**Assignment 9: Clustering**

**Name**: Jasmine Imtiyaj Sayyad

**PRN**: 22510033

**Batch**: T2

Here are the **observations** derived from the **code**, **graphs**, and **console output** you’ve provided:

### ****1.**** Elbow Method (Inertia vs K)

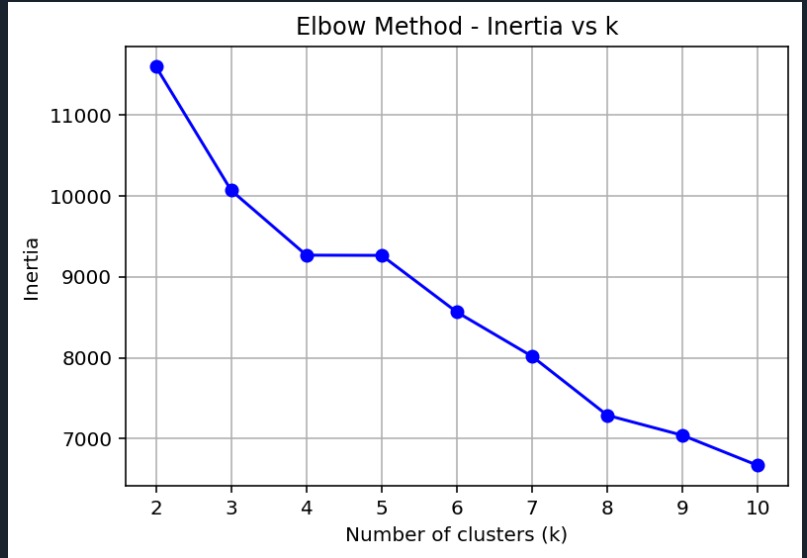
To Identify the point where adding more clusters doesn’t significantly reduce inertia (i.e., the "elbow").

**Observation from Graph:**

The elbow point appears to be at **k = 2**.

After k=2, the rate of decrease in inertia slows down.

**Conclusion:** Optimal k = 2 clusters based on the Elbow method.



### ****2.**** Silhouette Analysis (Silhouette Score vs K)

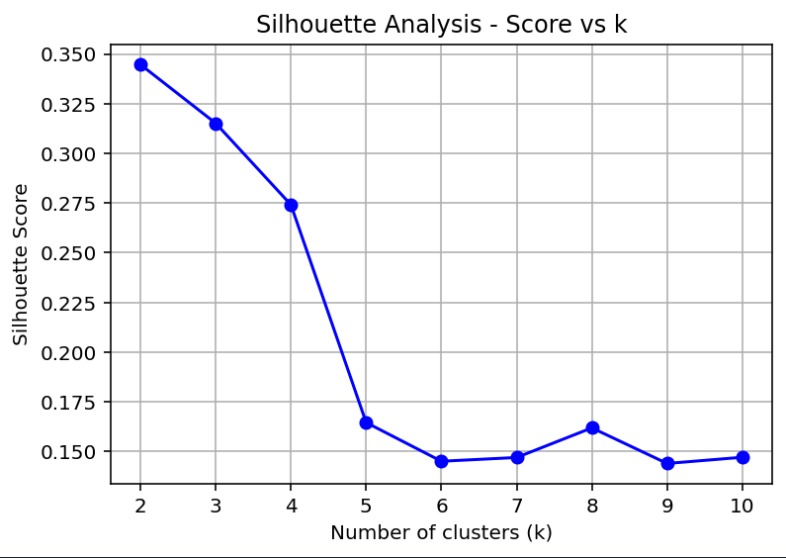
To Evaluate clustering quality. A higher silhouette score means better-defined clusters.

**Observation from Graph:**

Maximum silhouette score occurs at **k = 2** (~0.345).

Silhouette score drops sharply as k increases beyond 4.

**Conclusion:** **k = 2** is again optimal based on silhouette score, affirming the Elbow Method result

.

### ****3.**** KMeans Clustering Results

**Optimal k (from inertia):** 2

**Homogeneity Score:** 0.5249

Indicates **moderate clustering accuracy**, i.e., the clusters align decently with actual diagnosis (M or B).

**Interpretation:** KMeans with k=2 does reasonably well in separating malignant and benign tumors.

### ****4.**** DBSCAN Clustering

To Use density-based clustering without needing to specify k.

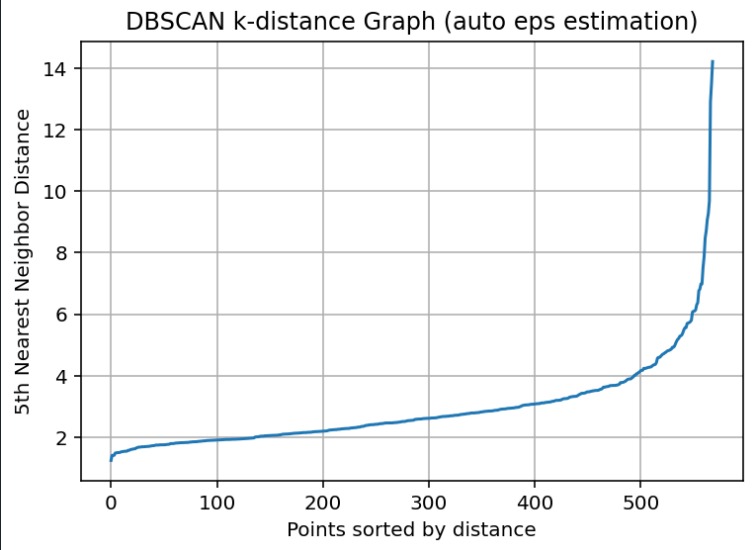
**k-distance graph:** Used to estimate a good eps value (distance threshold).

