|  |  |
| --- | --- |
| **EXP NO: 5** | **CLUSTERING WITH K-MEANS AND DIMENSIONALITY REDUCTION WITH PCA** |

# 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:

## 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.

## 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()

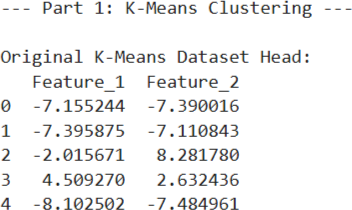
plt.grid(True) plt.show()

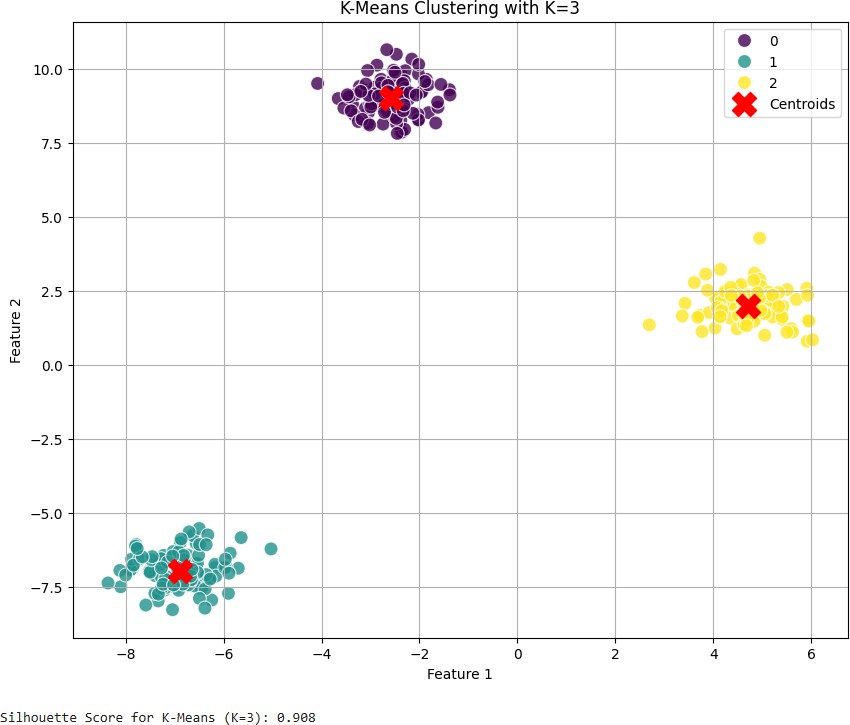
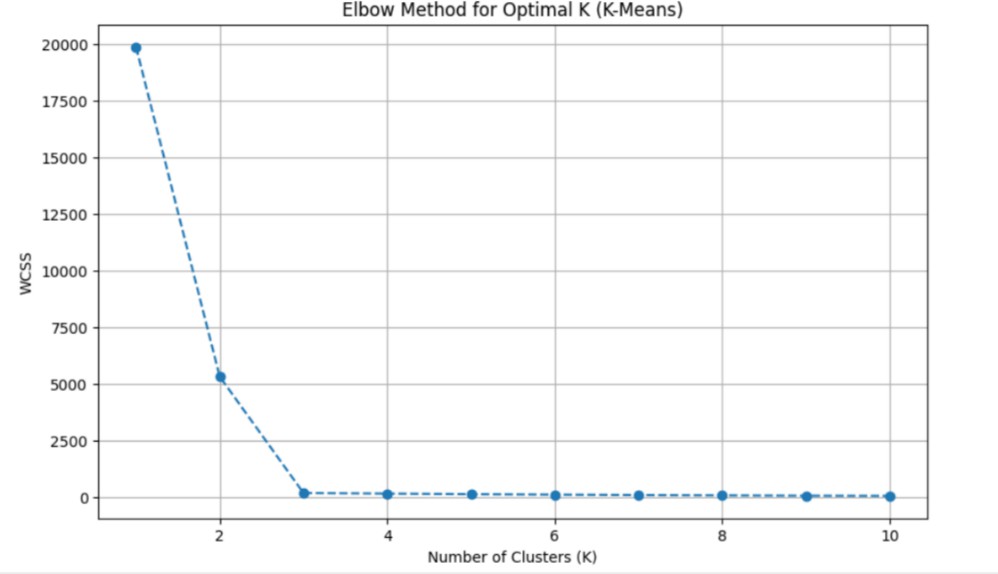
# 6. Silhouette Score for PCA-reduced KMeans

