T4 22510064

PARSHWA HERWADE

ML Assignment 9 : Clustering

Download the following dataset :

<https://www.kaggle.com/datasets/erdemtaha/cancer-data/data>

1. Drop Id and Diagnosis columns
2. Cluster the data using K means clustering :
   1. Determine optimal number of clusters (between 2-10) using
      1. Elbow method with Inertia and
      2. Silhouette Analysis
   2. Use the diagnosis column to calculate homogeneity score and evaluate quality of clustering
3. Cluster the data using DB scan clustering.
   1. Determine appropriate values of eps and min\_samples parameters
   2. Use the diagnosis column to calculate homogeneity score and evaluate quality of clustering

CODE:  
  
# Import necessary libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans, DBSCAN

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

from sklearn.neighbors import NearestNeighbors

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

#############################################

# 1. Data Loading and Preprocessing

#############################################

# Read the dataset

data = pd.read\_csv("Cancer\_Data.csv")

columns\_to\_drop = []

if "id" in data.columns:

    columns\_to\_drop.append("id")

if "diagnosis" in data.columns:

    columns\_to\_drop.append("diagnosis")

if "Unnamed: 32" in data.columns:

    columns\_to\_drop.append("Unnamed: 32")

if "diagnosis" in data.columns:

    diagnosis = data["diagnosis"]

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

else:

    diagnosis = None

    diagnosis\_numeric = None

features = data.drop(columns=columns\_to\_drop, errors='ignore')

features = features.dropna()

# Standardize the features

scaler = StandardScaler()

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

#############################################

# 2. K-Means: Elbow and Silhouette on Original Data

#############################################

print("\n--- K-Means Clustering on Original Data ---")

inertias = []

silhouette\_avgs = []

range\_n\_clusters = range(2, 11)

for n\_clusters in range\_n\_clusters:

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

    labels = kmeans.fit\_predict(X\_scaled)

    inertias.append(kmeans.inertia\_)

    silhouette\_avgs.append(silhouette\_score(X\_scaled, labels))

plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)

plt.plot(range\_n\_clusters, inertias, marker='o')

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

plt.ylabel('Inertia')

plt.title('Elbow Method (Original Data)')

plt.subplot(1, 2, 2)

plt.plot(range\_n\_clusters, silhouette\_avgs, marker='o', color='red')

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

plt.ylabel('Silhouette Score')

plt.title('Silhouette Analysis (Original Data)')

plt.tight\_layout()

plt.show()

kmeans\_optimal = KMeans(n\_clusters=2, random\_state=42)

kmeans\_labels = kmeans\_optimal.fit\_predict(X\_scaled)

if diagnosis\_numeric is not None:

    score = homogeneity\_score(diagnosis\_numeric.loc[features.index], kmeans\_labels)

    print(f"K-Means Homogeneity Score (Original Data): {score:.4f}")

#############################################

# 3. DBSCAN on Original Data

#############################################

print("\n--- DBSCAN Clustering on Original Data ---")

neighbors = 5

neigh = NearestNeighbors(n\_neighbors=neighbors)

nbrs = neigh.fit(X\_scaled)

distances, \_ = nbrs.kneighbors(X\_scaled)

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

plt.figure(figsize=(8, 4))

plt.plot(kth\_distances)

plt.xlabel("Sorted Data Points")

plt.ylabel(f"{neighbors}-th NN Distance")

plt.title("K-distance Graph (Original Data)")

plt.show()

best\_score = -1

best\_params = {}

for eps in np.linspace(0.5, 2.0, 16):

    for min\_samples in range(3, 10):

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

        labels = dbscan.fit\_predict(X\_scaled)

        unique\_labels = set(labels)

        if len(unique\_labels) <= 1 or (len(unique\_labels) == 1 and -1 in unique\_labels):

            continue

        score = homogeneity\_score(diagnosis\_numeric.loc[features.index], labels)

        if score > best\_score:

            best\_score = score

            best\_params = {'eps': eps, 'min\_samples': min\_samples}

if best\_params:

    dbscan = DBSCAN(\*\*best\_params)

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

    n\_clusters = len(set(dbscan\_labels)) - (1 if -1 in dbscan\_labels else 0)

    print(f"Best DBSCAN (Original): eps={best\_params['eps']:.2f}, min\_samples={best\_params['min\_samples']}")

    print(f"DBSCAN found {n\_clusters} clusters. Homogeneity Score: {best\_score:.4f}")

else:

    print("No valid DBSCAN clustering found on original data.")

