EXP NO: 5	CLUSTERING WITH K-MEANS AND DIMENSIONALITY REDUCTION WITH PCA
DATE: 28/08/2025	

AIM:
To demonstrate the application of Unsupervised Learning models, specifically K-Means clustering for grouping data points and Principal Component Analysis (PCA) for dimensionality reduction and visualization, using a suitable dataset.

ALGORITHM:

1. K-MEANS CLUSTERING

K-Means is an iterative clustering algorithm that aims to partition n observations into k clusters, where each observation belongs to the cluster with the nearest mean (centroid).

STEPS:
1. **INITIALIZATION:** Choose k initial centroids randomly from the dataset.
2. **ASSIGNMENT:** Assign each data point to the cluster whose centroid is closest (e.g., using Euclidean distance).
3. **UPDATE:** Recalculate the centroids as the mean of all data points assigned to that cluster.
4. **ITERATION:** Repeat steps 2 and 3 until the centroids no longer move significantly or a maximum number of iterations is reached.

2. PRINCIPAL COMPONENT ANALYSIS (PCA)

PCA is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.

STEPS:
1. **STANDARDIZATION:** Standardize the dataset (mean = 0, variance = 1).
2. **COVARIANCE MATRIX CALCULATION:** Compute the covariance matrix of the standardized data.
3. **EIGENVALUE DECOMPOSITION:** Calculate the eigenvalues and eigenvectors of the covariance matrix.
4. **FEATURE VECTOR CREATION:** Sort the eigenvectors by decreasing eigenvalues and select the top k eigenvectors to form a feature vector (projection matrix).
5. **PROJECTION:** Project the original data onto the new feature space using the feature vector.

CODE:

```
# ================================
# EXPERIMENT – K-Means & PCA
# ================================

# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.datasets import make_blobs
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score

# --- Part 1: K-Means Clustering ---

print("--- Part 1: K-Means Clustering ---")

# 1. Generate dataset
X, y = make_blobs(n_samples=300, centers=3, cluster_std=0.60, random_state=42)
df_kmeans = pd.DataFrame(X, columns=['Feature_1', 'Feature_2'])
print("\nOriginal K-Means Dataset Head:")
print(df_kmeans.head())

# 2. Elbow Method
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10,
random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), wcss, marker='o', linestyle='--')
plt.title('Elbow Method for Optimal K (K-Means)')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('WCSS')
plt.grid(True)
plt.show()
```

