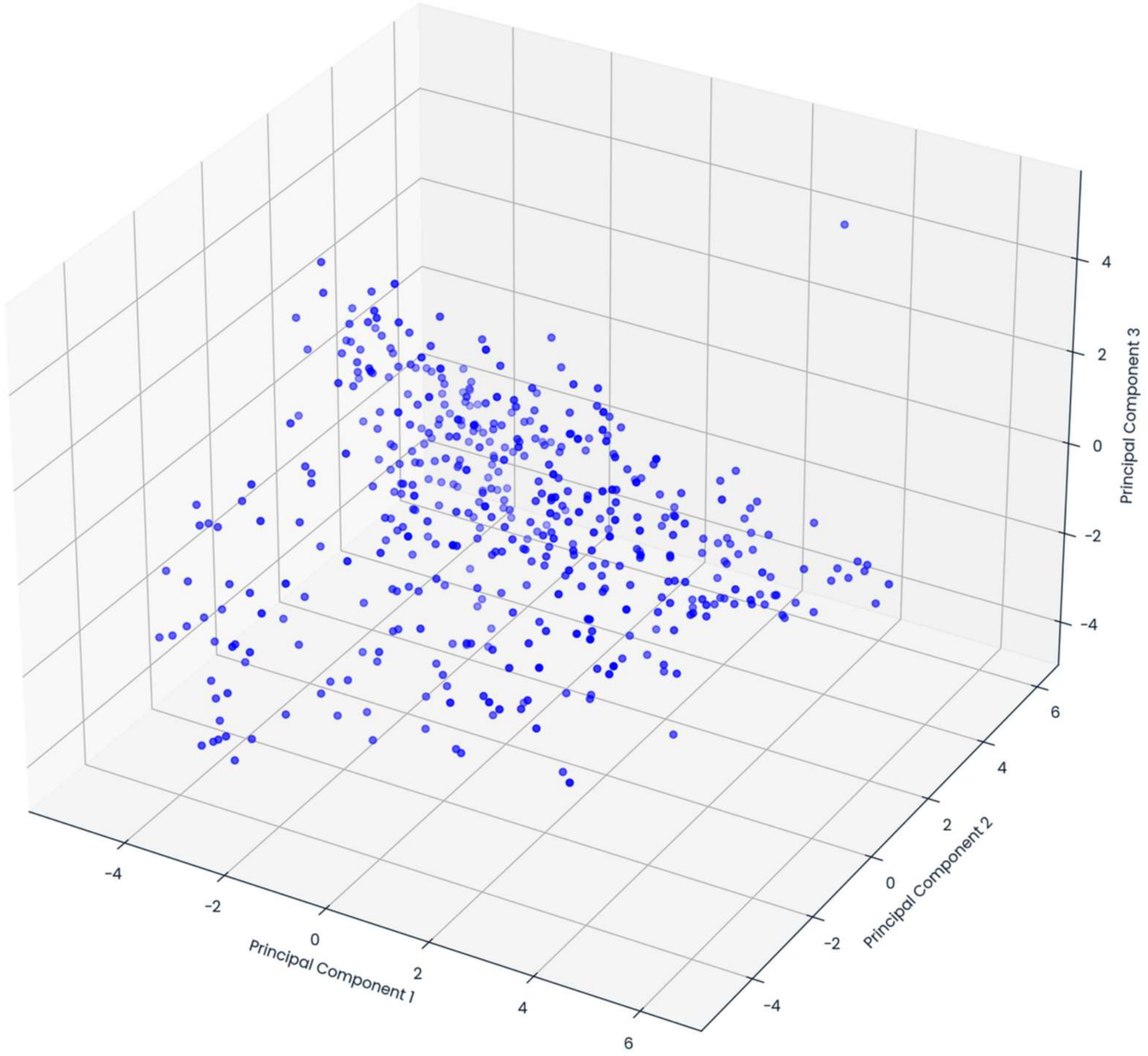


- PC1 (Red): Direction of maximum variance.
- PC2 (Green): Orthogonal to PC1, second-highest variance.

- Shows data in the new coordinate system, emphasizing variance.


Scree Plot - PCA
Explained Variance Ratio
0.175
0.150
0.125
0.100
0.075
0.050
0.025
2
4
6
8
10
Principal Component

3D PCA of the Dataset

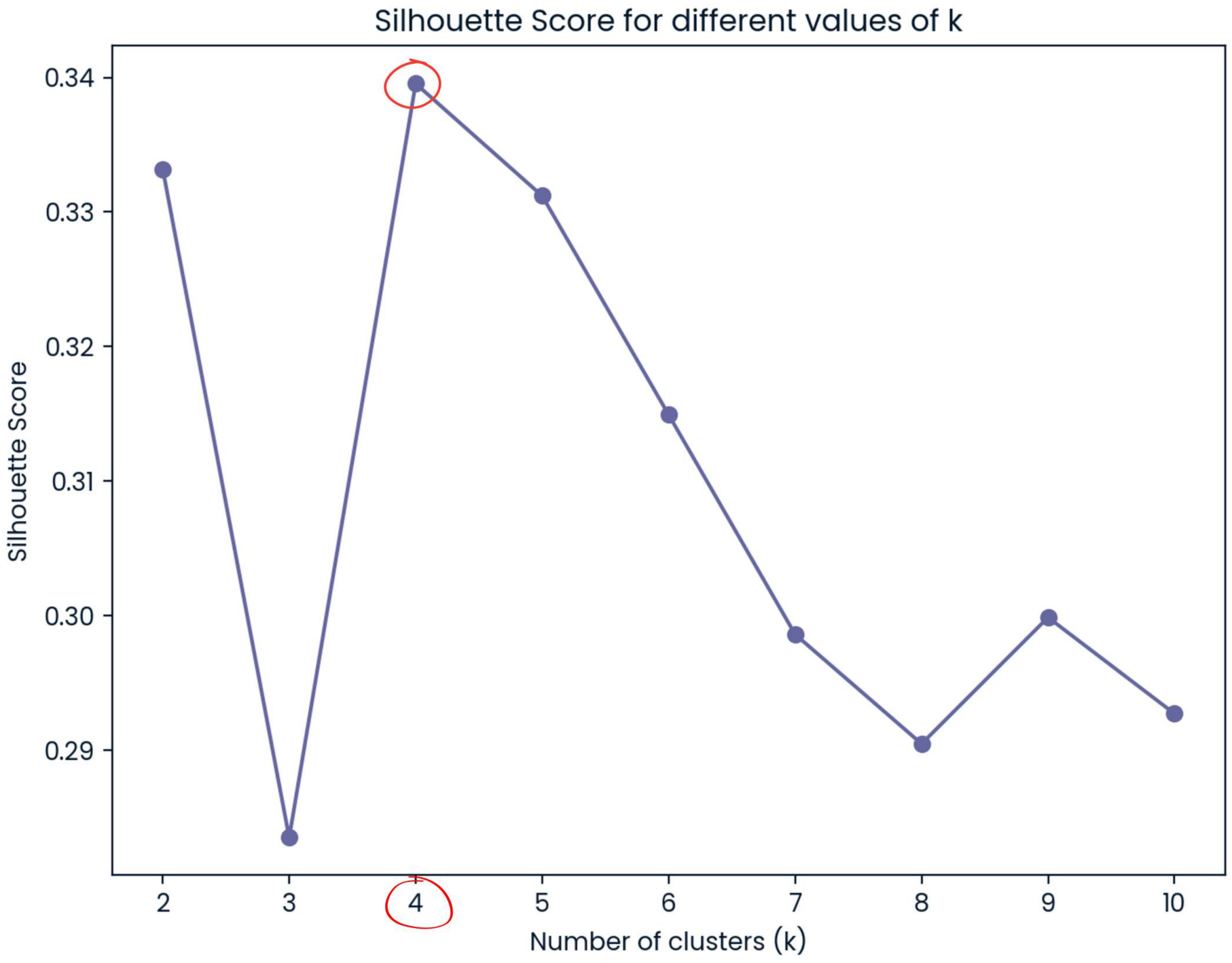



```
print(data_reduced_df.head())
```

```
        PC1       PC2       PC3
0  0.538819 -0.027434 -0.718416
1  0.834581 -1.115632  0.498874
2  0.518843 -0.938400  0.027769
3  3.560249 -0.137027  1.142511
4  1.057130 -0.241060  0.208300
```

```
print(data_reduced.shape)
```

```
(459, 3)
```


Silhouette Score for different values of k
Silhouette Score
0.34
0.33
0.32
0.31
0.30
0.29
2
3
4
5
6
7
8
9
10
Number of clusters (k)

M	P
array([[1, 1, 1, ..., 0, 0, 0], [1, 1, 1, ..., 0, 0, 0], [1, 1, 1, ..., 0, 0, 0], ..., [0, 0, 0, ..., 1, 1, 1], [0, 0, 0, ..., 1, 1, 1], [0, 0, 0, ..., 1, 1, 1]])	array([[1, 1, 1, ..., 1, 1, 1], [1, 1, 1, ..., 1, 1, 1], [1, 1, 1, ..., 1, 1, 1], ..., [1, 1, 1, ..., 1, 1, 1], [1, 1, 1, ..., 1, 1, 1], [1, 1, 1, ..., 1, 1, 1]])

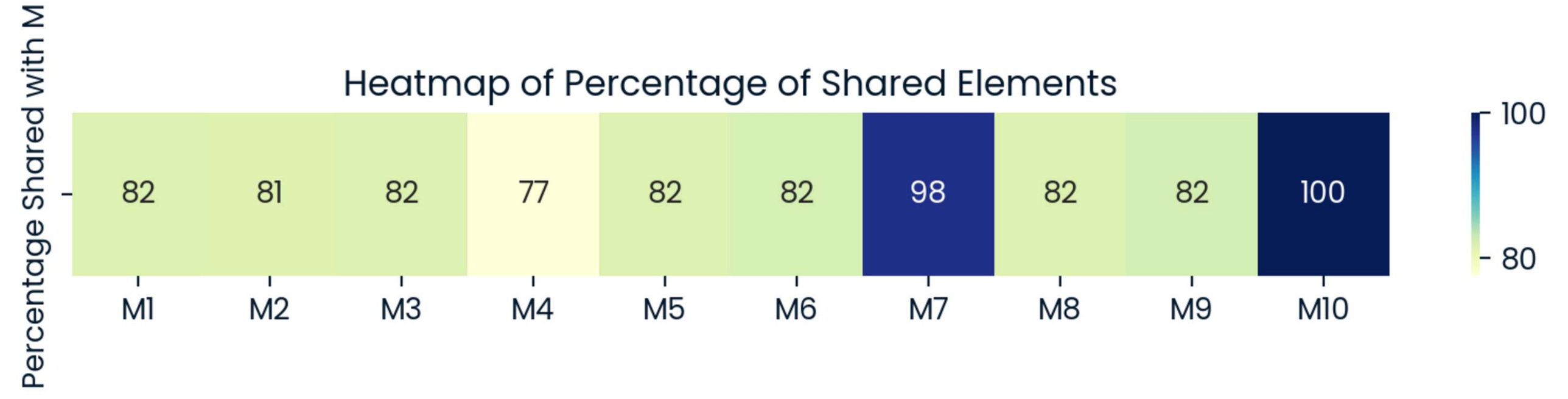
```
# Calculate the number of entries that are the same between M and P
same_entries = np.sum(M == P)
same_entries
```