**Best Parameters (from grid search):**

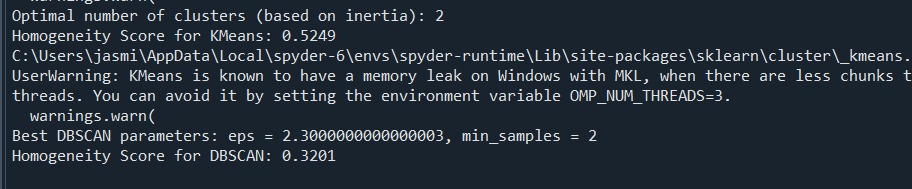
eps = 2.3, min\_samples = 2

**Homogeneity Score:** 0.3201

**Interpretation:** DBSCAN was able to find clusters but performed **worse than KMeans** in terms of homogeneity. It may not be well-suited due to the dataset’s structure or scaling.



### OUTPUT :



Code :

import pandas as pd

import numpy as np

from sklearn.cluster import KMeans, DBSCAN

from sklearn.metrics import silhouette\_score, homogeneity\_score

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

from sklearn.neighbors import NearestNeighbors

df = pd.read\_csv('Cancer\_Data.csv')

original\_diagnosis = df['diagnosis']

df = df.drop(columns=['id', 'diagnosis'])

df = df.dropna(axis=1, how='all')

df = df.fillna(df.mean())

scaler = StandardScaler()

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

inertia = []

k\_range = range(2, 11)

for k in k\_range:

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

    kmeans.fit(df\_scaled)

    inertia.append(kmeans.inertia\_)

silhouette\_scores = []

for k in k\_range:

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

    kmeans.fit(df\_scaled)

    score = silhouette\_score(df\_scaled, kmeans.labels\_)

    silhouette\_scores.append(score)

plt.plot(k\_range, inertia, 'bo-')

plt.xlabel('Number of clusters (k)')

plt.ylabel('Inertia')

plt.title('Elbow Method - Inertia vs k')

plt.grid(True)

plt.show()

plt.plot(k\_range, silhouette\_scores, 'bo-')

plt.xlabel('Number of clusters (k)')

plt.ylabel('Silhouette Score')

plt.title('Silhouette Analysis - Score vs k')

plt.grid(True)

plt.show()

optimal\_k\_inertia = k\_range[inertia.index(max(inertia))]

optimal\_k\_silhouette = k\_range[silhouette\_scores.index(max(silhouette\_scores))]

optimal\_k = optimal\_k\_inertia

print(f"Optimal number of clusters (based on inertia): {optimal\_k}")

kmeans = KMeans(n\_clusters=optimal\_k, random\_state=20)

kmeans.fit(df\_scaled)

labels = kmeans.labels\_

diagnosis\_numerical = original\_diagnosis.map({'M': 1, 'B': 0})

homogeneity = homogeneity\_score(diagnosis\_numerical, labels)

print(f"Homogeneity Score for KMeans: {homogeneity:.4f}")

min\_samples\_for\_eps = 5

neighbors = NearestNeighbors(n\_neighbors=min\_samples\_for\_eps)

neighbors\_fit = neighbors.fit(df\_scaled)

distances, indices = neighbors\_fit.kneighbors(df\_scaled)

k\_distances = np.sort(distances[:, -1])

plt.plot(k\_distances)

plt.ylabel(f'{min\_samples\_for\_eps}th Nearest Neighbor Distance')

plt.xlabel('Points sorted by distance')

plt.title('DBSCAN k-distance Graph (auto eps estimation)')

plt.grid(True)

plt.show()

eps\_values = np.arange(2.2,4.2,0.1)

min\_samples\_values = range(2,8)

best\_homogeneity = -1

best\_eps = None

best\_min\_samples = None

for eps in eps\_values:

    for min\_samples in min\_samples\_values:

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

        dbscan\_labels = dbscan.fit\_predict(df\_scaled)

        if len(set(dbscan\_labels)) > 1:

            homogeneity\_dbscan = homogeneity\_score(diagnosis\_numerical, dbscan\_labels)

            if homogeneity\_dbscan > best\_homogeneity:

                best\_homogeneity = homogeneity\_dbscan

                best\_eps = eps

                best\_min\_samples = min\_samples

print(f"Best DBSCAN parameters: eps = {best\_eps}, min\_samples = {best\_min\_samples}")

print(f"Homogeneity Score for DBSCAN: {best\_homogeneity:.4f}")