EXPERIMENT 2.1

Student Name: Samuel **UID: 24MAI10018**

Branch: ME-AIML **Section/Group: 24MAI-1**

Semester: 2 **Date of Performance: / /2025**

Subject Name: SC Lab **Subject Code: 24CSH–668**

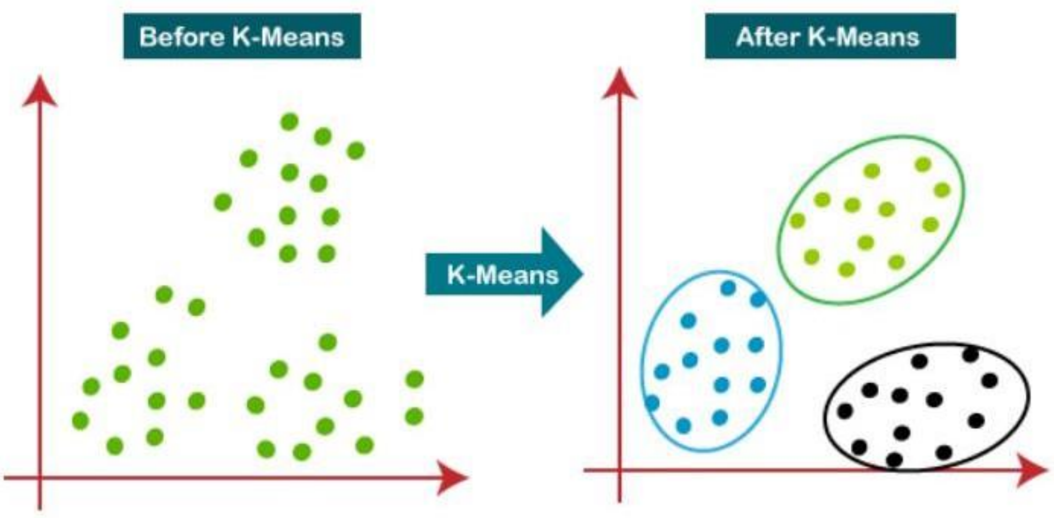
AIM: Implementation of unsupervised learning algorithm

SOFTWARE USED :JUPYTER LAB

THEORY: K-Means Clustering is a type of unsupervised machine learning algorithm used for clustering similar data points into groups or clusters. The key idea behind the K-Means algorithm is to partition data points into KKK distinct, non-overlapping subsets (clusters), where each data point belongs to the cluster whose centroid is nearest to the point.

How K-Means Works:

1. Initialize K centroids:
    - First, you select the number of clusters (K).
    - K initial centroids (randomly chosen points) are selected from the data points.
2. Assign Points to Nearest Centroid:
    - Each data point is assigned to the nearest centroid, based on the Euclidean distance between the point and the centroid.
    - This forms K clusters of data points.
3. Update Centroids:
    - After all the data points are assigned to clusters, the centroid of each cluster is recalculated.
    - The new centroid is the average of all the points in the cluster.
4. Repeat the Process:
    - Steps 2 and 3 are repeated until the centroids no longer change or the changes become very small (convergence). This indicates that the clusters have stabilized, and the algorithm has converged.



ALGORITHM:

K-Means Algorithm:

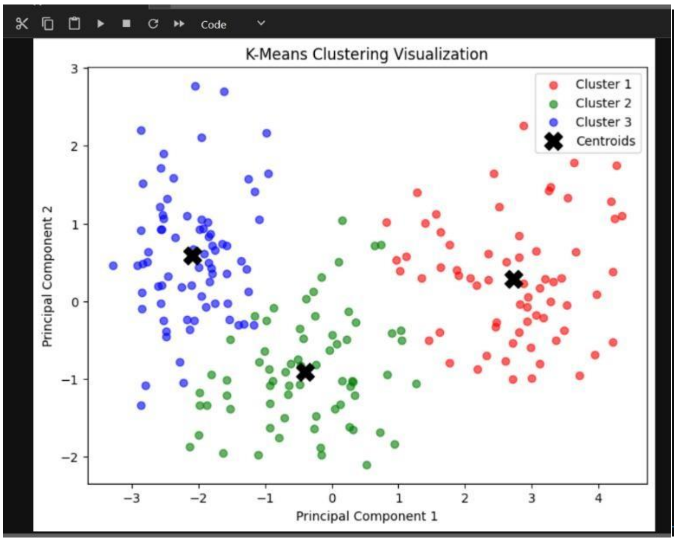
1. Initialize:
    - Choose the number of clusters KKK.
    - Randomly select KKK data points from the dataset as the initial centroids.
2. Assign Points to Clusters:
    - For each data point, compute the Euclidean distance from the point to each centroid.
    - Assign each point to the cluster whose centroid is the nearest.
3. Update Centroids:
    - For each cluster, compute the new centroid. The centroid is the mean of all data points assigned to the cluster: Centroid=1n∑i=1nxi\text{Centroid} = \frac{1}{n} \sum_{i=1}^{n} x_iCentroid=n1 i=1∑n xi where xix_ixi represents the data points assigned to the cluster, and nnn is the number of data points in that cluster.
4. Repeat:
    - Repeat steps 2 and 3 until the centroids do not change significantly or the maximum number of iterations is reached (convergence).
5. Output:
    - The algorithm terminates when the centroids stabilize, meaning there is no significant change in the centroids' positions.
    - The final clusters and their centroids are the output of the algorithm.