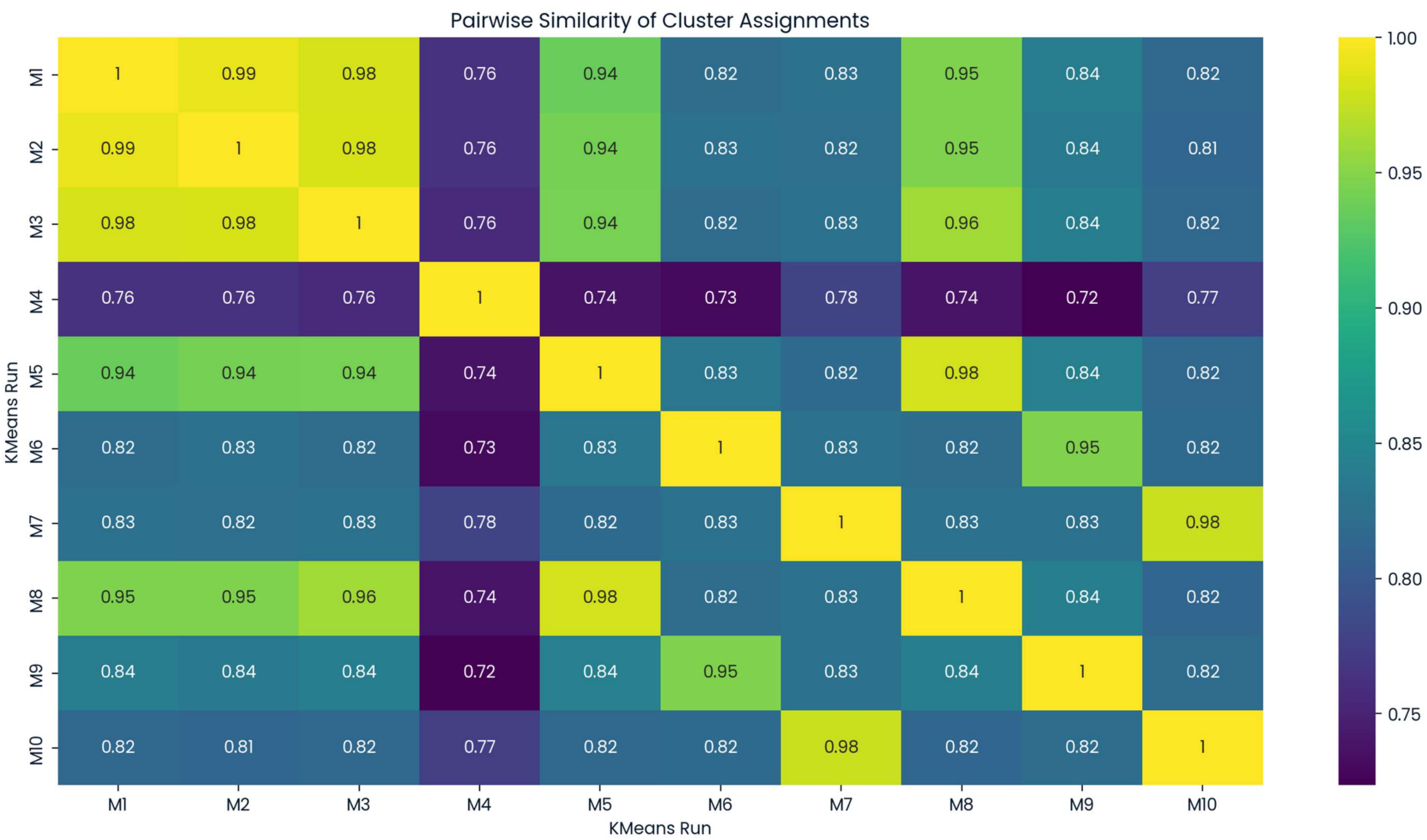
```
192787
```

- Matrix M is an n × n matrix where M[i, j] is 1 if the i-th and j-th observations belong to the same cluster based on the original data clustering, and 0 otherwise.
- Matrix P is an n × n matrix where P[i, j] is 1 if the i-th and j-th observations belong to the same cluster based on the PCA-reduced data clustering, and 0 otherwise.

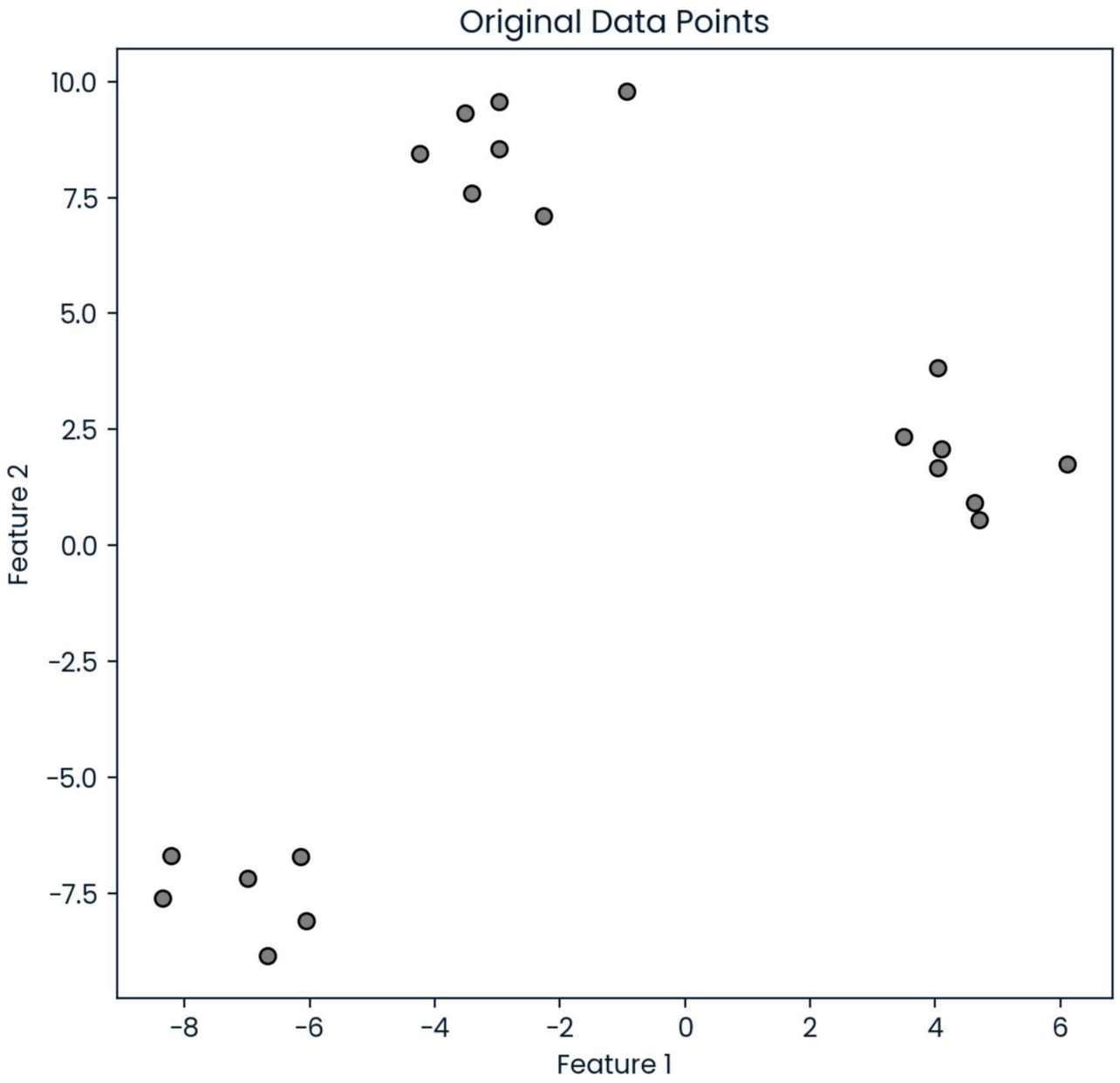
The value 192787 indicates the total number of positions (i, j) where both M and P have the same value (either both 1 or both 0). This can be interpreted as a measure of similarity between the clustering results obtained from the original data and the PCA-reduced data. A higher number suggests that the clustering results are more consistent between the two methods.

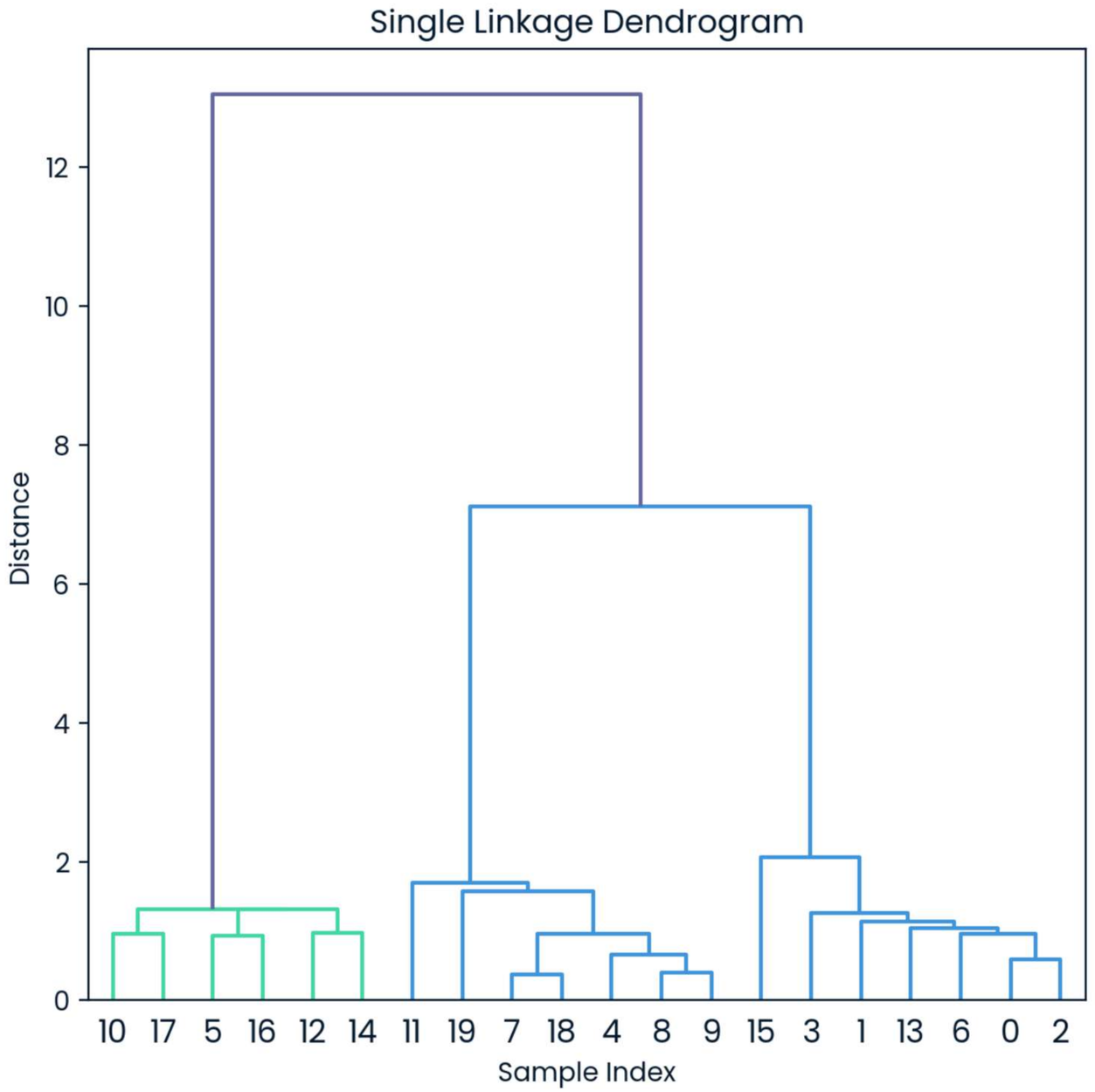
2. Perform k-means on the full-dimensional data ten times, using k clusters each time but ten different sets of initial values for cluster centroids. Generate matrices $M_1, \ldots, M_{10}$ indicating whether given pairs of observations are in the same cluster, as you did for M and P. Are the results of k-means on this dataset robust to the initial cluster centroid values chosen? Out of all of the pairs of observations that were recorded to be in the same cluster in at least one run, what percentage were in the same cluster in all ten? **Ans. 58.55%**




Pairwise Similarity of Cluster Assignments
KMeans Run
M1 1 0.99 0.98 0.76 0.94 0.82 0.83 0.95 0.84 0.82
M2 0.99 1 0.98 0.76 0.94 0.83 0.82 0.95 0.84 0.81
M3 0.98 0.98 1 0.76 0.94 0.82 0.83 0.96 0.84 0.82
M4 0.76 0.76 0.76 1 0.74 0.73 0.78 0.74 0.72 0.77
M5 0.94 0.94 0.94 0.74 1 0.83 0.82 0.98 0.84 0.82
M6 0.82 0.83 0.82 0.73 0.83 1 0.83 0.82 0.95 0.82
M7 0.83 0.82 0.83 0.78 0.82 0.83 1 0.83 0.83 0.98
M8 0.95 0.95 0.96 0.74 0.98 0.82 0.83 1 0.84 0.82
M9 0.84 0.84 0.84 0.72 0.84 0.95 0.83 0.84 1 0.82
M10 0.82 0.81 0.82 0.77 0.82 0.82 0.98 0.82 0.82 1
M1 M2 M3 M4 M5 M6 M7 M8 M9 M10
KMeans Run
1.00
0.95
0.90
0.85
0.80
0.75