#############################################

# 4. PCA Transformation

#############################################

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X\_scaled)

print("\nExplained Variance Ratio (PCA):", pca.explained\_variance\_ratio\_)

print("Total Variance Explained: {:.2f}%".format(100 \* np.sum(pca.explained\_variance\_ratio\_)))

if diagnosis\_numeric is not None:

    plt.figure(figsize=(7, 5))

    plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c=diagnosis\_numeric.loc[features.index], cmap='viridis', edgecolor='k')

    plt.xlabel("PCA 1")

    plt.ylabel("PCA 2")

    plt.title("PCA Projection (True Labels)")

    plt.colorbar(label='Diagnosis (0: Benign, 1: Malignant)')

    plt.show()

#############################################

# 5. K-Means on PCA Data

#############################################

print("\n--- K-Means Clustering on PCA Data ---")

kmeans\_pca = KMeans(n\_clusters=2, random\_state=42)

kmeans\_pca\_labels = kmeans\_pca.fit\_predict(X\_pca)

if diagnosis\_numeric is not None:

    score = homogeneity\_score(diagnosis\_numeric.loc[features.index], kmeans\_pca\_labels)

    print(f"K-Means Homogeneity Score (PCA): {score:.4f}")

plt.figure(figsize=(7, 5))

plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c=kmeans\_pca\_labels, cmap='viridis', edgecolor='k')

plt.xlabel("PCA 1")

plt.ylabel("PCA 2")

plt.title("K-Means Clusters (PCA)")

plt.show()

#############################################

# 6. DBSCAN on PCA Data

#############################################

print("\n--- DBSCAN Clustering on PCA Data ---")

neighbors = 5

neigh = NearestNeighbors(n\_neighbors=neighbors)

nbrs = neigh.fit(X\_pca)

distances, \_ = nbrs.kneighbors(X\_pca)

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

plt.figure(figsize=(8, 4))

plt.plot(kth\_distances)

plt.xlabel("Sorted Points")

plt.ylabel(f"{neighbors}-th NN Distance")

plt.title("K-distance Graph (PCA Data)")

plt.show()

best\_dbscan\_score = -1

best\_dbscan\_params = {}

for eps in np.linspace(0.1, 1.5, 15):

    for min\_samples in range(2, 10):

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

        labels = dbscan.fit\_predict(X\_pca)

        unique\_labels = set(labels)

        if len(unique\_labels) < 2 or (len(unique\_labels) == 1 and -1 in unique\_labels):

            continue

        score = homogeneity\_score(diagnosis\_numeric.loc[features.index], labels)

        if score > best\_dbscan\_score:

            best\_dbscan\_score = score

            best\_dbscan\_params = {"eps": eps, "min\_samples": min\_samples}

if best\_dbscan\_params:

    dbscan = DBSCAN(\*\*best\_dbscan\_params)

    labels = dbscan.fit\_predict(X\_pca)

    n\_clusters = len(set(labels)) - (1 if -1 in labels else 0)

    print(f"Best DBSCAN (PCA): eps={best\_dbscan\_params['eps']:.2f}, min\_samples={best\_dbscan\_params['min\_samples']}")

    print(f"DBSCAN found {n\_clusters} clusters. Homogeneity Score: {best\_dbscan\_score:.4f}")

    plt.figure(figsize=(7, 5))

    plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c=labels, cmap='viridis', edgecolor='k')

    plt.xlabel("PCA 1")

    plt.ylabel("PCA 2")

    plt.title("DBSCAN Clustering (PCA)")

    plt.show()

else:

    print("No valid DBSCAN clustering found on PCA data.")

#############################################

# 7. Observations:

#############################################

print("\n#############################################")

print("# Observations:")

print("#############################################")

print("# 1. K-Means Clustering on Original Data:")

print("#    - The elbow plot shows that inertia decreases as the number of clusters increases,")

print("#      while the silhouette plot peaks near k=2, suggesting an optimal choice for two clusters.")

print("#    - The homogeneity score indicates moderate alignment between the K-Means clusters and")

print("#      the true diagnosis labels in the original data.")

print("")

print("# 2. DBSCAN Clustering on Original Data:")

print("#    - The k-distance graph assists in choosing an appropriate eps value; the knee of the graph")

print("#      serves as a visual cue for selecting eps.")

print("#    - DBSCAN labels noise points (typically as -1), and the resulting homogeneity score")

print("#      reflects how purely the clusters correspond to the true classes.")

print("")

print("# 3. PCA Transformation and Clustering:")

print("#    - PCA reduces the dimensionality and captures a significant amount of the data variance,")

print("#      allowing clearer visualization of the cluster structure with true labels.")

print("#    - K-Means on PCA data yields a higher or comparable homogeneity score, with well-separated")

print("#      clusters visible in the 2D projection.")

print("#    - DBSCAN applied in the PCA space benefits from reduced noise and dimensionality,")

print("#      resulting in more interpretable clusters based on the tuning of eps and min\_samples.")

print("")

print("# Overall, the comparison between clustering on the original data and PCA-transformed data")

print("# demonstrates that PCA can enhance cluster separation and improve the alignment of clusters")

print("# with the underlying diagnosis labels. Fine-tuning parameters in both methods further")

print("# helps in achieving better clustering performance.")

print("#############################################")











