Name - Gaurang A Raorane Div - D15A Roll no - 49 Batch - C

**Experiment - 7**

**AIM:** To implement different clustering algorithms.

**PROBLEM STATEMENT:**

a) Clustering algorithm for unsupervised classification (K-means, density based

(DBSCAN), Hierarchical clustering)

b) Plot the cluster data and show mathematical steps.

**Theory:-**

### **a) Clustering Algorithms for Unsupervised Classification:**

* **K-means Clustering:**
  + Theory: K-means is a centroid-based clustering algorithm that partitions data into K clusters. It iteratively assigns each data point to the nearest cluster centroid and then updates the centroid based on the mean of the points assigned to that cluster.
  + Steps:

1)Initialize K cluster centroids randomly.

2)Repeat until convergence:

* + - * Assign each data point to the nearest centroid.
      * Update the centroids based on the mean of the points assigned to each cluster.
    - Convergence occurs when the centroids no longer change significantly or after a fixed number of iterations.
  + Parameters: Number of clusters K.

**Input : -**

X = df.drop(['Date', 'District'], axis=1) # Dropping irrelevant columns

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X) # Standardizing numerical features

k = 10 # Choose the number of clusters

kmeans = KMeans(n\_clusters=k, random\_state=42)

df['Cluster'] = kmeans.fit\_predict(X\_scaled)

plt.scatter(df['Precip'], df['Temp\_2m'], c=df['Cluster'], cmap='viridis', alpha=0.5)

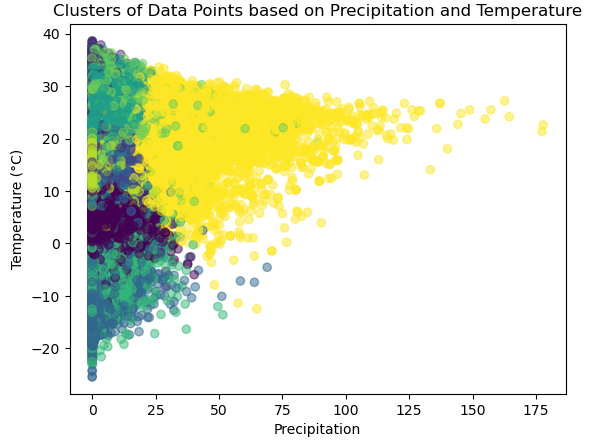
plt.xlabel('Precipitation')

plt.ylabel('Temperature (°C)')

plt.title('Clusters of Data Points based on Precipitation and Temperature')

plt.show()

**Output :-**



* **Density-based Spatial Clustering of Applications with Noise (DBSCAN):**
  + Theory: DBSCAN is a density-based clustering algorithm that groups together closely packed points based on two parameters: epsilon (ε), which defines the radius of the neighborhood around a point, and minPts, the minimum number of points required to form a dense region (core point).
  + Steps:

1)Randomly select a point not visited.

2)If it has enough neighbors within ε, mark it as a core point and expand the cluster by recursively adding its neighbors.

3)If it's not a core point but within ε of another core point, consider it part of the same cluster.

4)Repeat until all points are visited.

* + Parameters: ε (neighborhood radius), minPts (minimum number of points).

**Input:-**

import pandas as pd

from sklearn.cluster import DBSCAN

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

X = df[['Precip', 'Temp\_2m']]

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

eps = 0.5 # maximum distance between two samples for one to be considered as in the neighborhood of the other

min\_samples = 5 # number of samples in a neighborhood for a point to be considered as a core point

dbscan = DBSCAN(eps=eps, min\_samples=min\_samples)

df['Cluster'] = dbscan.fit\_predict(X\_scaled)

plt.scatter(X['Precip'], X['Temp\_2m'], c=df['Cluster'], cmap='viridis', alpha=0.5)