Single Linkage Clustering





```
S_matrix
```

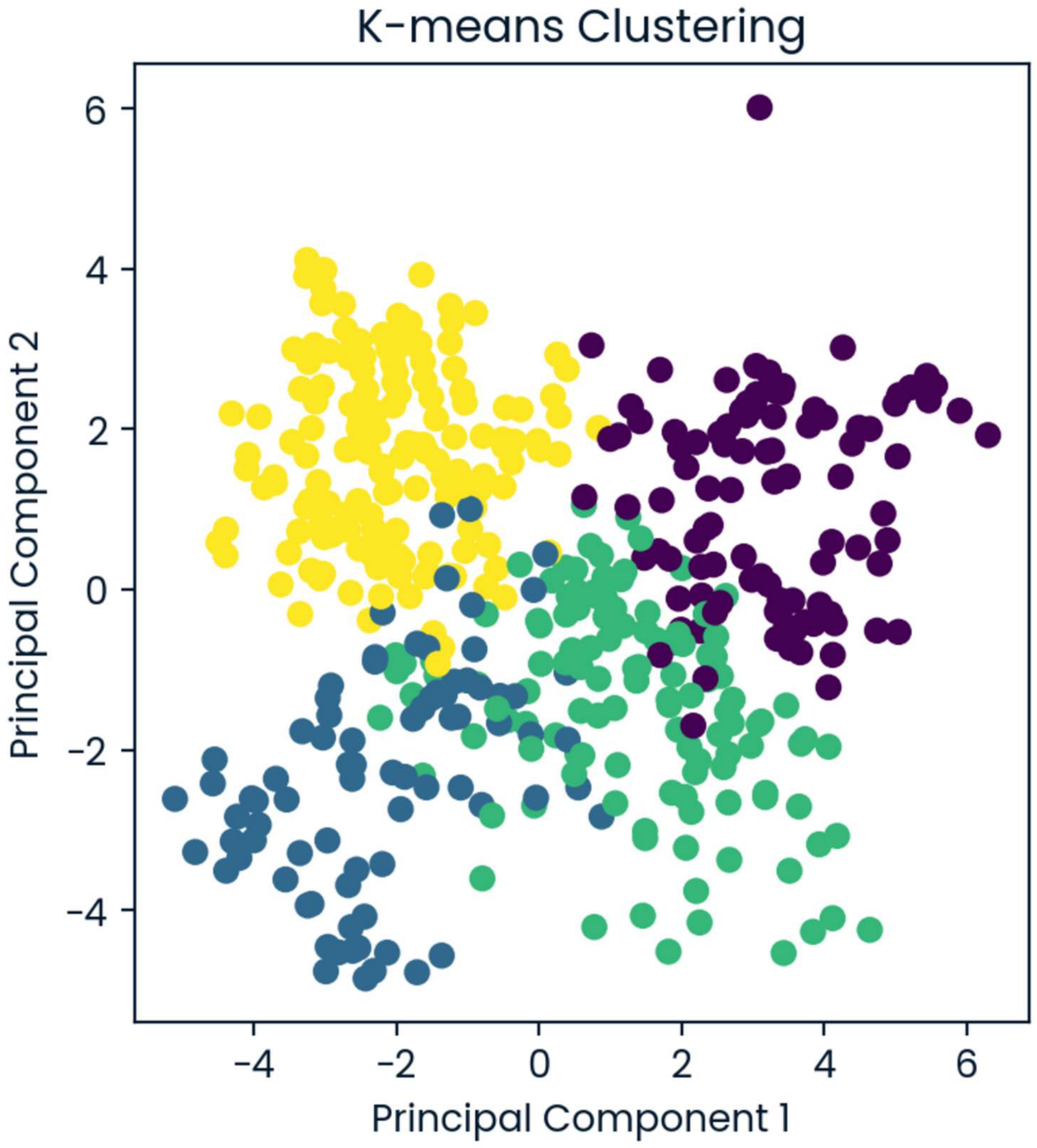
```
array([[1, 1, 1, ..., 1, 1, 1],
       [1, 1, 1, ..., 1, 1, 1],
       [1, 1, 1, ..., 1, 1, 1],
       ...,
       [1, 1, 1, ..., 1, 1, 1],
       [1, 1, 1, ..., 1, 1, 1],
       [1, 1, 1, ..., 1, 1, 1]])
```

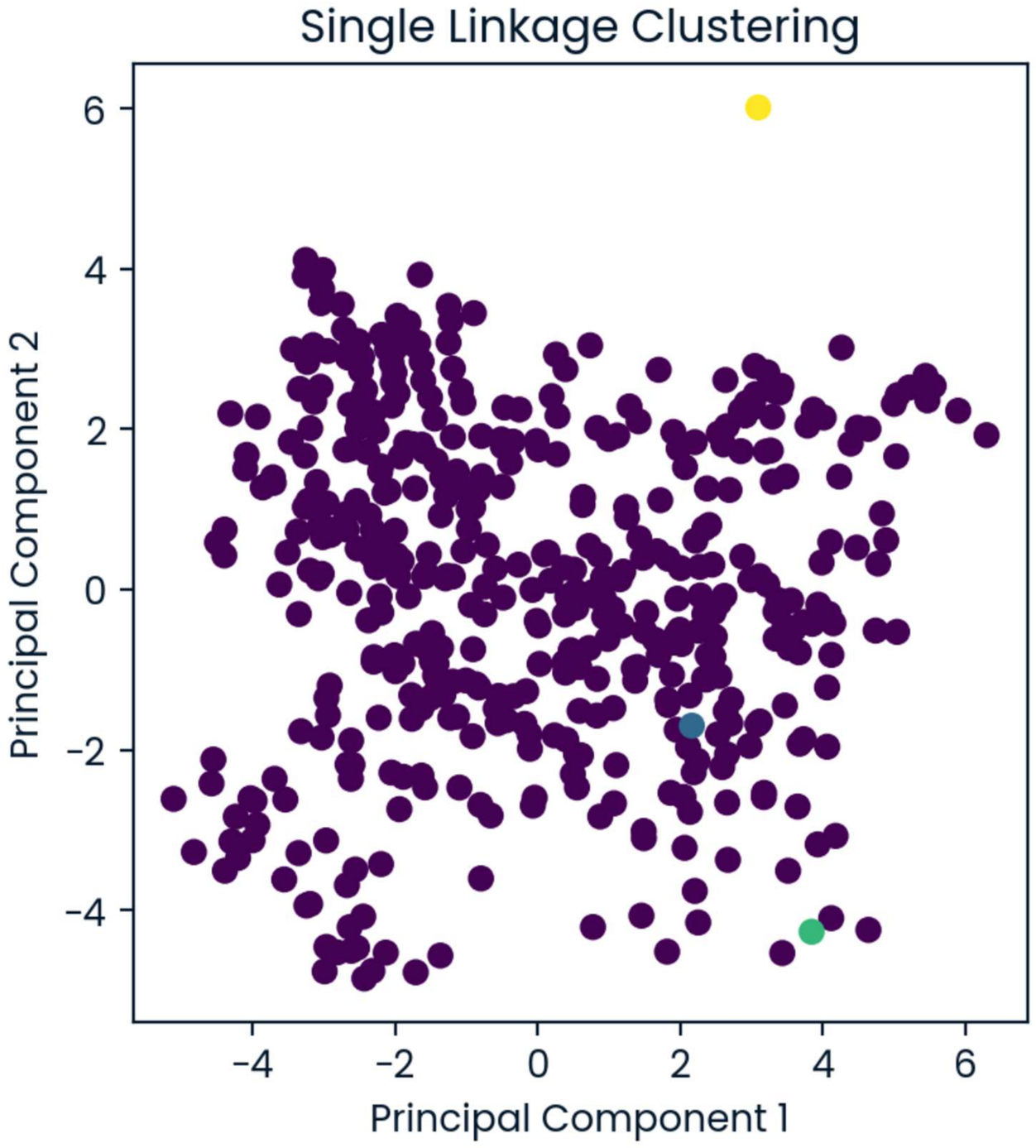
```
common_entries = np.sum(S_matrix == P)
common_entries
```