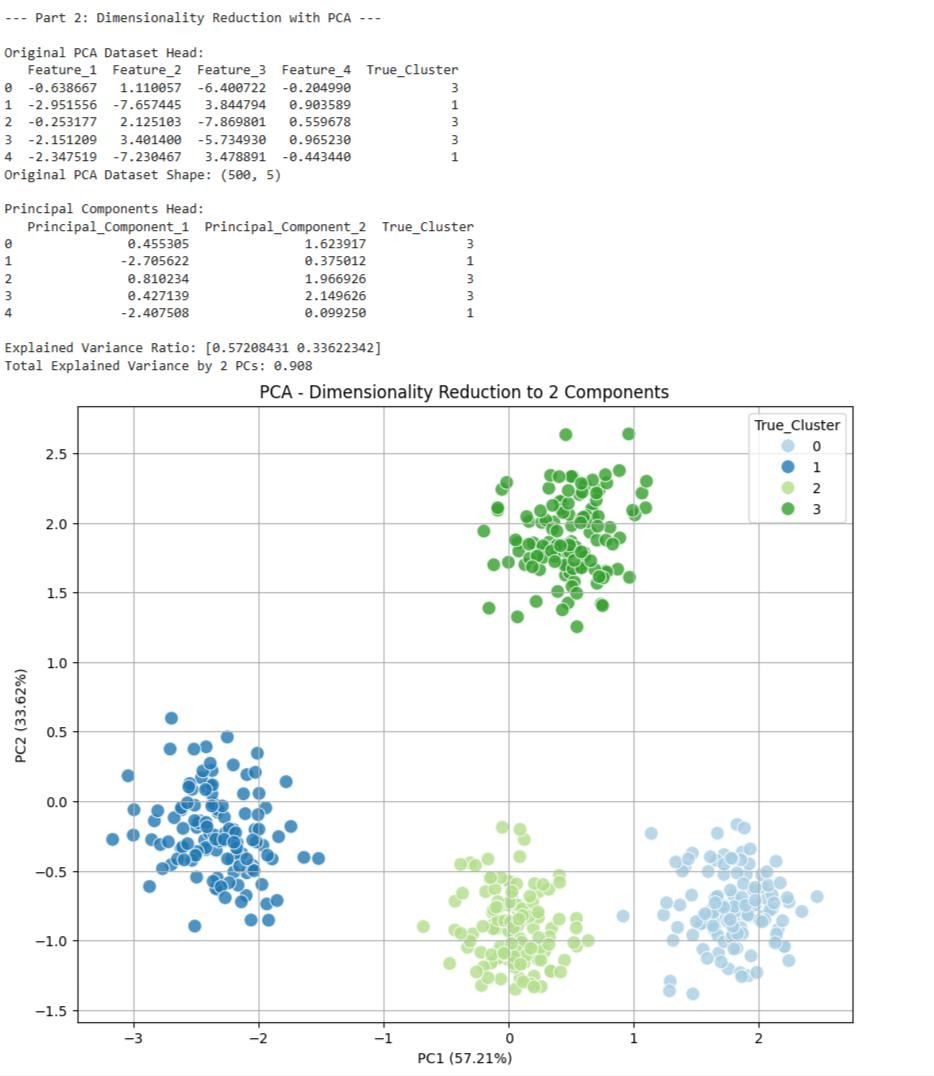
silhouette\_avg\_pca = silhouette\_score(principal\_components, clusters\_pca) print(f"\nSilhouette Score for K-Means on PCA-Reduced Data (K=4):

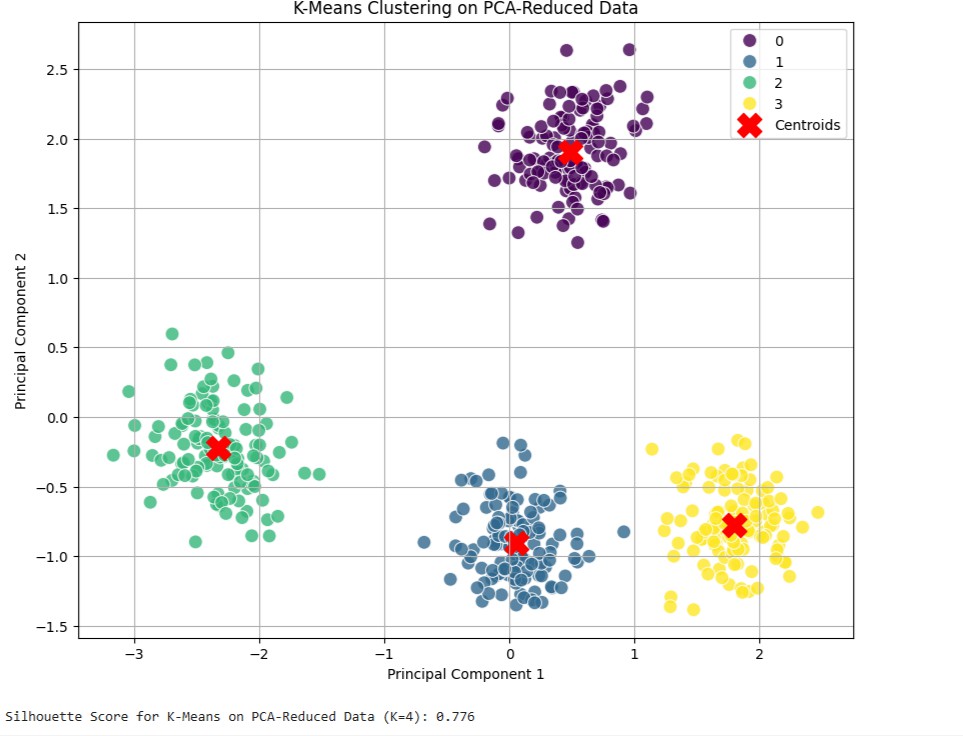
{silhouette\_avg\_pca:.3f}")

# OUTPUT:

****

****





**RESULT:**

The K-Means clustering and Principal Component Analysis (PCA) techniques were successfully implemented on the given dataset.

* **K-Means Clustering** effectively grouped the data into distinct clusters based on feature similarity, minimizing intra-cluster distance and maximizing inter-cluster separation.
* **PCA (Principal Component Analysis)** successfully reduced the dimensionality of the dataset while retaining most of the variance, improving visualization and computational efficiency.

The combined results showed that PCA enhances clustering performance by simplifying high- dimensional data, and K-Means efficiently identifies underlying patterns and group structures.