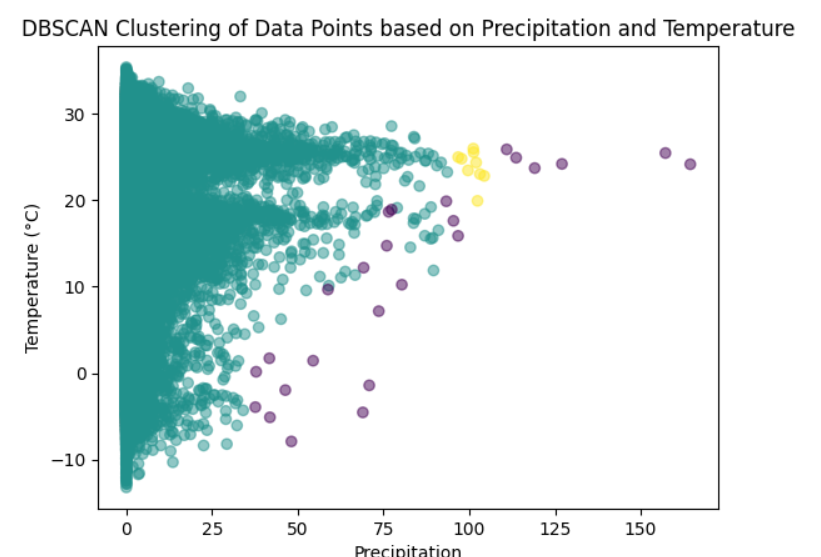
plt.xlabel('Precipitation')

plt.ylabel('Temperature (°C)')

plt.title('DBSCAN Clustering of Data Points based on Precipitation and Temperature')

plt.show()

**Output:-**

****

* **Hierarchical Clustering:**
  + Theory: Hierarchical clustering builds a hierarchy of clusters by recursively merging or splitting clusters based on their proximity. It can be agglomerative (bottom-up) or divisive (top-down).
  + Steps:

1)Start with each data point as its cluster (agglomerative) or all points in one cluster (divisive).

2)Merge or split clusters based on their distance until a single cluster (agglomerative) or individual clusters (divisive) remain.

* + Parameters: Distance metric, linkage method (e.g., single, complete, average).

### **b) Plotting Cluster Data and Mathematical Steps:**

* Scatter Plot:
  + Plot data points on a 2D or 3D scatter plot.
  + Color-code points based on their assigned cluster.
  + Axes represent feature dimensions.
* Mathematical Steps:
  + For K-means: Plot the initial cluster centroids and update them iteratively until convergence. Show how data points are assigned to clusters based on centroid proximity.
  + For DBSCAN: Illustrate how clusters are formed based on density and connectivity. Plot core points, border points, and noise points.
  + For Hierarchical Clustering: Display dendrogram to visualize the hierarchy of clusters and the order of merges or splits.
* Evaluation Metrics:
  + Silhouette Score: Measure of how similar an object is to its cluster compared to other clusters.
  + Davies-Bouldin Index: Measure of cluster separation and compactness.
  + Calinski-Harabasz Index: Measure of cluster quality based on between-cluster and within-cluster dispersion.

**Input:-**

import matplotlib.pyplot as plt

import pandas as pd

from scipy.cluster.hierarchy import dendrogram, linkage

numeric\_columns = [ 'Precip', 'Pressure']

df\_sampled = df.sample(n=20000, random\_state=42)

merg = linkage(df\_sampled[numeric\_columns], method='ward')

plt.figure(figsize=[10, 10])

dendrogram(merg, leaf\_rotation=90)

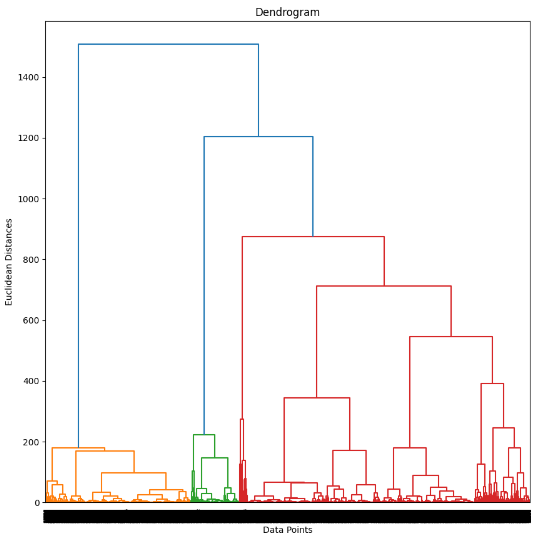
plt.title('Dendrogram')

plt.xlabel('Data Points')

plt.ylabel('Euclidean Distances')

plt.show()

**Output:-**

****

**Conclusion :-**

Through this experiment we understood K-means clustering,DBSCAN and hierarchical clustering.We successfully implemented all 3 on the dataset and gained insights about the dataset which has been helpful in grouping/segregating the dataset to parts.