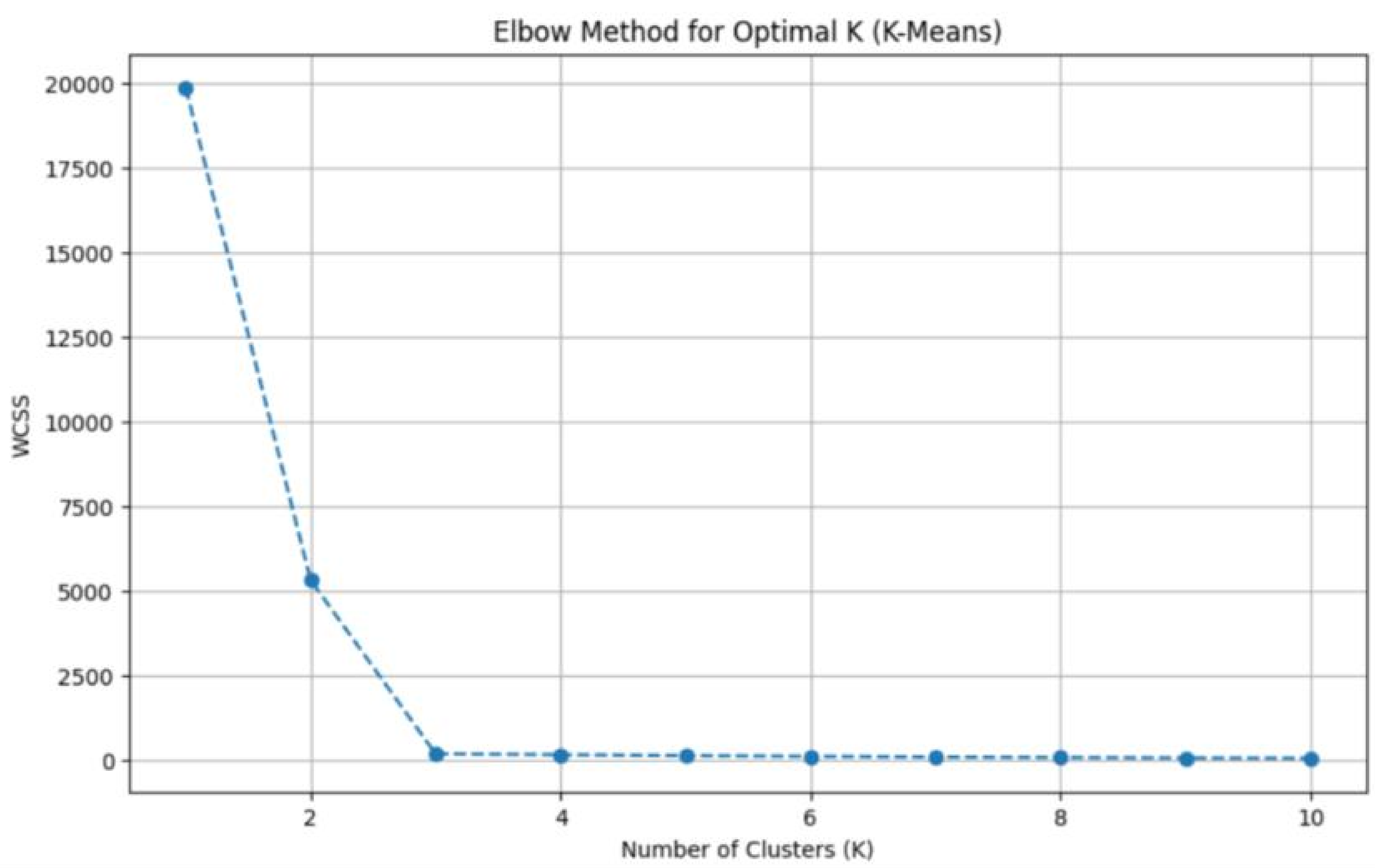
```
# 3. Apply K-Means with chosen K
optimal_k = 3
kmeans = KMeans(n_clusters=optimal_k, init='k-means++', max_iter=300,
n_init=10, random_state=42)
clusters = kmeans.fit_predict(X)
df_kmeans['Cluster'] = clusters

# 4. Visualize K-Means clusters
plt.figure(figsize=(10, 8))
sns.scatterplot(x='Feature_1', y='Feature_2', hue='Cluster', data=df_kmeans,
palette='viridis', s=100, alpha=0.8)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
s=300, c='red', marker='X', label='Centroids')
plt.title(f'K-Means Clustering with K={optimal_k}')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
plt.grid(True)
plt.show()

# 5. Silhouette Score
silhouette_avg = silhouette_score(X, clusters)
print(f"\nSilhouette Score for K-Means (K={optimal_k}): {silhouette_avg:.3f}")

# --- Part 2: Dimensionality Reduction with PCA ---

print("\n--- Part 2: Dimensionality Reduction with PCA ---")

# 1. Generate 4D dataset
X_pca, y_pca = make_blobs(n_samples=500, n_features=4, centers=4,
cluster_std=1.0, random_state=25)
df_pca_original = pd.DataFrame(X_pca, columns=[f'Feature_{i+1}' for i in
range(X_pca.shape[1])])
df_pca_original['True_Cluster'] = y_pca
print("\nOriginal PCA Dataset Head:")
print(df_pca_original.head())
print(f"Original PCA Dataset Shape: {df_pca_original.shape}")

# 2. Standardize
scaler = StandardScaler()
X_pca_scaled = scaler.fit_transform(X_pca)

# 3. PCA (4D → 2D)
pca = PCA(n_components=2)
```

