Name:	Pratik Sanjay Avhad
Roll No:	01
Class/Sem:	TE/V
Experiment No.:	8
Title:	Implementation of any one clustering algorithm using languages like JAVA/ python.
Date of Performance:	
Date of Submission:	
Marks:	
Sign of Faculty:	

Aim: To Study and Implement K-Means algorithm

Objective:- Understand the working of K-Means algorithm and its implementation using python.

Theory:

In statistics and machine learning, k-means clustering is a method of cluster analysis which aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean.

Input

K:-number of clusters

D:- data set containing n objects

Output

A set of k clusters

Given k , the k-means algorithm is implemented in 5 steps:

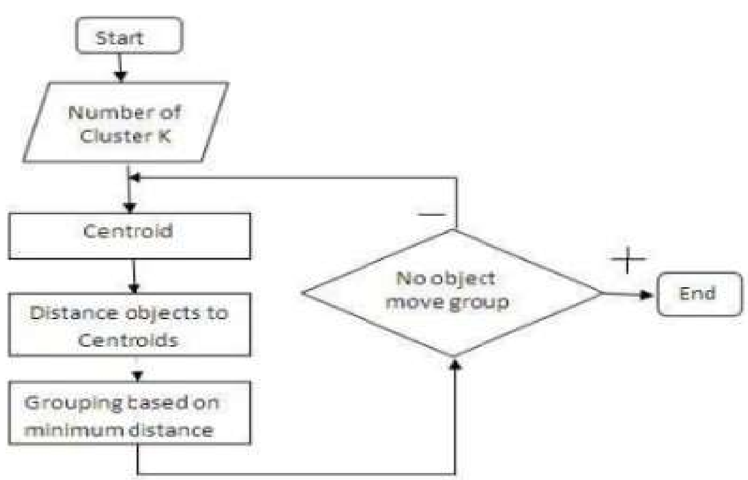
Step 1: Arbitrarily choose k objects from D as the initial cluster centers.

Step 2: Find the distance from each object in the dataset with respect to cluster centers

Step 3: Assign each object to the cluster with the nearest seed point based on the mean value of the objects in the cluster.

Step 4: Update the cluster means i.e calculate the mean value of the objects for each cluster.

Step 5: Repeat the procedure, until there is no change in meaning.



Example: d= {2,4,10,12,3,20,30,11,25} k =2

1. Randomly assign mean m1=3 and m2 = 4

Therefore, k1 = {2,3} Therefore, k1 = {4,10,12,20,30,11,25}

2. Randomly assign mean m1=2.5 and m2 = 16

Therefore, k1 = {2,3,4} Therefore, k1 = {4,10,12,20,30,11,25}

3. Randomly assign mean m1=3 and m2 = 18

Therefore, k1 = {2,3,4,10} Therefore, k1 = {12,20,30,11,25}

4. Randomly assign mean m1=7 and m2 = 25

Therefore, k1 = {2,3,4,10,11,12} Therefore, k1 =
{20,30,25}

5. Randomly assign mean m1=7 and m2 = 25

Therefore, we stop as we are getting same mean
values.

6. Therefore, Final clusters are: k1 = {2,3,4,10,11,12} Therefore, k1 = {20,30,25}

CODE:

```
import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette_score, classification_report

from sklearn.datasets import load_iris

from sklearn.impute import SimpleImputer


# Load the Iris dataset (or replace it with your dataset)

iris = load_iris()

X = iris.data  # Features

y = iris.target  # Target labels (optional, if you're doing comparison)


# Split the data into training and test sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)


# Initialize and train the K-Means model

kmeans_model = KMeans(n_clusters=len(set(y)), random_state=42)

kmeans_model.fit(X_train)
```

Predict the cluster labels on the test set

y_pred = kmeans_model.predict(X_test)

Evaluate the model using Silhouette Score (common for clustering)

sil_score = silhouette_score(X_test, y_pred)

print(f'Silhouette Score: {sil_score}')

Optionally, compare predicted clusters with true labels using a classification report

print(f'Classification Report (with original labels):\n{classification_report(y_test, y_pred)}')

Plotting the clusters (optional, useful for visualizing 2D data)

plt.scatter(X_test[:, 0], X_test[:, 1], c=y_pred, cmap='viridis')

plt.title('K-Means Clusters')

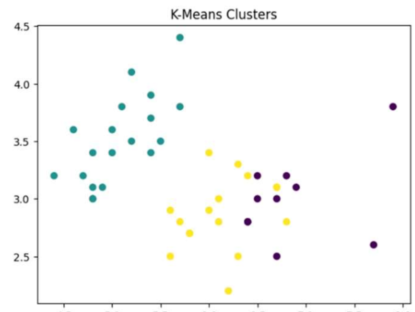
plt.show()

OUTPUT:

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
Silhouette Score: 0.5798360174465277
Classification Report (with original labels):
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        19
           1       0.00      0.00      0.00        13
           2       0.19      0.23      0.21        13

    accuracy                           0.07        45
   macro avg       0.06      0.08      0.07        45
weighted avg       0.05      0.07      0.06        45
```



CONCLUSION:

What types of data preprocessing are necessary before applying the K-Means algorithm?

Before applying the K-Means algorithm, the following data preprocessing steps are necessary:

1. **Data Cleaning**: Remove duplicates and handle missing values (imputation or removal).
2. **Feature Scaling**: Normalize or standardize features to ensure they are on the same scale, as K-Means is sensitive to the magnitude of data.
3. **Encoding Categorical Variables**: Convert categorical variables to numerical format using techniques like one-hot encoding or label encoding.
4. **Outlier Detection**: Identify and address outliers, as they can skew the results of the clustering.
5. **Dimensionality Reduction**: If necessary, apply techniques like PCA to reduce the number of features and improve clustering performance.
6. **Data Transformation**: Consider transforming features (e.g., log transformation) to achieve a more normal distribution if needed.
7. **Selection of Relevant Features**: Use feature selection techniques to keep only the most relevant features for clustering.
8. **Choosing the Number of Clusters (k)**: Use methods like the Elbow method or Silhouette score to determine an appropriate number of clusters.