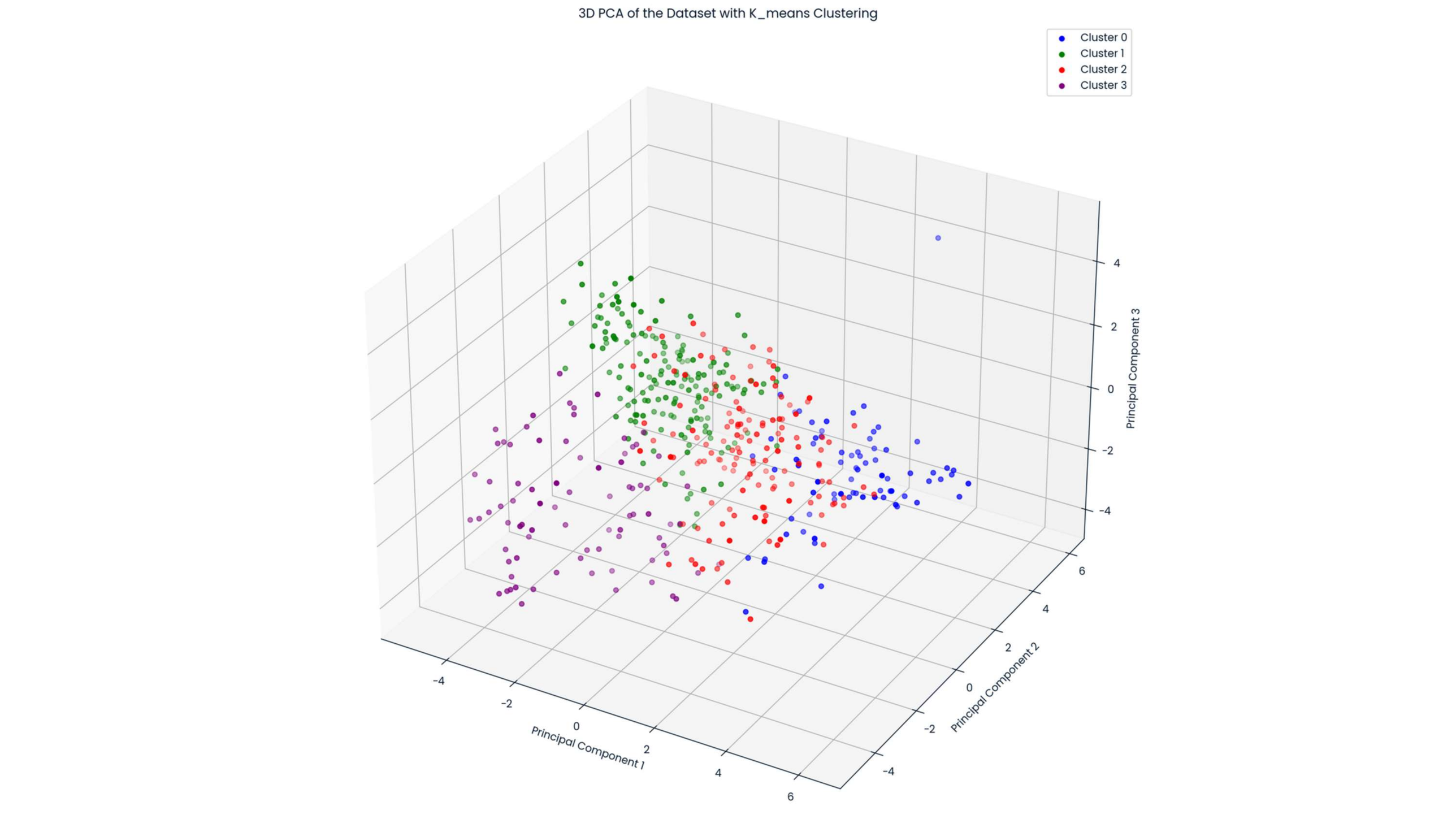
```
2435474
```

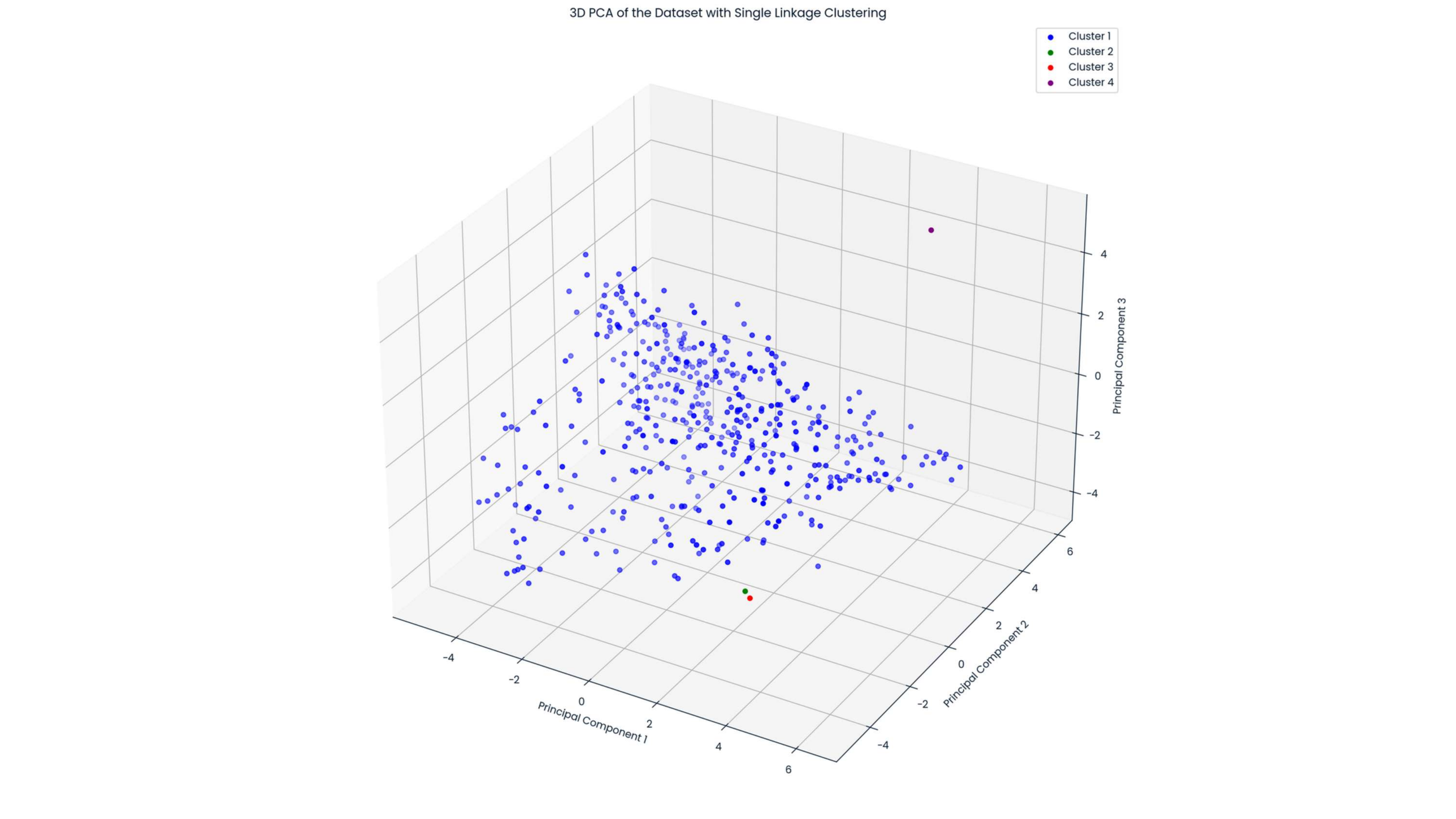
```
percentage_shared_between_S_and_P
```

```
50.00101624993918
```


K-means Clustering
Principal Component 2
Principal Component 1
6
4
2
0
−2
−4
−4
−2
0
2
4
6


Single Linkage Clustering
Principal Component 2
Principal Component 1
6
4
2
0
−2
−4
−4
−2
0
2
4
6


3D PCA of the Dataset with K_means Clustering
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Principal Component 1
Principal Component 2
Principal Component 3
-4
-2
0
2
4
6


3D PCA of the Dataset with Single Linkage Clustering
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Principal Component 1
Principal Component 2
Principal Component 3
-4
-2
0
2
4
6

4. In a Gaussian mixture model, each point in a dataset is assumed to have been drawn from one of many different Gaussian distributions. Hence, these distributions can be thought of as overlapping clusters. The expectation-maximization algorithm can be used to estimate the means and standard deviations of the distributions in a Gaussian mixture model (the centroids and sizes of the clusters, to put it a different way). Use the expectation-maximization algorithm with k clusters to probabilistically assign observations in the reduced-dimensional data to distributions/clusters. Create a new matrix E whose entries e_{ij} take values in $[0, 1]$: for observations i and j, e_{ij} is the probability that i and j are in the same cluster. What percentage of observations have at least two different clusters that they are assigned to with probability $> 1\%$?

```
print(E)
print(E.shape)
# E is now the matrix where E[i, j] is the probability that observations i and j are in the same cluster
```

```
[[9.80072507e-01 1.35645648e-02 2.13703203e-09 ... 1.00650519e-02
  1.00661729e-02 1.00650519e-02]
 [1.35645648e-02 9.92882698e-01 3.73593362e-11 ... 9.96428594e-01
  9.96427458e-01 9.96428594e-01]
 [2.13703203e-09 3.73593362e-11 9.99999231e-01 ... 2.97568826e-11
  2.97593180e-11 2.97568827e-11]
 ...
 [1.00650519e-02 9.96428594e-01 2.97568826e-11 ... 1.00000000e+00
  9.99998856e-01 1.00000000e+00]
 [1.00661729e-02 9.96427458e-01 2.97593180e-11 ... 9.99998856e-01
  9.99997712e-01 9.99998856e-01]
 [1.00650519e-02 9.96428594e-01 2.97568827e-11 ... 1.00000000e+00
  9.99998856e-01 1.00000000e+00]]
(2207, 2207)
```

```
# Threshold for the probability
threshold = 0.01

# Count the number of observations that meet the criteria
count = 0
for prob in probabilities:
    if np.sum(prob > threshold) >= 2:
        count += 1

# Calculate the percentage
percentage = (count / n_samples) * 100

percentage
```

The percentage of observations assigned to at least two different clusters with a probability exceeding 1% is **25.51%**.

t-SNE starts by understanding the data's **similarity structure**:

- It computes the **pairwise similarity** between every pair of points in the high-dimensional space.
- Similarity is modeled using a **Gaussian distribution** centered around each data point i. The probability that a point j is a neighbor of i is given by:

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$$

  where:

  - x_i and x_j are data points.
  - σ_i is a parameter that adjusts the neighborhood size for i to match a predefined **perplexity** (a user-defined parameter).

The joint probability for pairs is symmetrized:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

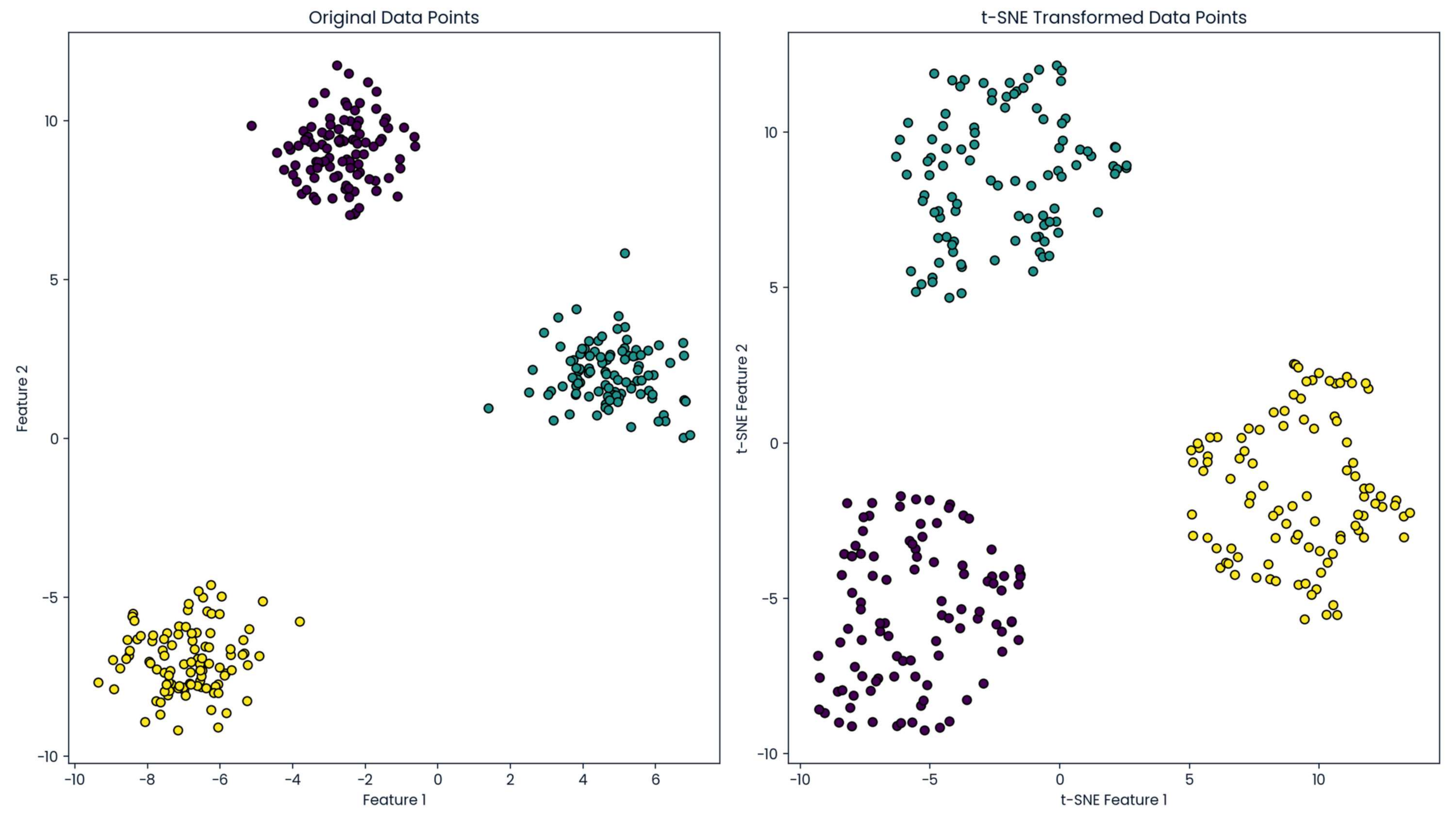
t-SNE minimizes the **Kullback-Leibler (KL) divergence** between the high-dimensional similarities p_{ij} and the low-dimensional ones q_{ij}:

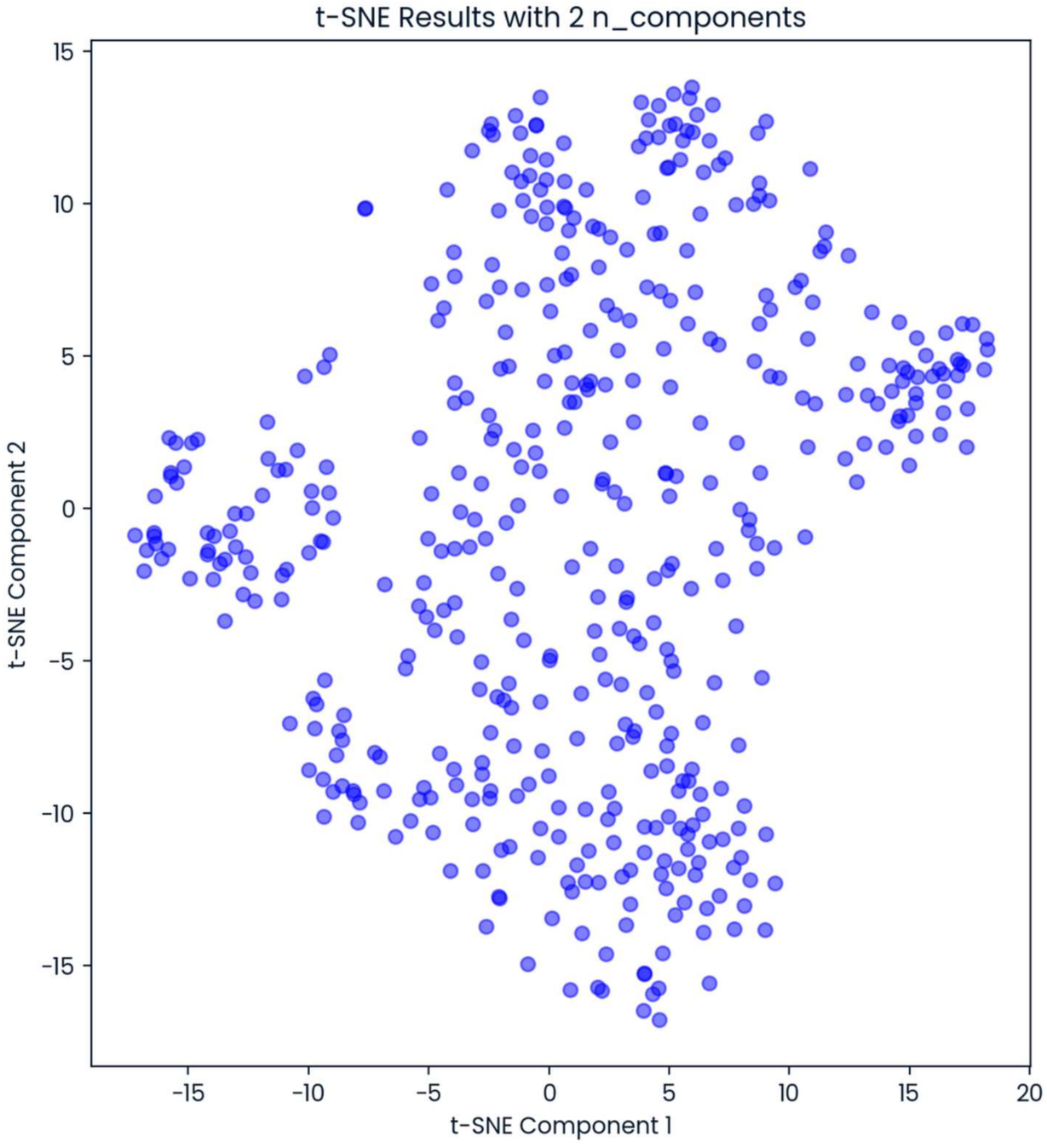
$$\mathrm{KL}(P\|Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