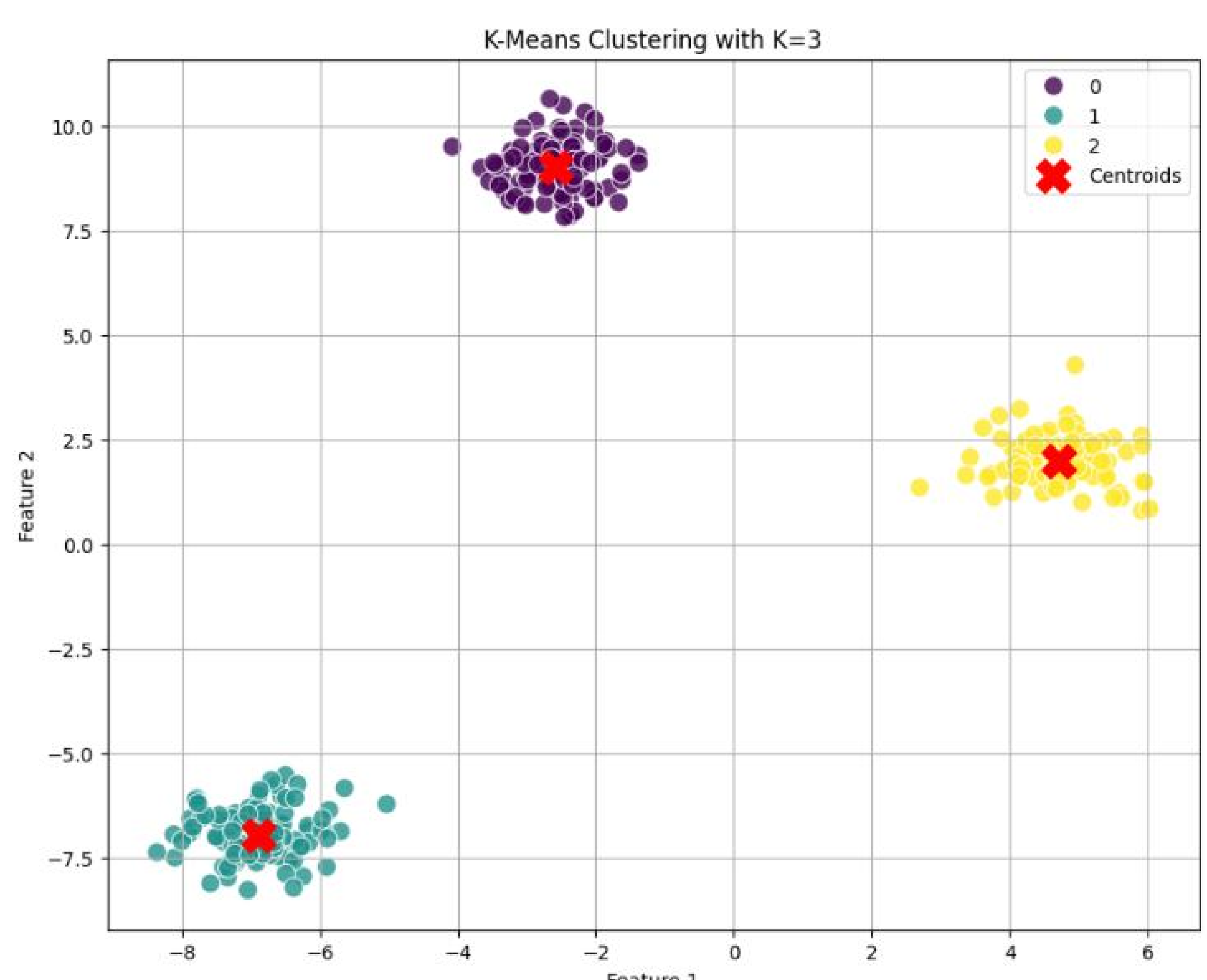
```
principal_components = pca.fit_transform(X_pca_scaled)
df_principal_components = pd.DataFrame(principal_components,
columns=['Principal_Component_1', 'Principal_Component_2'])
df_principal_components['True_Cluster'] = y_pca
explained_variance = pca.explained_variance_ratio_
print("\nPrincipal Components Head:")
print(df_principal_components.head())
print(f"\nExplained Variance Ratio: {explained_variance}")
print(f"Total Explained Variance by 2 PCs: {explained_variance.sum():.3f}")

# 4. Visualize PCA result
plt.figure(figsize=(10, 8))
sns.scatterplot(x='Principal_Component_1', y='Principal_Component_2',
hue='True_Cluster',
                data=df_principal_components, palette='Paired', s=100,
alpha=0.8)
plt.title('PCA - Dimensionality Reduction to 2 Components')
plt.xlabel(f'PC1 ({explained_variance[0]*100:.2f}%)')
plt.ylabel(f'PC2 ({explained_variance[1]*100:.2f}%)')
plt.grid(True)
plt.show()

# 5. K-Means on PCA-reduced data
kmeans_pca = KMeans(n_clusters=4, init='k-means++', max_iter=300, n_init=10,
random_state=42)
clusters_pca = kmeans_pca.fit_predict(principal_components)
df_principal_components['KMeans_Cluster_on_PCA'] = clusters_pca

plt.figure(figsize=(10, 8))
sns.scatterplot(x='Principal_Component_1', y='Principal_Component_2',
hue='KMeans_Cluster_on_PCA',
                data=df_principal_components, palette='viridis', s=100,
alpha=0.8)
plt.scatter(kmeans_pca.cluster_centers_[:, 0], kmeans_pca.cluster_centers_[:,
1], s=300, c='red', marker='X', label='Centroids')
plt.title('K-Means Clustering on PCA-Reduced Data')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.grid(True)
plt.show()

# 6. Silhouette Score for PCA-reduced KMeans
silhouette_avg_pca = silhouette_score(principal_components, clusters_pca)
```

```python
print(f"\nSilhouette Score for K-Means on PCA-Reduced Data (K=4): {silhouette_avg_pca:.3f}")
```

