Ex12:

Name: R.Athithraja

Reg.no:2022503702

1)

Code:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

df = pd.read\_csv("hd.csv")

#print(df)

# Remove traget

X=df.drop(columns=['target'])

#print(X)

scaler = StandardScaler()

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

#print(X\_scaled)

kmeans = KMeans(n\_clusters=2, random\_state=42, n\_init=10)

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

print("cluster: \n"+str(clusters))

df['Cluster'] = clusters

print("\nData: \n"+str(df.tail(10)))

"""

pca = PCA(n\_components=2)

#print(pca)

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

#print(X\_pca)

plt.figure(figsize=(8,6))

sns.scatterplot(x=X\_pca[:, 0], y=X\_pca[:, 1], hue=clusters, palette="viridis", alpha=0.7)

plt.title("Heart Disease Clustering (K-Means, Without Target)")

plt.xlabel("Principal Component 1")

plt.ylabel("Principal Component 2")

plt.legend(title="Cluster", labels=["Cluster 0", "Cluster 1"])

plt.show()

"""

print("\nCluster counts:\n", df['Cluster'].value\_counts())

print("\nTarget counts:\n", df['target'].value\_counts())

print("\n")

Output:

cluster:

[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1

1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 1 0 0 0 0 1 1 0

1 0 0 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 1 0 1 0 0 0 0 1 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0

1 0 1 0 0 0 0 0 1 0 0 1 1 0 1 0 1 0 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0 0 1 0 0

0 0 0 1 0 0 1]

Data:

age sex cp trestbps chol fbs restecg thalach exang oldpeak \

293 67 1 2 152 212 0 0 150 0 0.8

294 44 1 0 120 169 0 1 144 1 2.8

295 63 1 0 140 187 0 0 144 1 4.0

296 63 0 0 124 197 0 1 136 1 0.0

297 59 1 0 164 176 1 0 90 0 1.0

298 57 0 0 140 241 0 1 123 1 0.2

299 45 1 3 110 264 0 1 132 0 1.2

300 68 1 0 144 193 1 1 141 0 3.4

301 57 1 0 130 131 0 1 115 1 1.2

302 57 0 1 130 236 0 0 174 0 0.0

slope ca thal target Cluster

293 1 0 3 0 1

294 0 0 1 0 0

295 2 2 3 0 0

296 1 0 2 0 0

297 1 2 1 0 0

298 1 0 3 0 0

299 1 0 3 0 1

300 1 2 3 0 0

301 1 1 3 0 0

302 1 1 2 0 1

Cluster counts:

Cluster

1 198

0 105

Name: count, dtype: int64

Target counts:

target

1 165

0 138

Name: count, dtype: int64

2)

Code:

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

wcss = []

for k in range(1, 11):

kmeans = KMeans(n\_clusters=k, random\_state=42, n\_init=10)

kmeans.fit(X\_scaled)

wcss.append(kmeans.inertia\_)

#print(wcss)

plt.figure(figsize=(8,5))

plt.plot(range(1, 11), wcss, marker='o', linestyle='-')

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

plt.ylabel("Inertia (WCSS)")

plt.title("Elbow Method to Determine Optimal k")

plt.show()

print(wcss)

#this code is used to find the big drop in graph to find k value

dif=[]

for i in range((len(wcss))):

if i+1!=len(wcss):

dif.append(wcss[i]-wcss[i+1])

print(dif)

big=0

bigindex=None

for i in range(0,len(dif)):

if dif[i]>big:

big=dif[i]

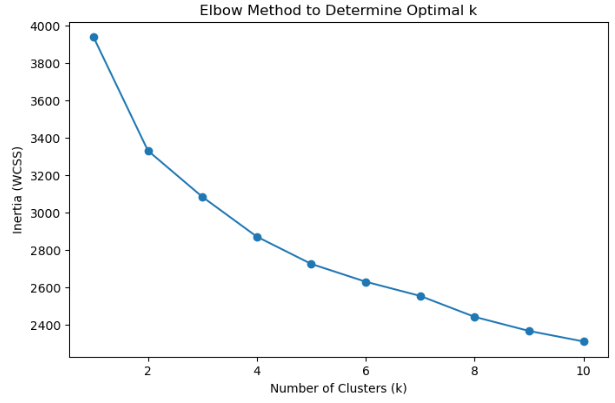
bigindex=i+2 #+2 is to match the graph k value

print(big)

print("k value="+str(bigindex))

print("when the graph drop suddenly and start linear way that point is optimal k")

Output:



[3938.9999999999995, 3330.76480686847, 3083.355474075446, 2870.5344628399125, 2724.422251012436, 2629.210291307322, 2553.2434149605165, 2441.1521637100527, 2365.693238342787, 2309.460702051542]

[608.2351931315297, 247.40933279302362, 212.82101123553366, 146.11221182747659, 95.21195970511371, 75.96687634680575, 112.09125125046376, 75.45892536726569, 56.2325362912452]

608.2351931315297

k value=2

when the graph drop suddenly and start linear way that point is optimal k

3)

Code:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

df = pd.read\_csv("hd.csv")

#print(df)

# Remove traget

X=df.drop(columns=['target'])

#print(X)

scaler = StandardScaler()

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

#print(X\_scaled)

kmeans = KMeans(n\_clusters=2, random\_state=42, n\_init=10)

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

#print("cluster: \n"+str(clusters))

df['Cluster'] = clusters

print("\nCluster counts:\n", df['Cluster'].value\_counts())

print("\nTarget counts:\n", df['target'].value\_counts())

print("\n")

inertia = kmeans.inertia\_

print(f"Inertia (WCSS): {inertia:.2f}")

silhouette = silhouette\_score(X\_scaled, clusters)

print(f"Silhouette Score: {silhouette:.2f}")

if 'target' in df.columns:

y\_true = df['target']

ari = adjusted\_rand\_score(y\_true, clusters)

print(f"Adjusted Rand Index (ARI): {ari:.2f}")

print("Confusion Matrix:\n", confusion\_matrix(y\_true, clusters))

Output:

Cluster counts:

Cluster

1 198

0 105

Name: count, dtype: int64

Target counts:

target

1 165

0 138

Name: count, dtype: int64

Inertia (WCSS): 3330.76

Silhouette Score: 0.17

Adjusted Rand Index (ARI): 0.39

Confusion Matrix:

[[ 93 45]

[ 12 153]]