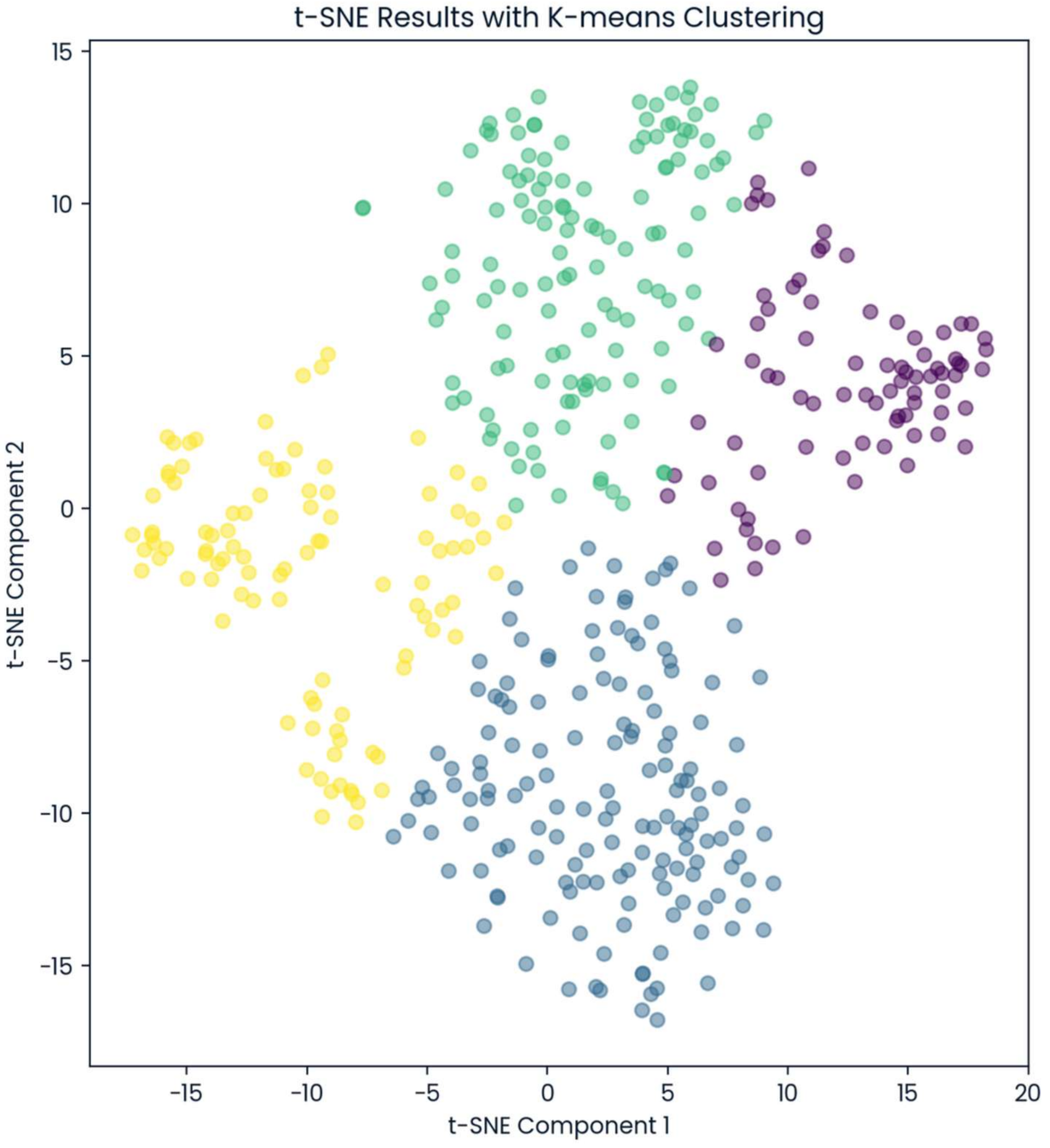
This ensures the low-dimensional representation reflects the local similarity structure of the original data.

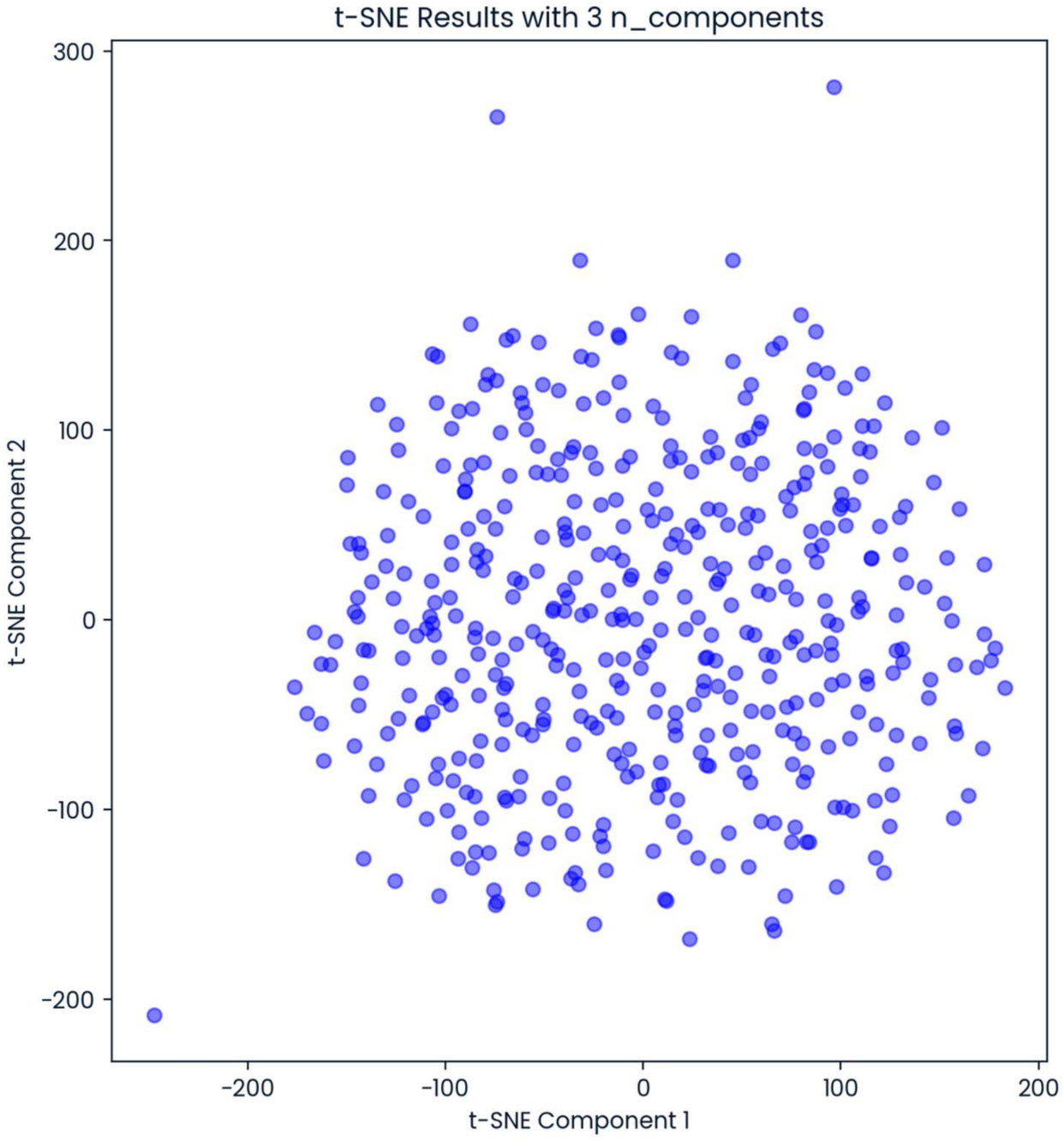
The algorithm adjusts the positions of points in the low-dimensional space (y_i) using **gradient descent** to minimize the KL divergence. Intuitively:

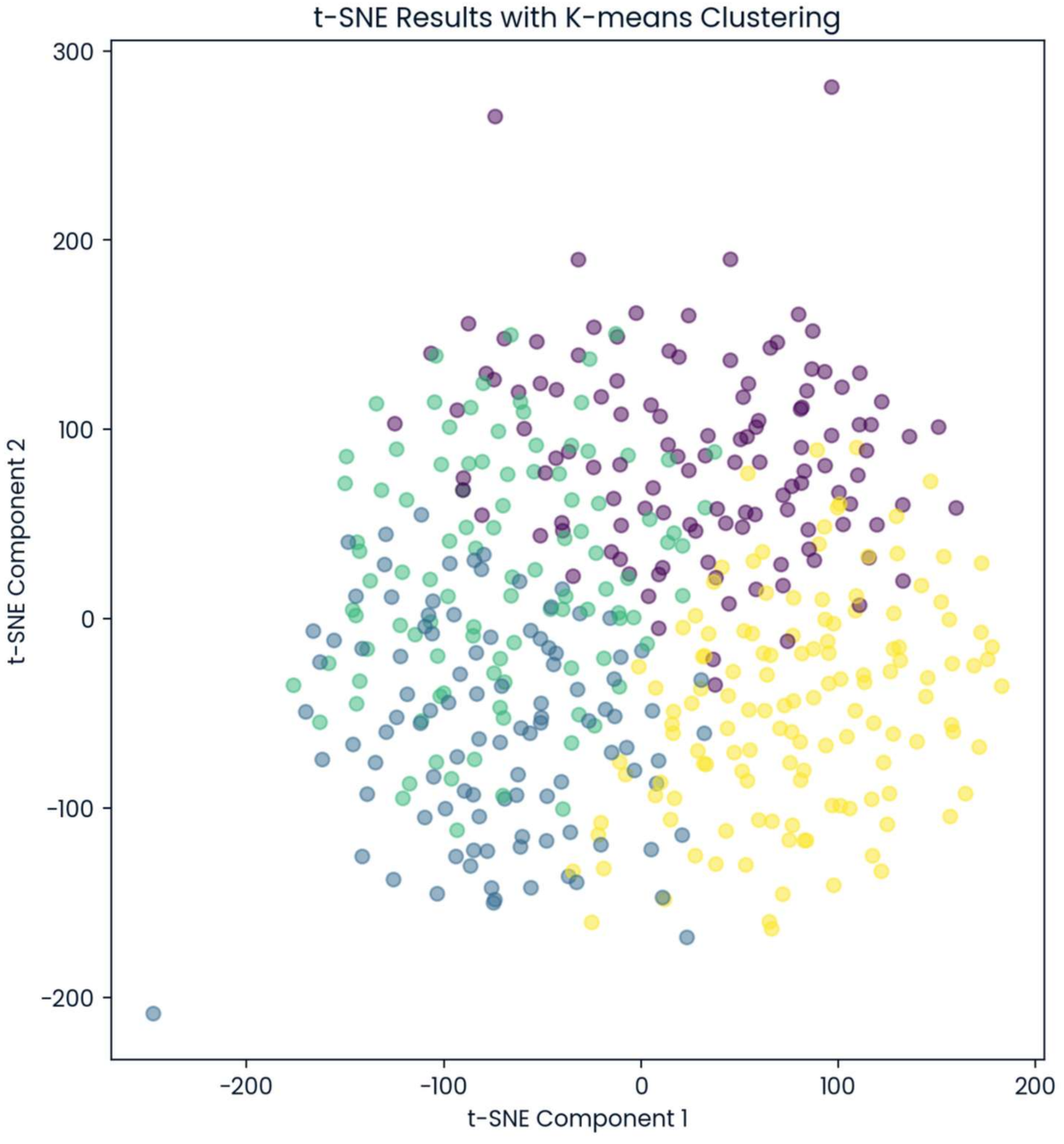
- Points that are similar (high p_{ij}) are pulled together.
- Points that are dissimilar (low p_{ij}) are pushed apart.