OUTPUT:

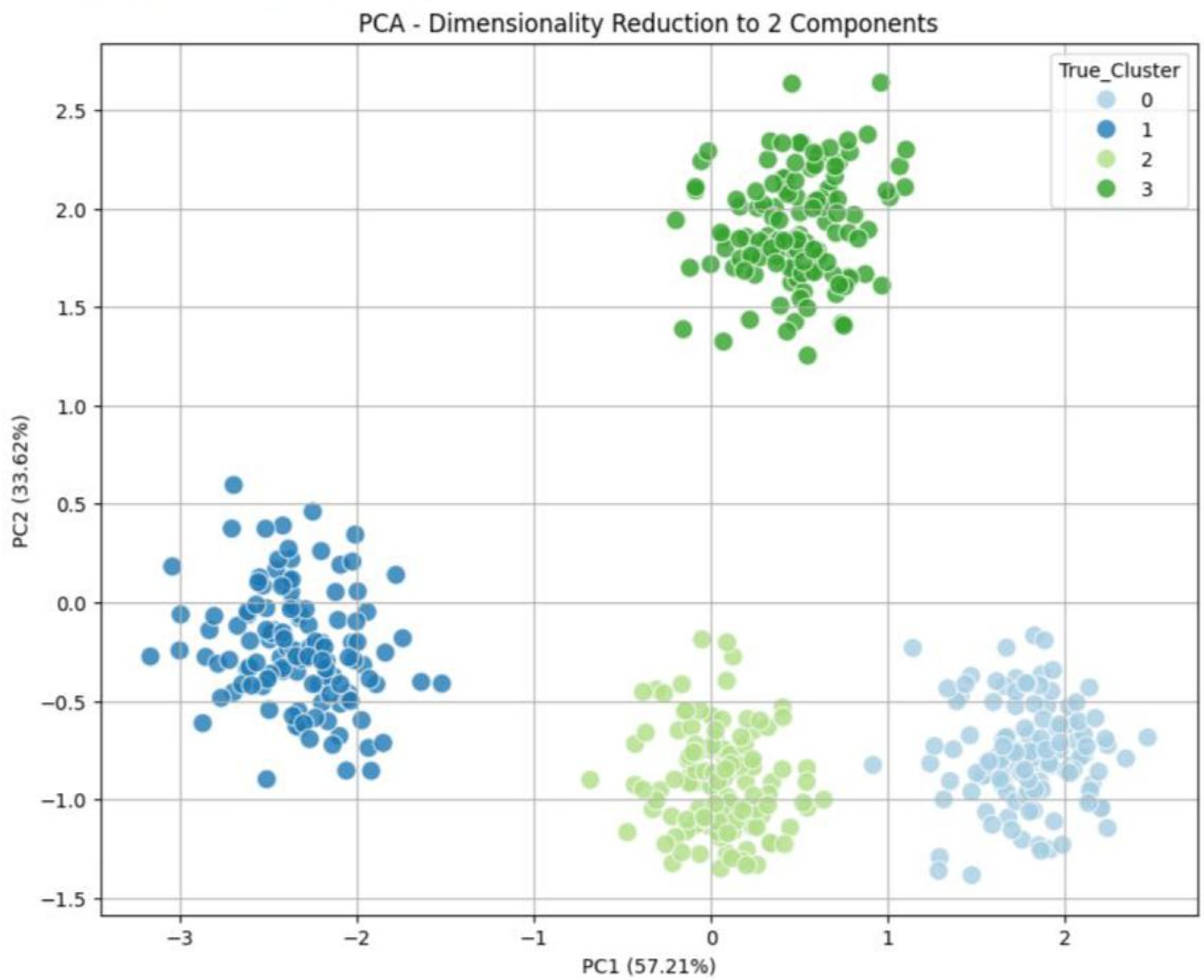
```
--- Part 1: K-Means Clustering ---

Original K-Means Dataset Head:
   Feature_1  Feature_2
0  -7.155244  -7.390016
1  -7.395875  -7.110843
2  -2.015671   8.281780
3   4.509270   2.632436
4  -8.102502  -7.484961
```