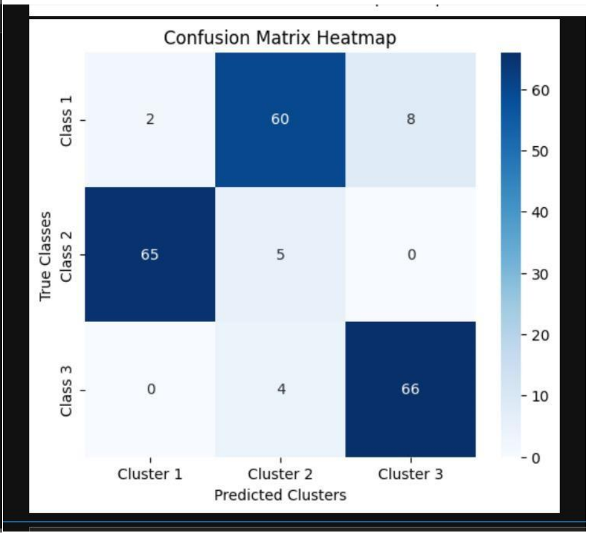
SOURCE CODE

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.metrics import confusion_matrix
my_data = np.loadtxt(r"C:\Users\ABHINAV RANA\Desktop\NLP LAB\seeds_dataset.txt")
data = my_data[:, 0:7]
labels = my_data[:, 7]
epsilon = 0.01
data_mean = np.mean(data, axis=0)
data_var = np.var(data, axis=0)
data = (data - data_mean) / np.sqrt(data_var + epsilon)
def k_means(data, clusters, num):
    indices = np.random.choice(num, clusters, replace=False)
    centers = data[indices]
    dis = np.zeros((num, clusters))
    labels = np.zeros(num, dtype=int)
    step = 0
    while True:
        prev_centers = np.copy(centers)

        # Compute distances and assign labels
        for i in range(num):
            for j in range(clusters):
```

```
            dis[i, j] = np.linalg.norm(data[i] - centers[j])
        for i in range(num):
            labels[i] = np.argmin(dis[i])
        # Update centroids
        for i in range(clusters):
            cluster_points = data[labels == i]
            if len(cluster_points) > 0:
                centers[i] = np.mean(cluster_points, axis=0)
        if np.all(centers == prev_centers):
            break
        step += 1
    return labels, centers
# Perform K-Means clustering
clusters = 3
num = data.shape[0]
cluster_labels, centers = k_means(data, clusters, num)

# Reduce dimensionality using PCA for visualization
pca = PCA(n_components=2)
data_2d = pca.fit_transform(data)
centers_2d = pca.transform(centers)

# Plot the clustered data
plt.figure(figsize=(8, 6))
colors = ['r', 'g', 'b']
for i in range(clusters):
    plt.scatter(data_2d[cluster_labels == i, 0], data_2d[cluster_labels == i, 1],
                color=colors[i], label=f'Cluster {i+1}', alpha=0.6)
plt.scatter(centers_2d[:, 0], centers_2d[:, 1], marker='X', s=200, color='black', label='Centroids')

plt.title("K-Means Clustering Visualization")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.legend()
plt.show()
true_labels = labels.astype(int) - 1  # Adjusting to zero-based index for comparison
conf_matrix = confusion_matrix(true_labels, cluster_labels)
plt.figure(figsize=(6, 5))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=[f'Cluster {i+1}' for i in
range(clusters)],
            yticklabels=[f'Class {i+1}' for i in range(clusters)])
plt.xlabel("Predicted Clusters")
plt.ylabel("True Classes")
plt.title("Confusion Matrix Heatmap")
plt.show()
```

SCREENSHOT OF OUTPUT:





LEARNING OUTCOME:

1. Understanding Clustering Algorithms – Learned how K-Means clustering groups data points based on similarity and updates centroids iteratively until convergence.
2. Data Preprocessing Techniques – Gained experience in normalizing data using mean and variance adjustments to ensure better clustering performance.
3. Dimensionality Reduction & Visualization – Applied PCA to reduce high-dimensional data to two principal components for effective visualization of clustered data.
4. Model Evaluation Using Confusion Matrix – Learned how to assess clustering performance by comparing predicted clusters with true labels using a confusion matrix and a heatmap.