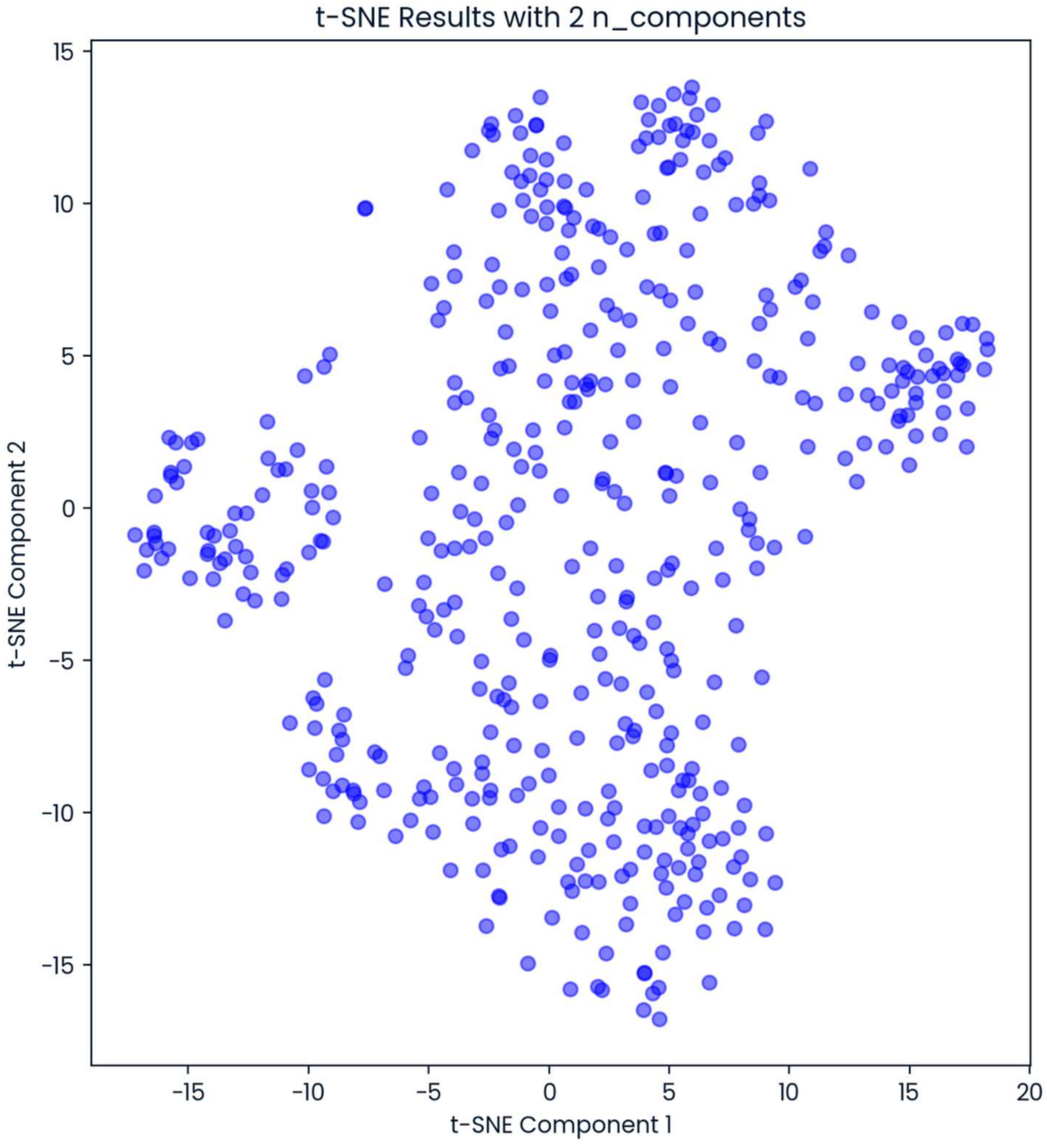

Original Data Points
Feature 2
Feature 1
t-SNE Transformed Data Points
t-SNE Feature 2
t-SNE Feature 1

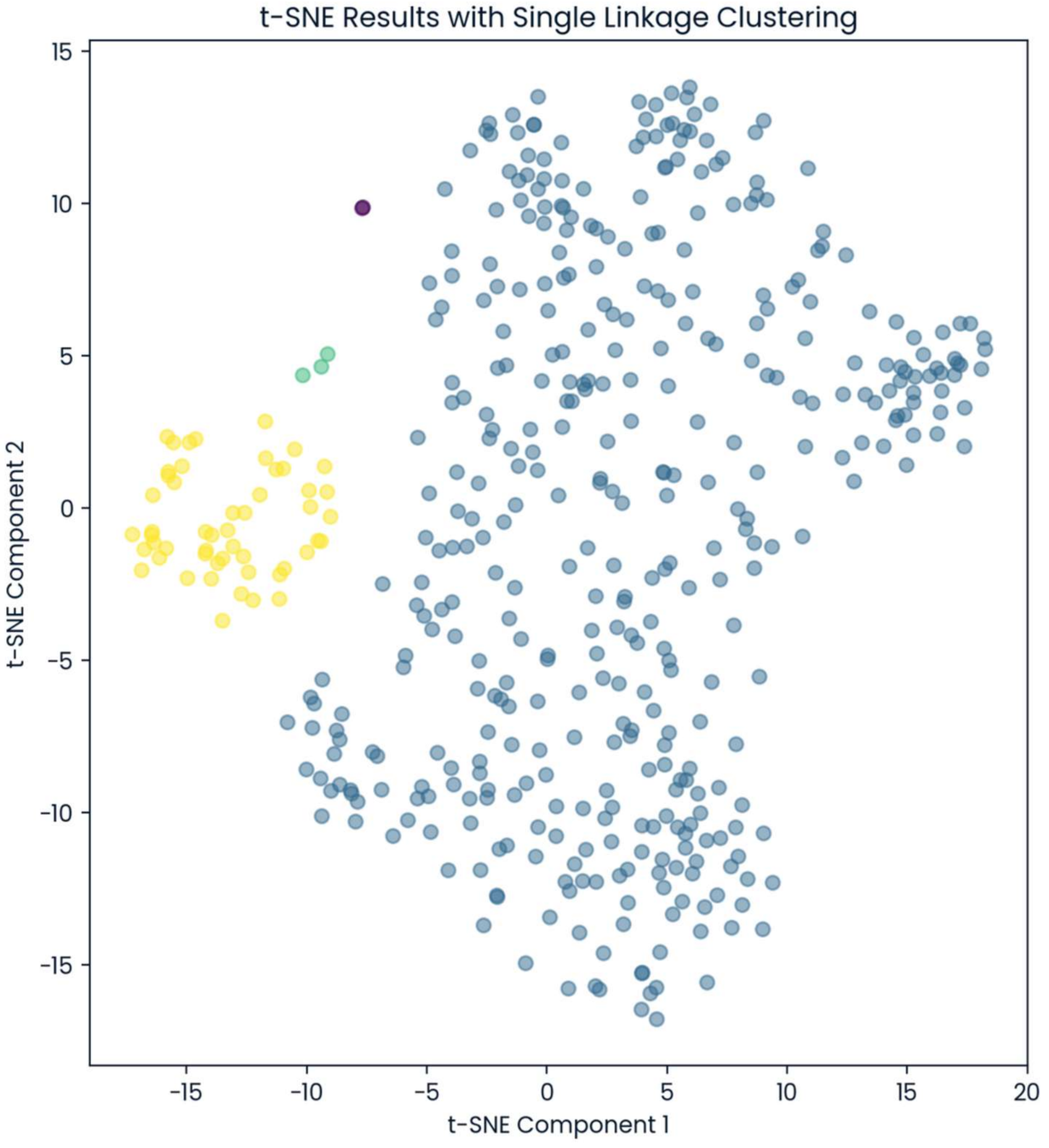

t-SNE Results with 2 n_components
t-SNE Component 2
t-SNE Component 1
15
10
5
0
-5
-10
-15
-15
-10
-5
0
5
10
15
20

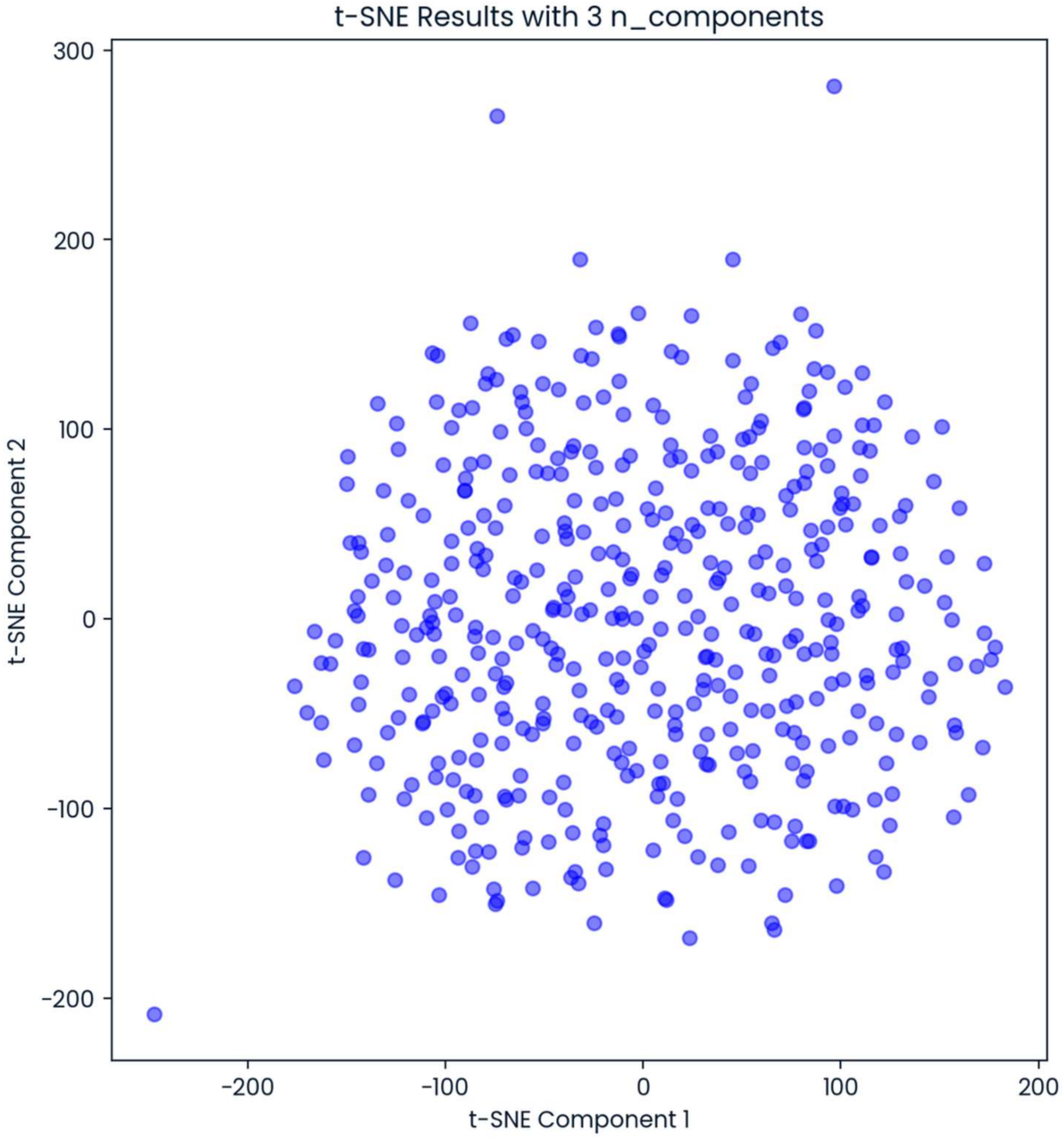

t-SNE Results with K-means Clustering
t-SNE Component 2
t-SNE Component 1
15
10
5
0
-5
-10
-15
-15
-10
-5
0
5
10
15
20