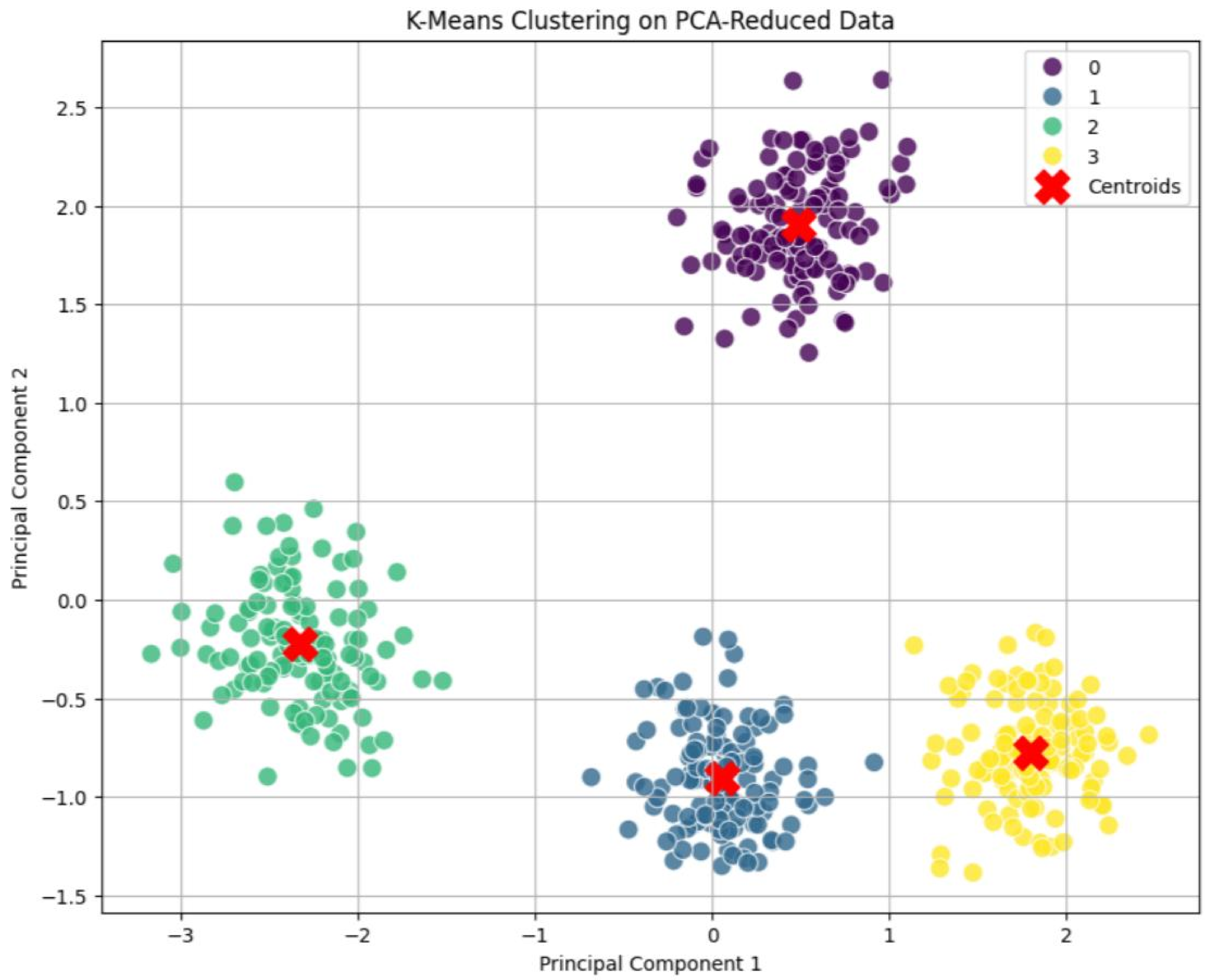





```
Silhouette Score for K-Means (K=3): 0.908
```

```
--- Part 2: Dimensionality Reduction with PCA ---

Original PCA Dataset Head:
   Feature_1  Feature_2  Feature_3  Feature_4  True_Cluster
0  -0.638667   1.110057  -6.400722  -0.204990             3
1  -2.951556  -7.657445   3.844794   0.903589             1
2  -0.253177   2.125103  -7.869801   0.559678             3
3  -2.151209   3.401400  -5.734930   0.965230             3
4  -2.347519  -7.230467   3.478891  -0.443440             1
Original PCA Dataset Shape: (500, 5)

Principal Components Head:
   Principal_Component_1  Principal_Component_2  True_Cluster
0               0.455305               1.623917             3
1              -2.705622               0.375012             1
2               0.810234               1.966926             3
3               0.427139               2.149626             3
4              -2.407508               0.099250             1

Explained Variance Ratio: [0.57208431 0.33622342]
Total Explained Variance by 2 PCs: 0.908
```






```
Silhouette Score for K-Means on PCA-Reduced Data (K=4): 0.776
```

RESULT:

Thus, the execution successfully demonstrated **Unsupervised Learning** techniques by applying **K-Means clustering** to group data points and **PCA** for dimensionality reduction and visualization, effectively revealing patterns and structure within the dataset.