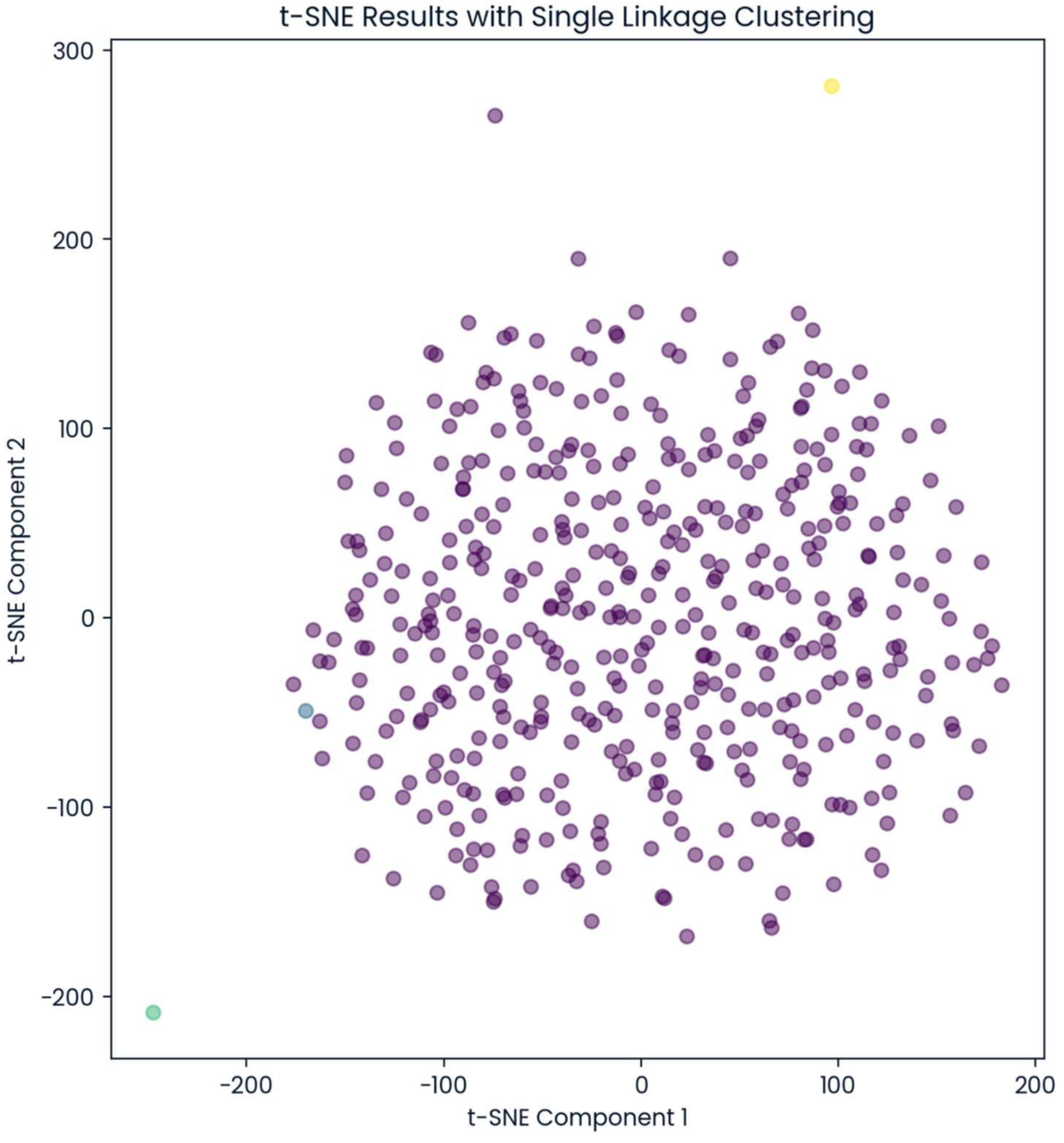

t-SNE Results with 3 n_components
300
200
100
0
-100
-200
t-SNE Component 2
-200
-100
0
100
200
t-SNE Component 1

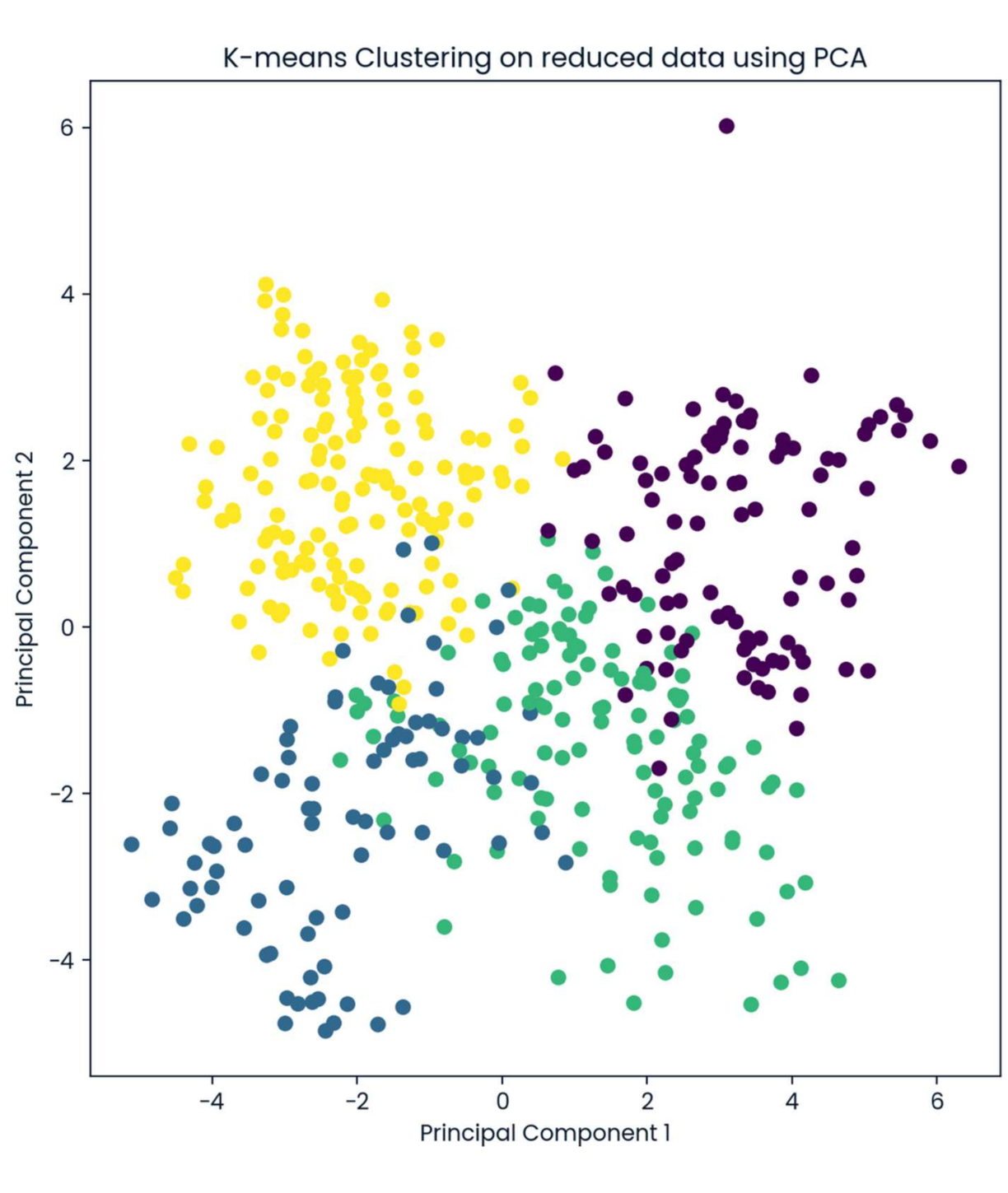

t-SNE Results with K-means Clustering
300
200
100
0
-100
-200
t-SNE Component 2
-200
-100
0
100
200
t-SNE Component 1

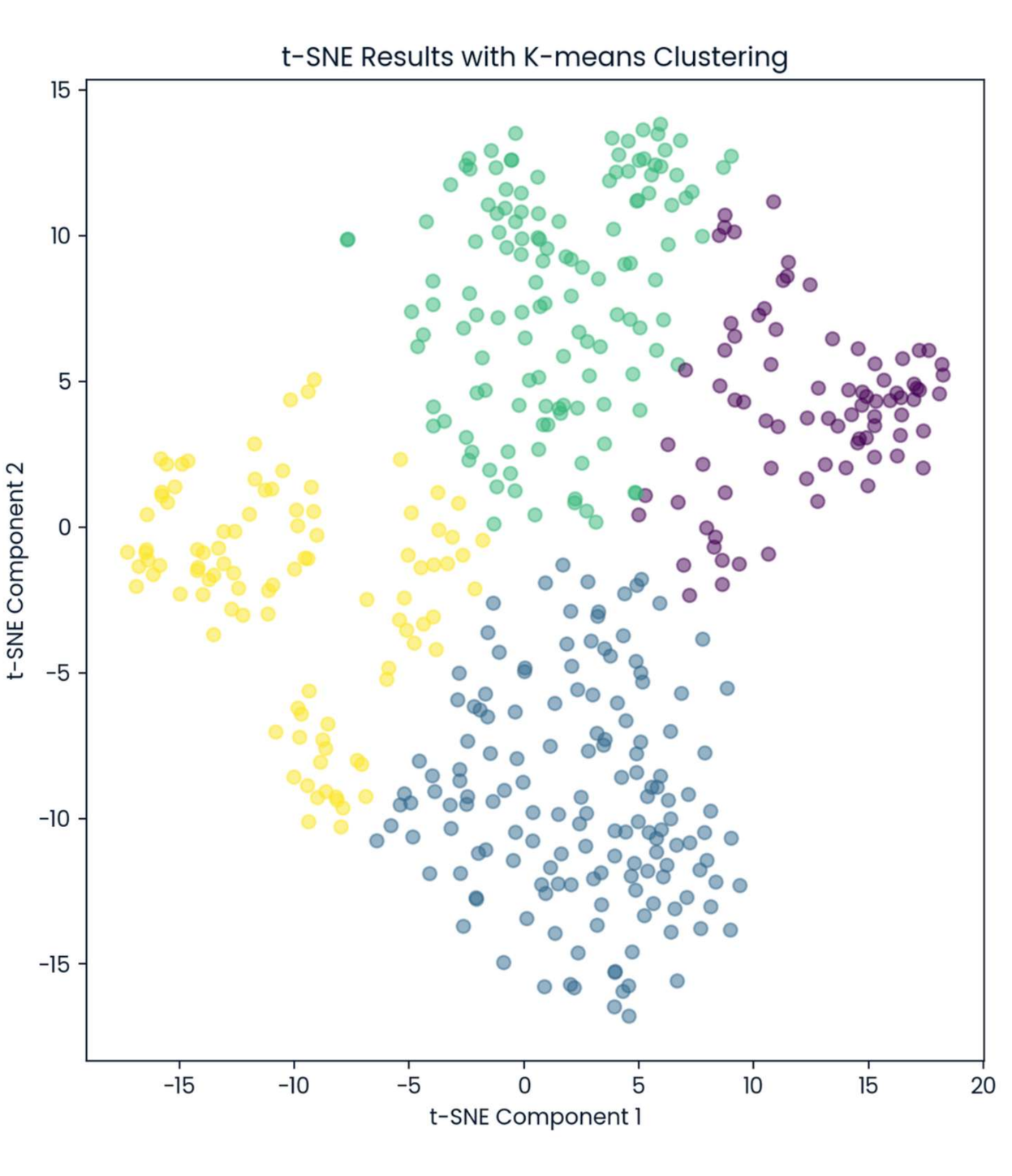

t-SNE Results with 2 n_components
t-SNE Component 2
t-SNE Component 1
15
10
5
0
-5
-10
-15
-15
-10
-5
0
5
10
15
20

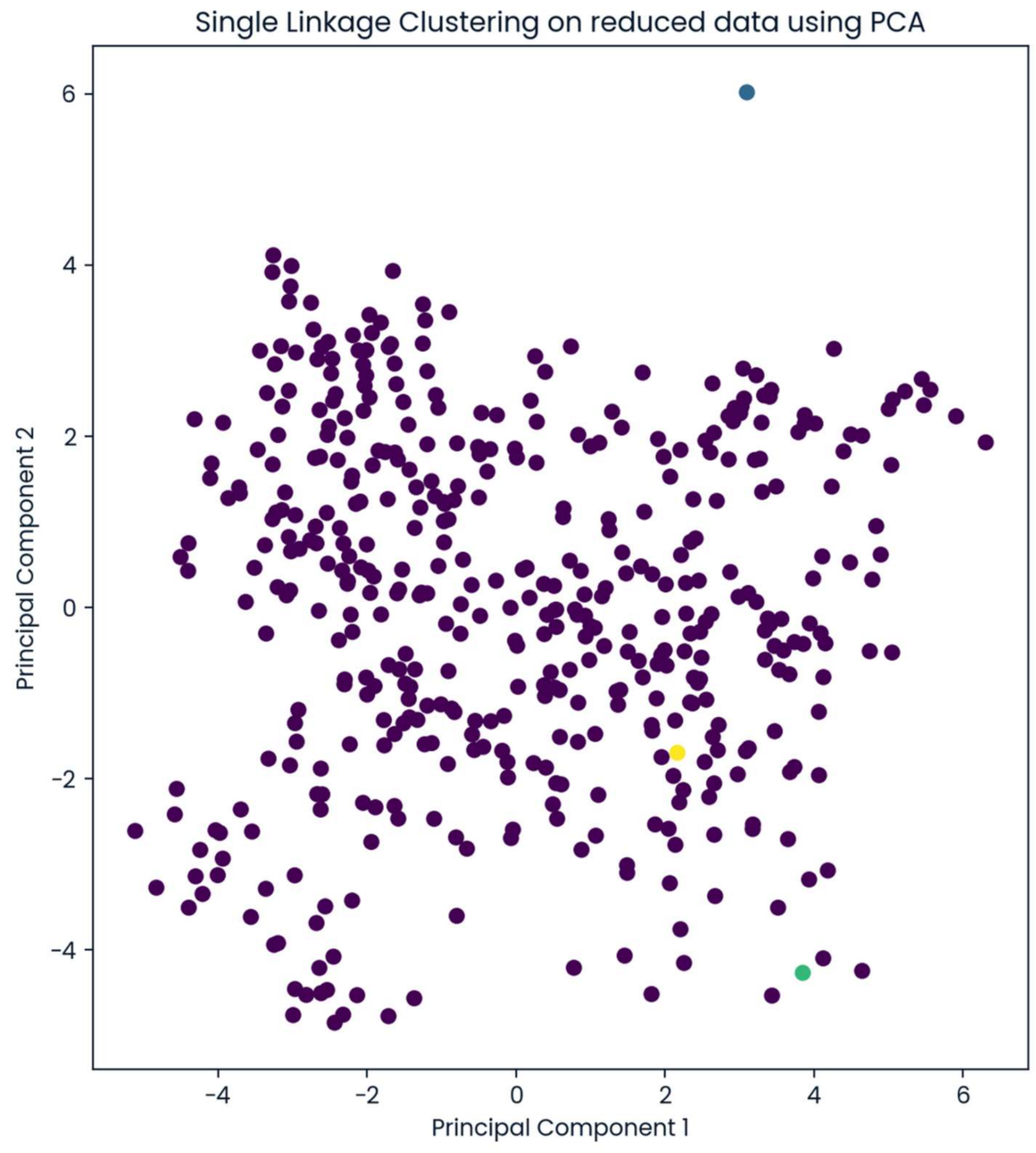

t-SNE Results with Single Linkage Clustering
t-SNE Component 2
t-SNE Component 1
15
10
5
0
-5
-10
-15
-15
-10
-5
0
5
10
15
20

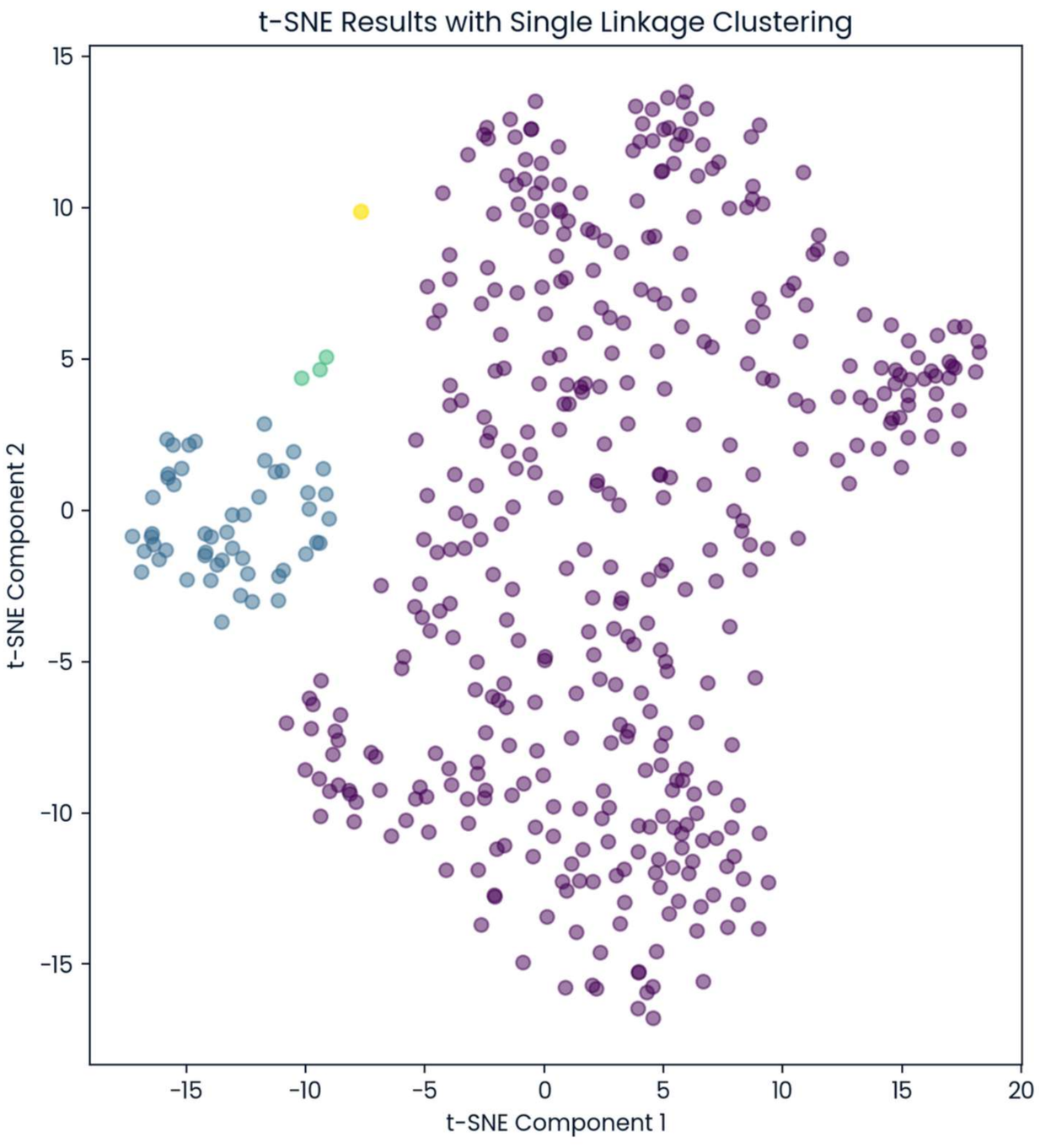

t-SNE Results with 3 n_components
300
200
100
0
-100
-200
t-SNE Component 2
-200
-100
0
100
200
t-SNE Component 1

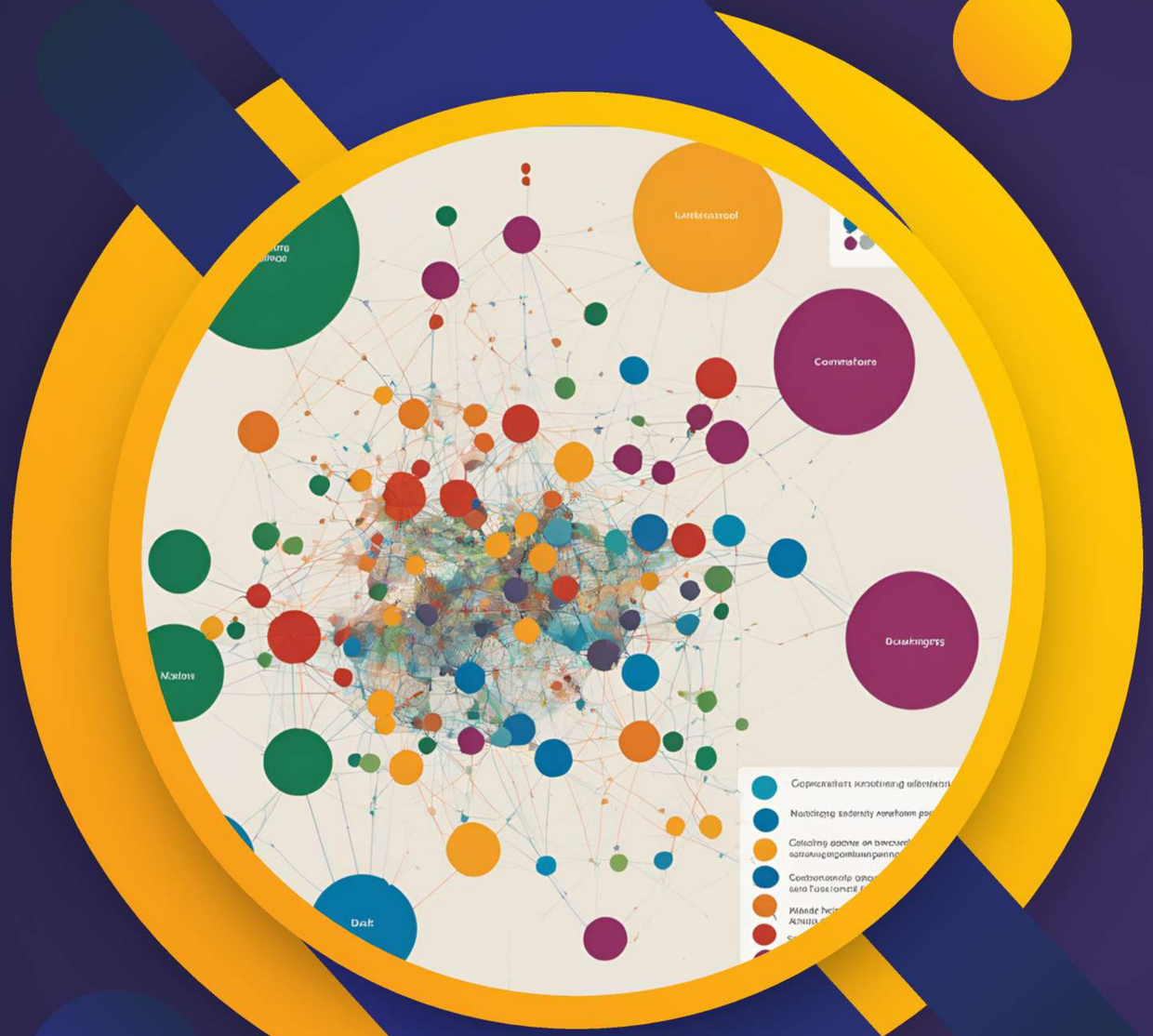

t-SNE Results with Single Linkage Clustering
300
200
100
0
-100
-200
t-SNE Component 2
-200
-100
0
100
200
t-SNE Component 1


K-means Clustering on reduced data using PCA
Principal Component 2
Principal Component 1
6
4
2
0
-2
-4
-4
-2
0
2
4
6


t-SNE Results with K-means Clustering
t-SNE Component 2
t-SNE Component 1
15
10
5
0
-5
-10
-15
-15
-10
-5
0
5
10
15
20


Single Linkage Clustering on reduced data using PCA
Principal Component 2
Principal Component 1
6
4
2
0
-2
-4
-4
-2
0
2
4
6


t-SNE Results with Single Linkage Clustering
t-SNE Component 2
t-SNE Component 1
15
10
5
0
-5
-10
-15
-15
-10
-5
0
5
10
15
20



UNIVERSITY OF ALBERTA

Thank You

For Your Attention

Aashish Kumar